NXP Semiconductors

Document Number: MPC5746RRM
Rev. 6.2, 06/2017

MPC5746R Reference Manual

This is the MPC5746R Reference Manual set, consisting of the following files:

- MPC5746R Reference Manual Addendum, Rev. 3
- MPC5746R Reference Manual Addendum, Rev. 2
- MPC5746R Reference Manual, Rev. 6

.

MPC5746R Reference Manual Addendum

Updates to the MPC5746R Reference Manual, Revision 6

Document Number: MPC5746RRMAD
Rev. 3, 06/2017

Contents

Section number **Title** **Page**

Chapter 1
Addendum for Rev 6 of MPC5746R

1.1 Overview....7

1.2 Introduction....11

1.2.1 Block diagrams.... 11

1.2.2 Feature list....13

1.3 Standby SRAM Regulator.... 14

1.4 INTC Master Protection Register (INTC_MPROT)....16

1.5 Clocking....18

1.5.1 Clock generation 18

1.5.2 Peripheral Clocks 20

1.6 CMU Control Status Register (CMU_CSR)....21

1.7 Aux Clock 0 Divider Configuration 4 (MC_CGM_AC0_DC4).... 23

1.8 Software Watchdog Timer (SWT)....24

1.8.1 Features.... 24

1.8.2 SWT Control Register (SWT_CR).... 24

1.8.3 Time-out....27

1.9 Enhanced Modular IO Subsystem (eMIOS200)....27

1.9.1 Modes of operation.... 27

1.9.2 eMIOS UC B register n (eMIOS_Bn)....28

1.9.3 eMIOS CNT register n (eMIOS_CNTn).... 29

1.9.4 eMIOS UC Control register n (eMIOS_Cn)....30

1.10 FEC interface selection....35

1.11 SENT Receiver (SRX)....36

1.11.1 Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CHn_CONFIG)....36

1.11.2 Input programmable filter.... 40

1.11.3 Pause pulse diagnostic.... 40

Section number	Title	Page

1.12 SIPI Error Register (SIPI_ERR)..... 44

1.13 PLL configuration example..... 46

1.14 LINFlexD..... 47

1.14.1 Introduction..... 47

1.14.2 LIN mode features..... 47

1.14.3 DMA Tx Enable Register (LINFlexD_DMATXE)..... 48

1.14.4 DMA Rx Enable Register (LINFlexD_DMARXE)..... 49

1.15 'Destructive' Bidirectional Reset Enable (MC_RGM_DBRE)..... 50

1.16 Power Management..... 53

1.16.1 Voltage Monitoring..... 53

1.16.2 ADC Interface..... 53

1.17 Power Management Controller..... 53

1.17.1 PMC introduction..... 53

1.17.2 Reset Enable Control Register (PMC_REE)..... 53

1.17.3 Reset Type (Destructive / Functional) Selection Register (PMC_RES)..... 56

1.17.4 ADC Channel Select Register (PMC_ADC_CS)..... 59

1.17.5 ADC Interface..... 60

1.18 Power Control Unit (MC_PCU)..... 62

1.18.1 Introduction..... 62

1.18.2 Functional description..... 62

1.19 NXMC block connections..... 62

1.20 Power..... 64

1.21 Fault Collection and Control Unit (FCCU)..... 64

1.21.1 Available FCCU output signals..... 64

1.21.2 FCCU failure inputs from other modules..... 64

1.22 Self-Test Control Unit (STCU2)..... 73

1.22.1 STCU2 chip-specific module information..... 73

1.22.2 STCU2 Off-Line MBIST Status Medium Register (STCU2_MBSM)..... 74

1.23 PMC protected registers..... 76

1.24 BAR execution protection....77
1.25 Password and Device Security Module (PASS)....77
1.25.1 Password and Device Security Module (PASS) chip-specific module information....77
1.25.2 Life Cycle Status Register (PASS_LCSTAT)....79
1.25.3 Debug Disable (DD)....81

Chapter 1
Addendum for Rev 6 of MPC5746R

1.1 Overview

This addendum describes corrections or updates to the MPC5746R Reference Manual. The currently available version of the MPC5746R Reference Manual is Revision 6.

Navigation tip: When navigating this PDF file using the hyperlinks, pressing the ALT and "left arrow" keys simultaneously will return you to the previously viewed page.

Table 1-1. MPC5746RRM Rev6 Addendum

Location	Description
Chapter 2, Introduction	
Page 99	• In "Peripherals allocation" figure, changed PMC_DIG to PMC. • See Block diagrams

Page 102	• In "Feature list" table, updated the following:
	  • Under MPC5746R Column, for Core Frequency row, changed “200MHz" to “200/150MHz)"
	  • Under MPC5745R Column, for Core Frequency row, changed “150MHz (Note 2)" to “200/150MHz"
	  • Under MPC5746R Column, for DSPI row, changed “5+2MSB Support" to “5+2MSB Support (Note 4)"
	  • Under MPC5746R Column, for Package row, changed “MAPBGA252" to “LQFP176 or MAPBGA252"
	• See Feature list

Chapter 3, Embedded memories	
Page 106	• In "Standby SRAM Regulator" section, changed "description in the Power Management Controller digital interface (PMC_dig)" to "description in the Power Management Controller (PMC). • See Standby SRAM Regulator
Chapter 17, Interrupt Controller (INTC)	
Page 446	• Updated last sentence in INTC_MPROT[MPROT] field description from "Once MPROT is enabled by a master, only that master may modify INTC_MPROT" to "Once MPROT is enabled for a master, only that master may modify INTC_MPROT" • See INTC Master Protection Register (INTC_MPROT)
Chapter 22, Clocking	

Table continues on the next page...

Table 1-1. MPC5746RRM Rev6 Addendum (continued)

Location	Description
Page 643	• In Clock distribution figure, changed SIUL2_SCR0[FEC_MODE] as selector for TXCLK_REF_CLK instead of IOMUX. • See Clock generation
Page 655	• In Clock generation figure, changed the name of eTPU2_CLK to eTPU_CLK, eTPU2+ to eTPU, REACT to REACM2, and GLITCH_FILTER to IGF. • See Peripheral Clocks
Chapter 24, Clock Monitor Unit (CMU)	
Page 690	• In CSR[SFM] field, added the following note "MSR[MD] must be written before enabling frequency measurement (CSR[SFM] = 1). Do not write MDR[MD] while CSR[SFM] = 1." • In CSR[RCDIV] field, added the following note "Ensure clock source for CLKMN0_RMT is off before writing CMU_CSR[RCDIV]". • See CMU Control Status Register (CMU_CSR)
Chapter 25, Clock Generation Module (MC_CGM)	
Page 723	• In Aux Clock 0 Divider Configuration 4 register, changed the width of MC_CGM_AC0_DC4[DIV] from 6 to 4. • See Aux Clock 0 Divider Configuration 4 (MC_CGM_AC0_DC4)
Chapter 40, Software Watchdog Timer (SWT)	
Page 1203, 1205, and 1215	• In the "Features" section, changed "Programmable selection of reset or interrupt..." to "Programmable selection of reset request or interrupt..." in list item. • See Features • In the "SWT_CR field descriptions" table, revised all the MAPn field descriptions. • In "SWT_CR[ITR]" field description, changed the spelled-out name from "Interrupt Then Reset" to "Interrupt Then Reset Request". • See SWT Control Register (SWT_CR). • In "Time-out" section, changed "The Interrupt Then Reset bit (SWT_CR[ITR])..." to "The SWT_CR[ITR] bit..." in first sentence. • See Time-out
Chapter 42 Enhanced Modular IO Subsystem (eMIOS200)	

Page 1239, 1249, 1250, 1251 and 1277	• Deleted "QDEC" reference from the following sections:
	  • Modes of operation
	  • eMIOS UC B register n (eMIOS_Bn)
	  • eMIOS CNT register n (eMIOS_CNTn)
	  • Deleted "Quadrature Decode (QDEC) Mode" section.
	• See Modes of operation, eMIOS UC B register n (eMIOS_Bn), eMIOS CNT register n (eMIOS_CNTn)
	• In eMIOS_Cn[EDPOL] bit description deleted the notes and "QDEC" reference
	• See eMIOS UC Control register n (eMIOS_Cn)

Chapter 50, SIPI	
Page 2130	• In SIPI_ERR[TOEx] field description, changed "0 Timeout error occurred 1 Timeout error didn't occur" to "0 Timeout error didn't occur 1 Timeout error occurred" • See SIPI Error Register (SIPI_ERR)
Chapter 51, LVDS Fast Asynchronous Serial Transmission (LFAST) – Interprocessor Communications	
Page 2220	• Added the section "PLL Configuration example". • See PLL configuration example
Chapter 52, Fast Ethernet Controller (FEC)	
Page 2221	• Added a note regarding the recommended clock setup before FEC is operational. • See FEC interface selection
Chapter 53, SENT Receiver (SRX)	

Table continues on the next page...

Table 1-1. MPC5746RRM Rev6 Addendum (continued)

Location	Description
Page 2342, 2359, and 2363	• In "Channel 'n' Configuration Register" changed the description of FIL_CNT field from "Only one bit should be set in this field and the output of filter will be offset by twice the same number of clocks plus additional 3 clocks due to synchronization. Default value is '4'" to "Only one bit should be set in this field. Output of the filter is equal to 2*FIL_CNT +4". • See Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CHn_CONFIG) • In "Input programmable filter" section, changed the FIL_CNT value to 1. • See Input programmable filter • In "Pause pulse diagnostic" section, added figure SENT receiver resynchronization. • See Pause pulse diagnostic
Chapter 54, LINFlexD	
Page 2372 and 2374	• In "Introduction" and "LIN mode features" sections, added support for LINFlexD protocol feature 2.2. • See Introduction and LIN mode features
Page 2460	• In DMA Tx Enable Register, updated the note in Reserved field from "The number of reserved.....TX_CH_NUM used in this device" to "The number of reserved.....DMA_TX_CH_NUM used in this device" and changed the note in LINFlexD_DMATXE[DTE] field from "The actual size of the register.....(soft reset) in Idle state" to "The number of DTE bit.... (soft reset) in Idle state". • See DMA Tx Enable Register (LINFlexD_DMATXE) • In DMA Rx Enable Register, updated the note in Reserved field from "The number of reserved.....RX_CH_NUM used in this device" to "The number of reserved.....DMA_RX_CH_NUM used in this device" and changed the note in LINFlexD_DMARXE[DRE] field from "The actual size of the register.....(soft reset) in Idle state" to "The number of DTE bit.... (soft reset) in Idle state". • See DMA Rx Enable Register (LINFlexD_DMARXE)
Chapter 56, Reset Generation Module (MC_RGM)	
Page 2496	• In 'Destructive' Bidirectional Reset Enable register, changed the access of MC_RGM_DBRE[BE_PMC_STF] from R/W to RO. • See 'Destructive' Bidirectional Reset Enable (MC_RGM_DBRE)
Chapter 59, Power Management	
Page 2562 and 2567	• In "Voltage Monitoring" section, changed "descriptions in the PMC_dig" to "descriptions in the PMC". • See Voltage Monitoring and ADC Interface • In "ADC interface" section, deleted the table ADC interface. • See ADC Interface
Chapter 60, Power Management Controller (PMC)	
Page 2572, 2592, 2595 and 2599	• In "PMC introduction" section, changed "The PMC_dig block contains" to "The PMC block contains". • See PMC introduction • In "PMC_REE", "PMC_RES", and "PMC_ADC_CS" registers, changed the reset footnote from "The reset value is chip-specific. Refer to the PMC_dig chip-specific module information for the reset value" to "The reset value is chip-specific. Refer to the See the chip-specific PMC information for the reset value". • See Reset Enable Control Register (PMC_REE), Reset Type (Destructive / Functional) Selection Register (PMC_RES), and ADC Channel Select Register (PMC_ADC_CS)
Chapter 61, Power Control Unit (MC_PCU)	
Page 2607 and 2608	• In "Introduction" section, changed "This microcontroller has.....control the analog PMC" to "This microcontroller has.....control the PMC.". • See Introduction

Table continues on the next page...

Table 1-1. MPC5746RRM Rev6 Addendum (continued)

Location	Description
	• In "Functional description" section, changed "PMC_dig" to "PMC". • See Functional description
Chapter 63, Calibration and Debug	

Page 2762	• In "NXMC block connections" section, added the following footnotes:
	  • To provide Nexus data trace messaging from bus masters on the master ports of the crossbar (XBAR) that do not inherently have a trace client, a Nexus Crossbar Multi-Master Client (NXMC) monitors traffic into the master port of the XBAR. Two NXMCs are implemented: one for the fast masters and one for the slow masters.
	  • NXMC clients are connected between E2E-ECC master client and AXBS. Therefore any corrupted data, correctable (single bit error) or un-correctable data (multi bit error), is traced without any trace error information. ECC errors are captured in MEMU.
	• See NXMC block connections

Chapter 75, Functional Safety	
Page 3218	• In Power section, changed "see Power Management Controller digital interface (PMC_dig)" to "see Power Management Controller (PMC)". • See Power
Chapter 78, Fault Collection and Control Unit (FCCU)	
Page 3276	• In Available FCCU output signals section added the note "The FCCU EOUT signals can toggle after LBIST execution during reset.". • See Available FCCU output signals.

Page 3278	• In FCCU Fault Inputs table updated the description of the following NCF Index:
	  • Changed the word Critical to Unrecoverable in Description of NCF Index 6
	  • Changed the word Non Critical to Recoverable in Description of NCF Index 7
	  • Changed the Description from "Critcal Fault in case LBIST or MBIST control signals go to wrong condition during User application" to "The fault in case the self-test was wrongly triggered without STCU programming" for NCF Index 8
	• See FCCU failure inputs from other modules.

Chapter 79, Self-Test Control Unit (STCU2)	
Page 3351	• In "STCU2 chip-specific module information" section added the note "Before running either online or offline self-test, user needs to mask the core LVD, but should enable the core LVDs before running the application via SW through the PMC_REE register". • See STCU2 chip-specific module information.
Page 3405	• In STCU2 Off-Line MBIST Status Medium register, changed the note from "This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished" to "This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished". • See STCU2 Off-Line MBIST Status Medium Register (STCU2_MBSM).
Chapter 80, Register Protection (REG_PROT)	
Page 3455	• Changed the name of section from "PMC_DIG protected registers" to "PMC protected registers" and changed the name of the table from "Protected PMC_dig registers" to "Protected PMC registers". • See PMC protected registers.
Chapter 81 Security	
Page 3481	• In "BAR execution protection" section deleted the text "But by default this error is masked.....the register is locked". • See BAR execution protection.
Chapter 82 Password and Device Security Module (PASS)	

Table continues on the next page...

Table 1-1. MPC5746RRM Rev6 Addendum (continued)

Location	Description
Page 3489	• In "Password and Device Security Module (PASS) chip-specific module information", added the note " For availability of RL0, RL1, and RL2 (Flash Region Lock bit fields) definitions in Password Group n Lock n Status Registers for the device, please refer to the Flash_Block_Assignment sheet in the pdf attachments". • See Password and Device Security Module (PASS) chip-specific module information
Page 3491	• Deleted the section "External signal description".
Page 3494	• In PASS_LCSTAT[JUN] bit, deleted the text "(In life cycle FA, the JUN bit will always be 0.)". • See Life Cycle Status Register (PASS_LCSTAT)
Page 3506	• In "Debug Disable (DD)" section, updated the text from "Debug Disable is an optional feature that, once the life cycle is In Field or beyond" to "Debug Disable is an optional feature that, once the life cycle is In Field". • See Debug Disable (DD)

1.2 Introduction

1.2.1 Block diagrams

The figures below are the top-level block diagrams. They show the general interconnection of functional modules through the crossbar switch and provide an indication of the modules that connect to external pins.

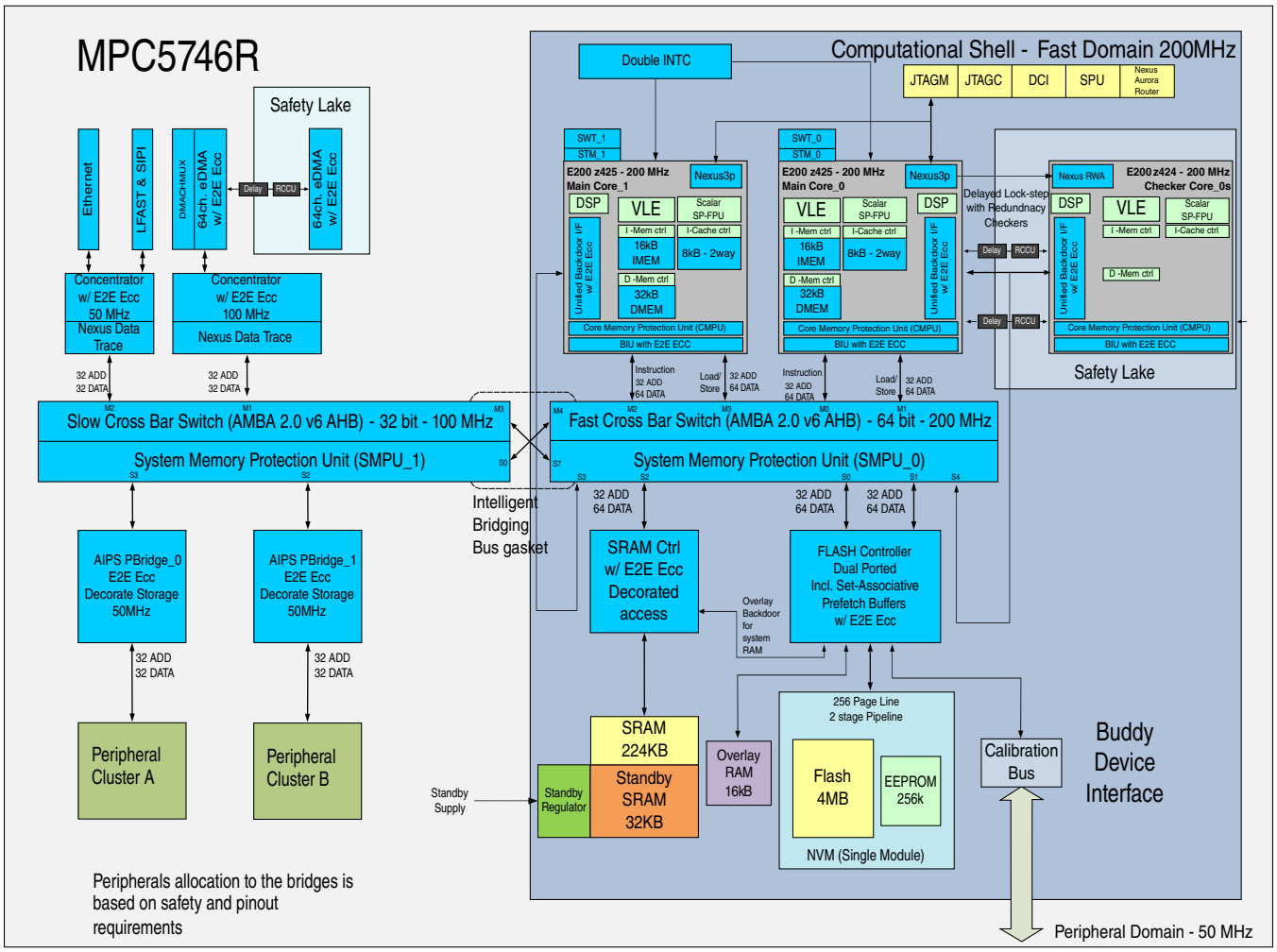


Figure 1-1. Core block diagram

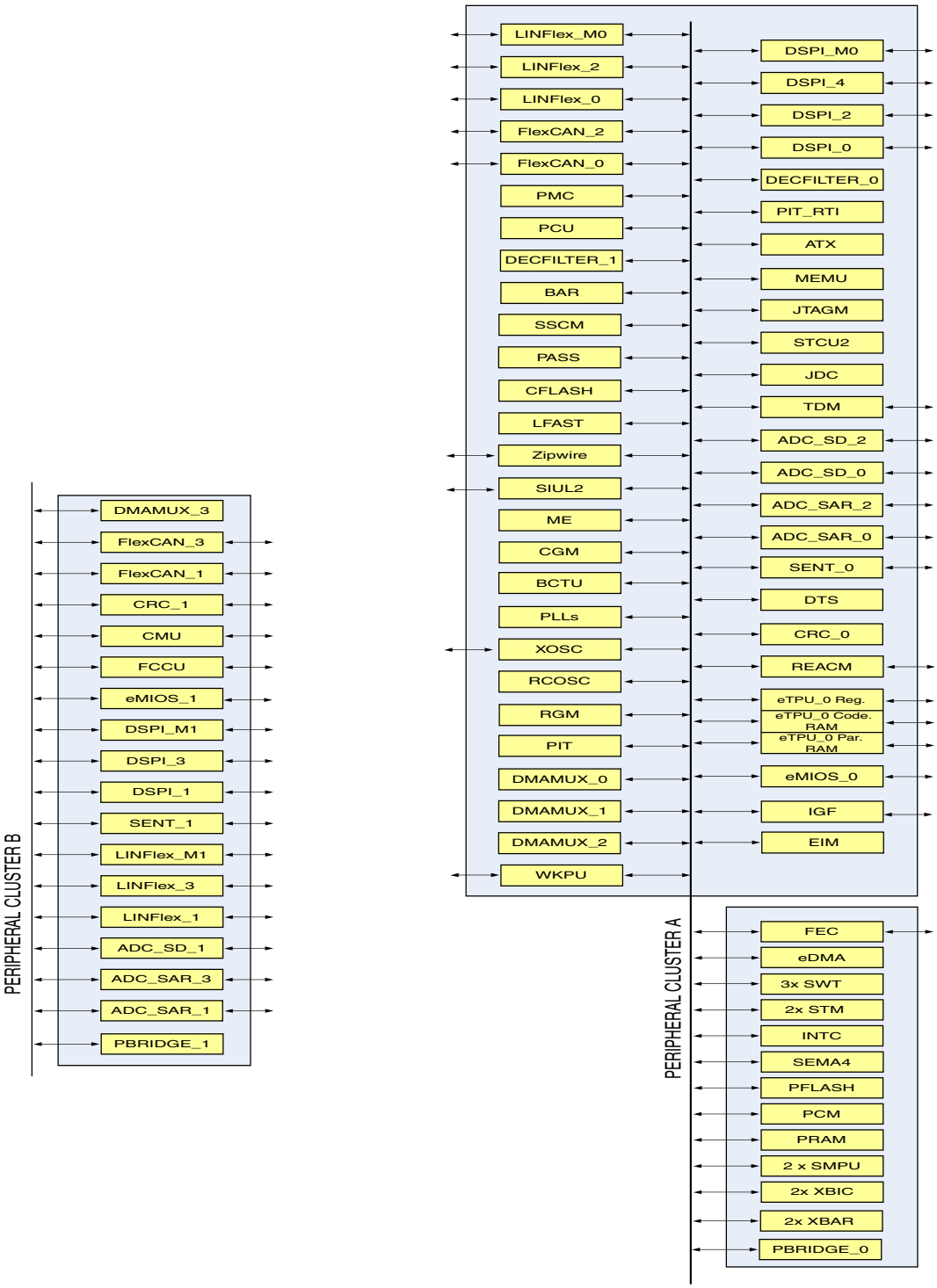


Figure 1-2. Peripherals allocation

1.2.2 Feature list

The following table lists a summary of major features of the device.

Table 1-2. Features list

Feature	MPC5746R	MPC5745R	MPC5743R
Main core	2	2	1[1]
Locked step core	1	1	1
Flash	4256KB + 16KB	3232KB + 16KB	2208KB + 16KB
EEPROM	2x64KB + 8x16KB	2x64KB + 8x16KB	2x64KB + 8x16KB
System RAM	256KB (including standby RAM)	192KB (including standby RAM)	128KB (including standby RAM)
I-CACHE	8KB/core	8KB/core	8KB/core
Local IRAM	16KB/core	16KB/core	16KB/core
Local DRAM	32KB/core	32KB/core	32KB/core
Stby RAM	32KB	32KB	32KB
Overlay RAM	16KB	16KB	16KB
Core freq	200/150MHz[2]	200/150MHz [2]	200MHz
SAR	4	4	4
SD ADC	3	3	3
Decimation Filters	2	2	2
eTPU	32ch x 2	32ch x 2	32ch x 2
Reaction module	10ch	10ch	10ch
DMA controller	64ch	64ch	64ch
FlexCAN	4	4	4
LINFlex	4+2MSB support	4+2MSB support	3+2MSB support[3]
DSPI	5+2MSB support[4]	5+2MSB support[4]	4+2MSB support[5]
Ethernet	10/100MBPS	10/100MBPS	10/100MBPS
SENT	6ch	6ch	6ch
Package	LQFP176 or MAPBGA252 MAPBGA292 Emulation Device	LQFP176 or MAPBGA252	LQFP176 or LQFP144

1. Core0 + Locked step core will be supported. Core 1 will be used for BAF execution at start-up only.
2. 256 BGA package supports 200MHz operation.
3. LINFlex 3 is not supported.
4. For 176 LQFP DSPI 4 is not supported.
5. DSPI 4 is not supported. For 144 LQFP, DSPI M1 is also not supported.

1.3 Standby SRAM Regulator

A total of 32 KB of the system SRAM on the device can remain powered when the rest of the device is powered down. This is achieved with the dedicated VDDSTBY pin in the packages. When the standby RAM is powered by VDDSTBY, the RAM is not operational, but the contents are retained.

The Standby RAM regulator has the following key features:

- Automatic, safe switching between operational (VDD_LV) and standby (VDDSTBY) supplies
- Internal regulation of the high voltage VDDSTBY supply to the low voltage standby RAM supply
- Brownout detection (minimum RAM data retention voltage violation) and reporting via a S/W clearable register bit

The Standby RAM regulator block diagram is given in Figure 1-3.

There are two primary modes of operation with the standby RAM feature:

- Standby RAM feature is not required in the application - in this case, the VDDSTBY pin is connected to ground, and the standby RAM on the device is supplied by VDD_LV and will behave as normal RAM for all conditions.
- Standby RAM feature is required in the application - in this case, the VDDSTBY pin is connected to a voltage within the range given in the device data sheet. The VDDSTBY pin is assumed to be connected to a constant supply, and present before and after the power up/down of the device LV and HV supplies. When the VDD_LV supply to the RAM is above the LV operational voltage threshold, the standby RAM is fully operational and behaves exactly the same as the non-standby system RAM. When the VDD_LV supply falls below the LV Detect threshold, the RAM low voltage supply is switched to the standby supply. See the device data sheet for min/max VDDSTBY values, LV supply thresholds, and the operational and brownout detection voltages.

The brownout detection feature of the standby RAM regulator informs the user if any voltage dip caused a RAM retention failure in standby mode. The brownout detector monitors the supply to the RAM array, and sets a latch in the PMC digital interface, when the voltage falls below the minimum RAM retention voltage given in the device data sheet. The latch is set on power up of the device. S/W then clears the latch by following the instructions in PMC_ADC_CS[STND_BY_RAM_BR_DETCT] description in the Power Management Controller (PMC) chapter. If the minimum RAM retention voltage is violated, the latch will be set again. The latch is set on every power-up of VDD_LV unless VDDSTBY is present and within the specified ranges in the data sheet.

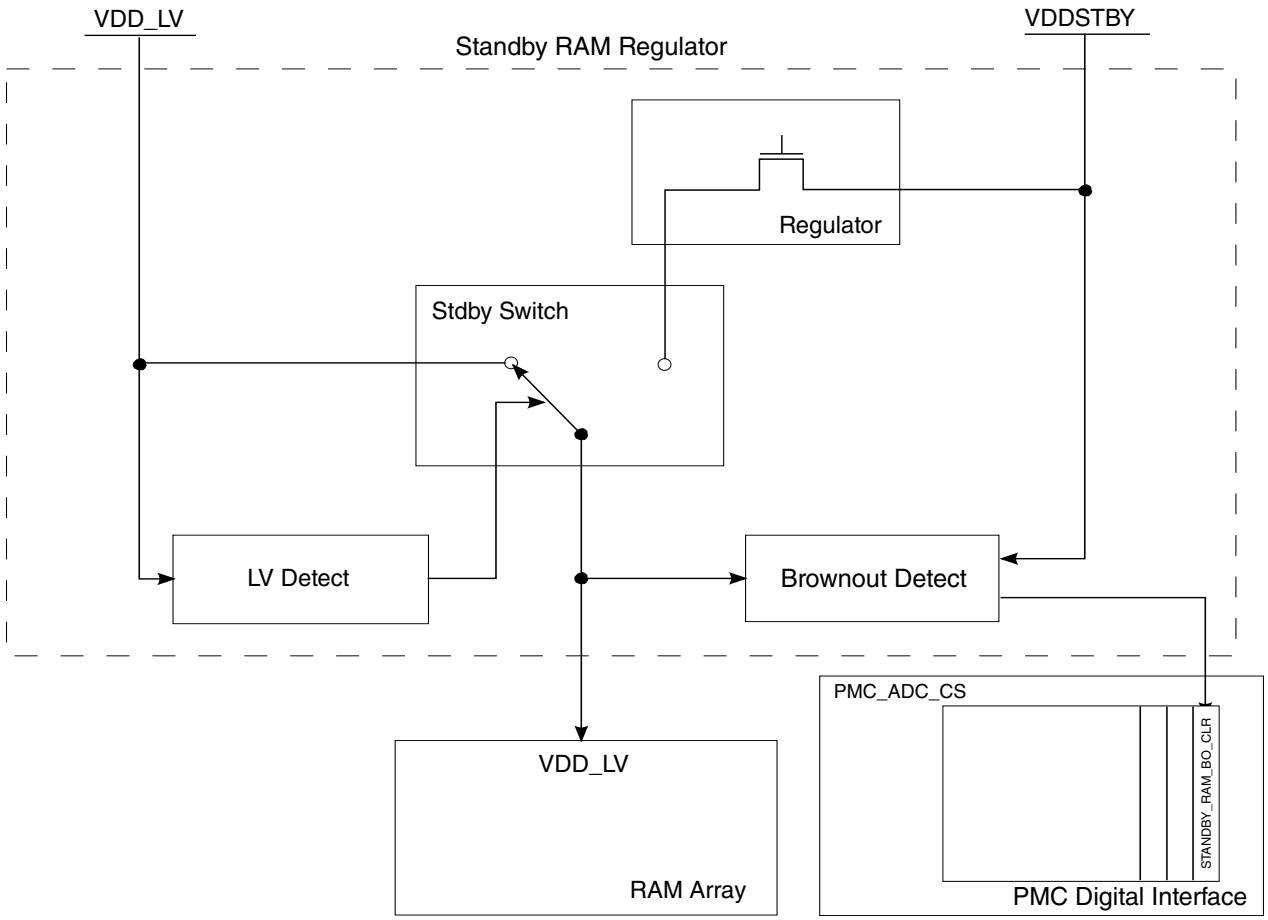


Figure 1-3. Standby RAM regulator block diagram

1.4 INTC Master Protection Register (INTC_MPROT)

The Master Protection Register is used to protect cores from writing to other cores INTC registers.

Address: 0h base + 4h offset = 4h

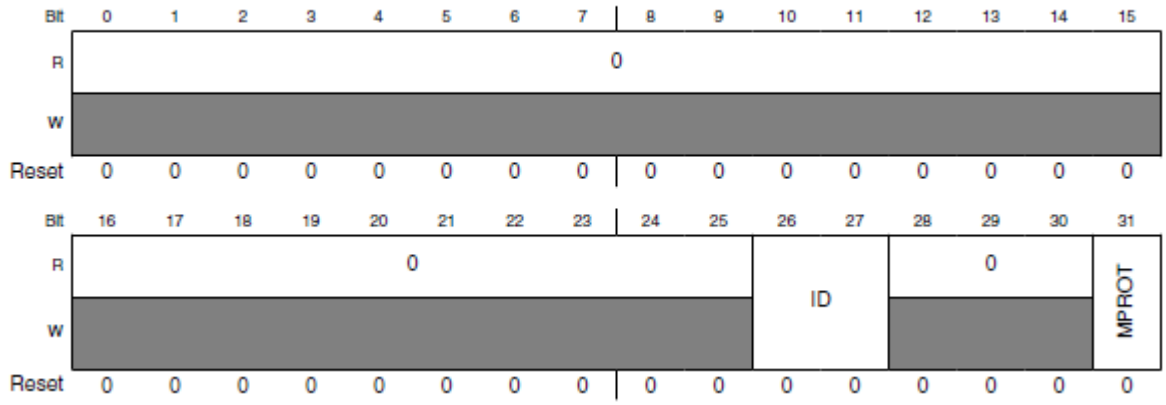


Table 1-3. INTC_MPROT field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26-27 ID	ID. This register can only be updated if MPROT=0 or if processor master ID=ID. When MPROT is set, the ID should also be updated to indicate which master can write to any register in the INTC memory map. Only the field values related to core bus masters can be selected. Non-core bus masters cannot be selected and would result in a bus access error. 00 master0 (reset default) 01 master1 10 master2 11 master3
28-30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 MPROT	Master Protection Controls what programming space a given master can write to. If MPROT is disabled, then all masters can write to any valid entry of the INTC memory map. If MPROT is enabled, INTC_MPROT[ID] can write to any valid entry of the INTC memory map but other masters can only write to their assigned resources. Once MPROT is enabled for a master, only that master may modify INTC_MPROT. 0 Master programming space protection disabled 1 Master programming space protection enabled

1.5 Clocking

1.5.1 Clock generation

The top-level clock generation architecture is given in the following figure. All clock selectors and dividers in the figure are located and programmed in the Clock Generation Module (MC_CGM). All dividers are integer dividers (1, 2, 3, …, n) with the ranges shown in Clock generation .

NOTE

CMU_0…CMU_9, CMU_12 in Clock generation use IRCOSC for their reference clock. However, the user can select XOSC as the reference in CMU_0 for monitoring IRCOSC.

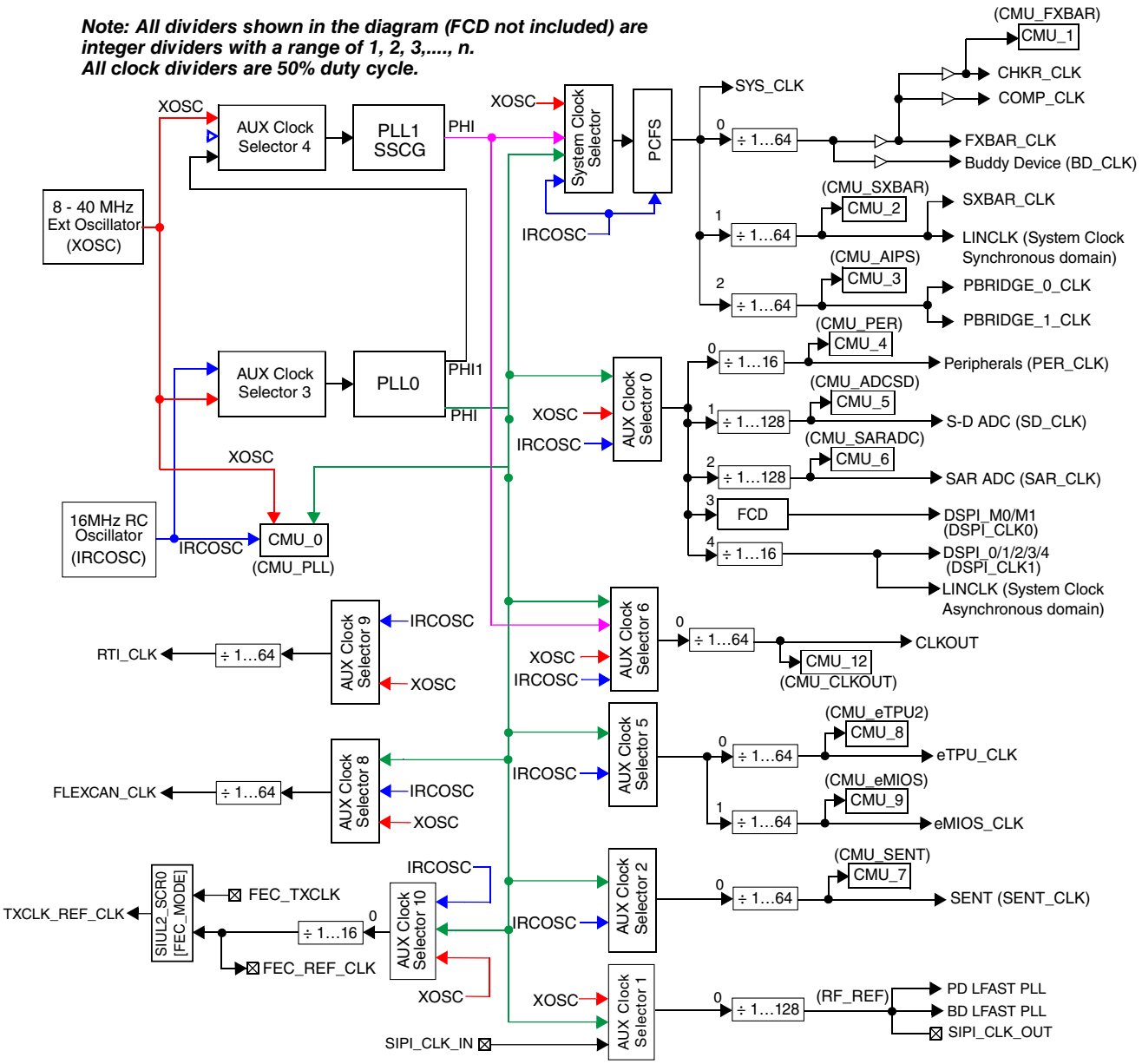


Figure 1-4. Clock generation

NOTE

Default clock source for all Auxillary Clock Selectors is the IRCOSC. User must configure MC_CGM_AC*n*_SC[SELCTL] before enabling the corresponding module functionality.

IRCOSC is the default clock for PLL1, but a PLL lock cannot be achieved using IRCOSC as the reference for PLL1.

NOTE

PBRIDGE_x_CLK may be referred to as "AIPS_CLK" elsewhere in this document.

NOTE

Do not bring out the ADC clock (XOSC + PLL0) or its component (XOSC and PLL0) on the CLKOUT pad when the ADC is in use using the same clock.

1.5.2 Peripheral Clocks

The following figure shows the clock distribution to the core and peripheral modules.

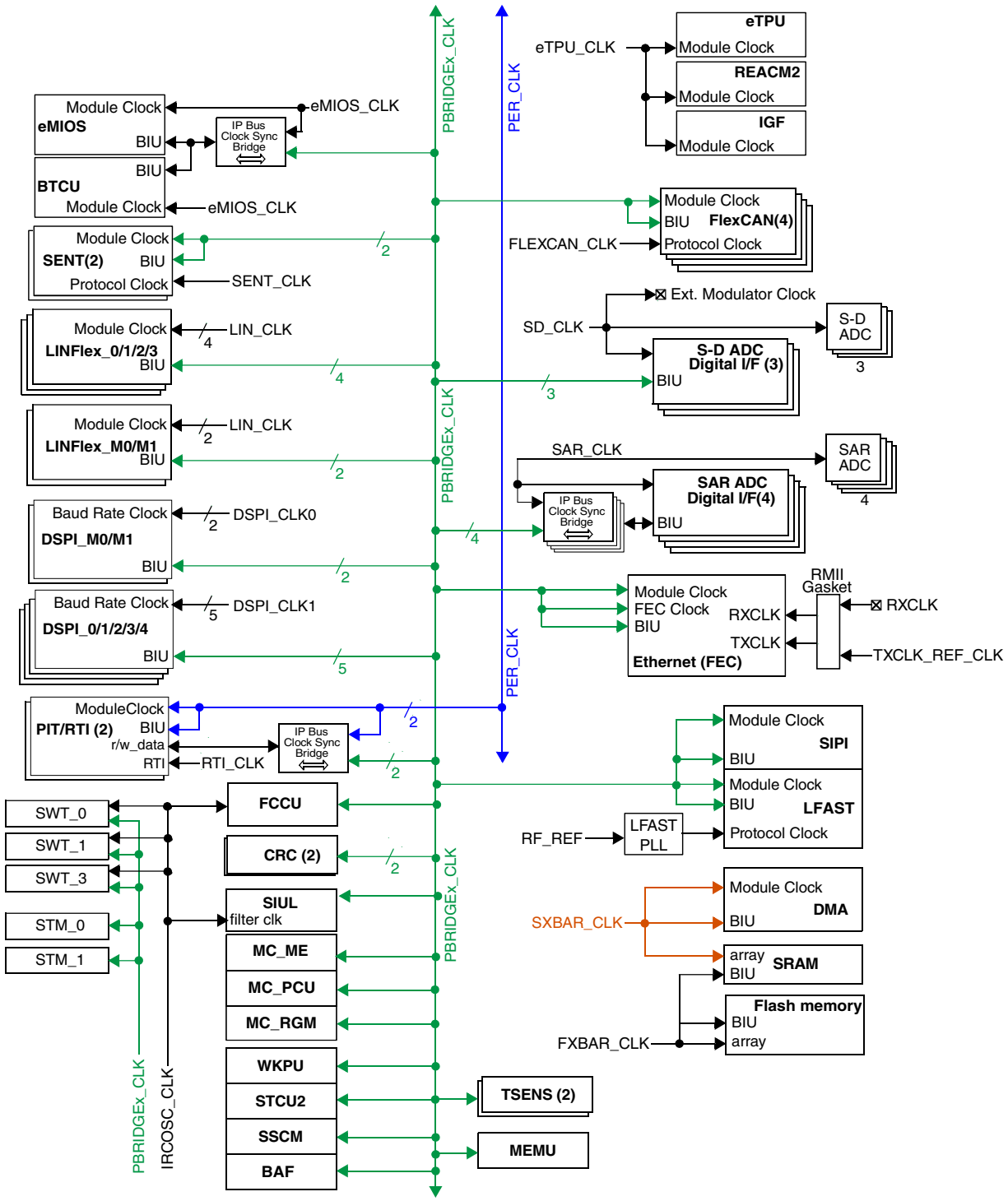


Figure 1-5. Clock distribution

1.6 CMU Control Status Register (CMU_CSR)

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								SFM	0						
W																
Reset	0	0	0	0	0	0	0	0	0*	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						CKSEL1		0					RCDIV		CME
W																
Reset	0	0	0	0	0	0	0*	0*	0	0	0	0	0	0*	0*	0

* Notes:
- RCDIV field: Not all CMU blocks will utilize this feature. See the “Clocking” chapter for CMU implementation details.
- CKSEL1 field: Not all CMU blocks will utilize this feature. See the “Clocking” chapter for CMU implementation details.
- SFM field: Not all CMU blocks will utilize this feature. See the “Clocking” chapter for CMU implementation details.

Table 1-4. CMU_CSR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 SFM	Start frequency measure. The software can only set this bit to start a clock frequency measure. It is reset by hardware when the measure is ready in the CMU_FDR. **Note**: MSR[MD] must be written before enabling frequency measurement (CSR[SFM] = 1). Do not write MDR[MD] while CSR[SFM] = 1. 0 Frequency measurement is completed or not yet started 1 Frequency measurement is not completed
9–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 CKSEL1	Frequency measure clock selection bit. CKSEL1 selects the clock to be measured by the frequency meter. This only affects CMU instances that utilizes clock metering. Not all CMU blocks will utilize this feature. See the “Clocking” chapter for device specific CMU implementation details. 00 CLKMT0_RMN is selected 01 Reserved 10 Reserved 11 CLKMT0_RMN is selected
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 RCDIV	These bits specify the CLKMT0_RMN division factor. The output clock frequency is $f_{CLKMT0_RMN} \div 2^{CMU_CSR[RCDIV]}$. This output clock is used as reference clock to compare with CLKMN0_RMT for crystal clock monitor feature. **NOTE**: Ensure clock source for CLKMN0_RMT is off before writing CMU_CSR[RCDIV]. 00 CLKMT0_RMN ÷ 1 (No division) 01 CLKMT0_RMN ÷ 2 10 CLKMT0_RMN ÷ 4

Table continues on the next page...

Table 1-4. CMU_CSR field descriptions (continued)

Field	Description
	11 CLKMT0_RMN ÷ 8
31 CME	CLKMN1 monitor enable. 0 CLKMN1 monitor is disabled 1 CLKMN1 monitor is enabled

1.7 Aux Clock 0 Divider Configuration 4 (MC_CGM_AC0_DC4)

This register controls auxilliary clock 0 divider 4.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 118h offset = 118h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0											DIV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1-5. MC_CGM_AC0_DC4 field descriptions

Field	Description
0 DE	Divider reserved Enable 0 Disable auxiliary clock 0 divider reserved 1 Enable auxiliary clock 0 divider reserved
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DIV	Divider reserved Division Value - For DSPI not supporting MSC. Divider range between 1 to 16. The divided DSPI 1 clock output will be generated such that it will have a period 'DIV + 1' times that of the auxiliary clock 0. If DE is set to '0' (divider reserved is disabled), any write access to the DIV field is ignored
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

1.8 Software Watchdog Timer (SWT)

1.8.1 Features

The SWT has the following features:

- 32-bit time-out register to set the time-out period
- Programmable selection of window mode or regular servicing
- Programmable selection of reset request or interrupt on an initial time-out
- Programmable selection of the servicing mode
- Master access protection
- Hard and soft configuration lock bits
- Reset configuration inputs allow timer to be enabled out of reset

1.8.2 SWT Control Register (SWT_CR)

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MAP0	MAP1	MAP2	MAP3	MAP4	MAP5	MAP6	MAP7	0							
W																
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SMD		RIA	WND	ITR	HLK	SLK	Reserved	STP	FRZ	WEN
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	1	1	0	1	1

Table 1-6. SWT_CR field description

Field	Description
0	Master Access Protection for Master 0.
MAP0	The number of this field corresponds to the logical ID of the bus master. The platform bus master assignments are device specific. For the masters with the logical IDs to which the MAPn fields correspond, see the chip-specific information for the Crossbar Switch.

Table continues on the next page...

Table 1-6. SWT_CR field description (continued)

Field	Description
	0 Access for the master is not enabled 1 Access for the master is enabled
1 MAP1	Master Access Protection for Master 1. The platform bus master assignments are device specific. For the masters with the logical IDs to which the MAPn fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
2 MAP2	Master Access Protection for Master 2. The number of this field corresponds to the logical ID of the bus master. The platform bus master assignments are device specific. For the masters with the logical IDs to which the MAPn fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
3 MAP3	Master Access Protection for Master 3. The number of this field corresponds to the logical ID of the bus master. The platform bus master assignments are device specific. For the masters with the logical IDs to which the MAPn fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
4 MAP4	Master Access Protection for Master 4. The number of this field corresponds to the logical ID of the bus master. The platform bus master assignments are device specific. For the masters with the logical IDs to which the MAPn fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
5 MAP5	Master Access Protection for Master 5. The number of this field corresponds to the logical ID of the bus master. The platform bus master assignments are device specific. For the masters with the logical IDs to which the MAPn fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
6 MAP6	Master Access Protection for Master 6. The number of this field corresponds to the logical ID of the bus master. The platform bus master assignments are device specific. For the masters with the logical IDs to which the MAPn fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
7 MAP7	Master Access Protection for Master 7. The number of this field corresponds to the logical ID of the bus master. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled

Table continues on the next page...

Table 1-6. SWT_CR field description (continued)

Field	Description
	1 Access for the master is enabled
8–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–22 SMD	Service Mode 00 Fixed Service Sequence: The watchdog is serviced by writing the fixed sequence 0xA602, 0xB480 to SWT_SR. 01 Keyed Service Sequence: The watchdog is serviced by writing two pseudorandom key values to SWT_SR. 10 Reserved—do not use. Writing a value can cause the SWT not to be serviced. 11 Reserved—do not use. Writing this value can cause the SWT not to be serviced.
23 RIA	Reset on Invalid Access **NOTE**: For a description of how this chip implements SWT reset requests resulting from SWT invalid accesses, see the chip-specific SWT information. 0 Invalid access to the SWT generates a bus error 1 Invalid access to the SWT causes a reset request if WEN is 1
24 WND	Window Mode 0 Regular mode: service sequence can be done at any time 1 Windowed mode: service sequence is valid only when the down counter is less than the value in the SWT_WN register
25 ITR	Interrupt Then Reset Request **NOTE**: For a description of how this chip implements SWT reset requests and interrupt requests resulting from SWT time-outs, see the chip-specific SWT information. 0 Generate a reset request on a time-out 1 Generate an interrupt on an initial time-out; generate a reset request on a second consecutive timeout
26 HLK	Hard Lock This bit is cleared only at reset. 0 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read/write registers if SLK is 0 1 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read-only registers
27 SLK	Soft Lock This bit is cleared by writing the unlock sequence to the service register. 0 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read/write registers if HLK is 0 1 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read-only registers
28 Reserved	Reserved This field is reserved.
29 STP	Stop Mode Control Allows the watchdog timer to be stopped when the device enters stop mode. 0 SWT counter continues to run in stop mode 1 SWT counter is stopped in stop mode
30	Debug Mode Control

Table continues on the next page...

Table 1-6. SWT_CR field description (continued)

Field	Description
FRZ	Allows the watchdog timer to be stopped when the device enters debug mode. 0 SWT counter continues to run in debug mode 1 SWT counter is stopped in debug mode
31 WEN	Watchdog Enabled 0 SWT is disabled 1 SWT is enabled

1.8.3 Time-out

The SWT_CR[ITR] bit controls the SWT's action taken when a time-out occurs.

- If the SWT_CR[ITR] bit is 0, the SWT generates a reset request immediately on any time-out.
- If the SWT_CR[ITR] bit is 1, an initial time-out causes the SWT to generate an interrupt and load the down counter with the time-out period.
  - The interrupt is indicated by the Time-out Interrupt Flag (SWT_IR[TIF]).
  - Clear the interrupt by writing 1 to SWT_IR[TIF].

  If the service sequence is not written before the second consecutive time-out, the SWT generates a reset request.

1.9 Enhanced Modular IO Subsystem (eMIOS200)

1.9.1 Modes of operation

The Unified Channels can be configured to operate in the following modes:

- General-Purpose Input/Output (GPIO)
- Single-Action Input Capture (SAIC)
- Single-Action Output Compare (SAOC)
- Input Pulse-Width Measurement (IPWM)

- Input Period Measurement (IPM)
- Double-Action Output Compare (DAOC)
- Pulse/Edge Accumulation (PEA)
- Pulse/Edge Counting (PEC)
- Windowed Programmable Time Accumulation (WPTA)
- Modulus Counter (MC)
- Modulus Counter Buffered (MCB)
- Output Pulse-Width and Frequency Modulation (OPWFM)
- Output Pulse-Width and Frequency Modulation Buffered (OPWFMB)
- Center Aligned Output Pulse Width Modulation with Dead Time Insertion (OPWMC)
- Center Aligned Output Pulse Width Modulation with Dead Time Insertion Buffered (OPWMCB)
- Output Pulse-Width Modulation (OPWM)
- Output Pulse-Width Modulation Buffered (OPWMB)
- Output Pulse-Width Modulation with Trigger (OPWMT)

These modes are described in Unified Channel modes of operation.

Each channel can have a specific set of modes.

1.9.2 eMIOS UC B register n (eMIOS_Bn)

Depending on the mode of operation, internal registers B1[0:23] or B2[0:23] can be assigned to address EMIOS_Bn. Both B1 and B2 are cleared by reset. The table below summarizes the EMIOS_Bn writing and reading accesses for all operation modes. Values not listed are reserved. For more information see the “UC Modes of Operation” section.

Depending on the channel configuration, it may have an EMIOS_B register or not. EMIOS_B register is required for the following modes: DAOC, IPM, IPWM, OPWM, OPWMB, OPWFM, OPWMT, OPWFMB, OPWMC, OPWMCB, MC, MCB, PEA, PEC, WPTA.

The following table summarizes the EMIOS_An, EMIOS_Bn, and EMIOS_ALTAn writing and reading accesses for all operation modes. Values not listed are reserved. For more information, see the “UC Modes of Operation” section.

Table 1-7. EMIOS_An, EMIOS_Bn and EMIOS_ALTAn values assignment

Operation mode	Register access					
	Write	Read	Write	Read	Alt write	Alt read
GPIO	A1, A2	A1	B1, B2	B1	A2	A2
SAIC[1]	–	A2	B2	B2	–	–
SAOC[1]	A2	A1	B2	B2	–	–
IPWM	–	A2	–	B1	–	–
IPM	–	A2	–	B1	–	–
DAOC	A2	A1	B2	B1	–	–
PEA	A1	A2	–	B1	–	–
PEC[1]	A1	A1	B1	B1	–	A2
WPTA	A1	A1	B1	B1	–	A2
MC[1]	A2	A1	B2	B2	–	–
OPWFM	A2	A1	B2	B1	–	–
OPWMC	A2	A1	B2	B1	–	–
OPWM	A2	A1	B2	B1	–	–
OPWMT	A2	A1	B2	B1	A2	A2
MCB[1]	A2	A1	B2	B1	–	–
OPWFMB	A2	A1	B2	B1	–	–
OPWMCB	A2	A1	B2	B1	–	–
OPWMB	A2	A1	B2	B1	–	–

1. In these modes, the register EMIOS_Bn is not used, but B2 can be accessed.

1.9.3 eMIOS CNT register n (eMIOS_CNTn)

When GPIO mode is selected or the channel is frozen, the EMIOS_CNTn register is read/write. For all others modes, the EMIOS_CNTn is a read-only register. When entering some operation modes, this register is automatically cleared (refer to the “UC Modes of Operation” section for details).

EMIOS_CNTn register is required for the following modes:

- OPWFM
- OPWFMB
- OPWMC
- OPWMCB

- PEA
- PEC
- WPTA
- MC
- MCB

For other modes, the internal counter is not used. Therefore, depending on the modes available in a channel, the internal counter may be absent.

1.9.4 eMIOS UC Control register n (eMIOS_Cn)

The UC Control register gathers several read/write control bits affecting the UC operation conditions.

NOTE

This register does not include a FORCMB bit for Channel Types A and C.

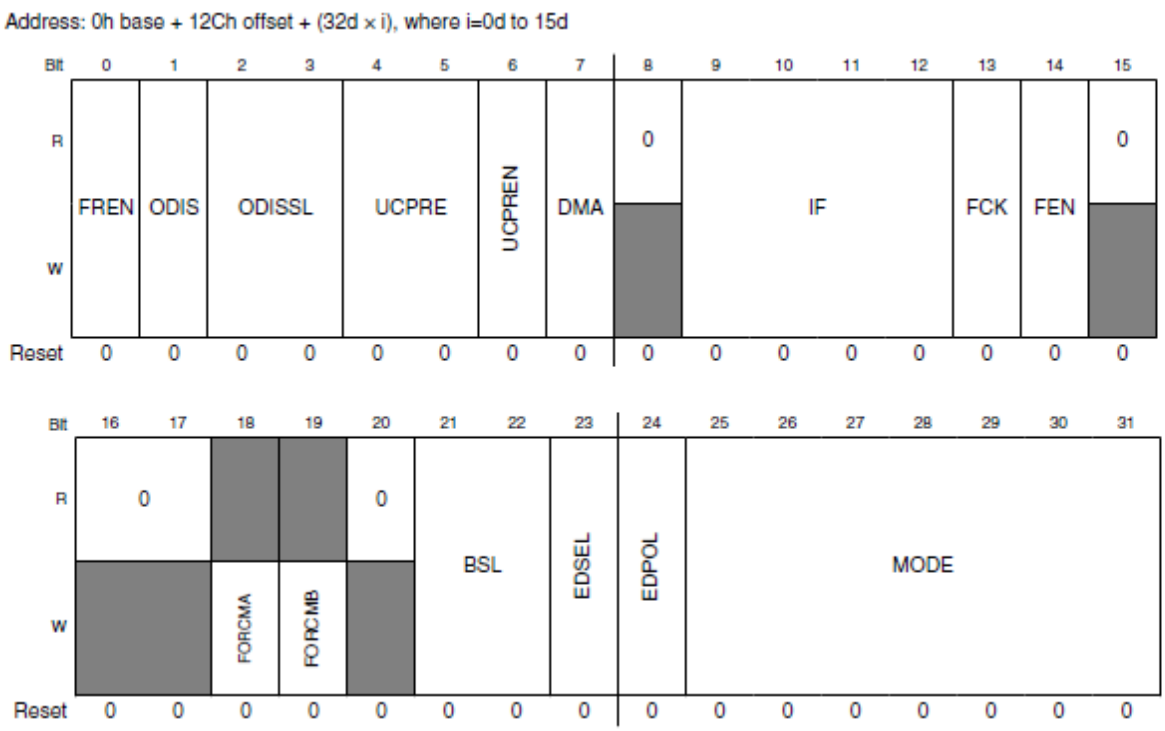


Table 1-8. eMIOS_Cn field descriptions

Field	Description
0 FREN	Freeze Enable bit.

Table continues on the next page...

Table 1-8. eMIOS_Cn field descriptions (continued)

Field	Description
	The FREN bit, if set and validated by FRZ bit in EMIOS_MCR register allows the channel to enter freeze state, freezing all registers values when in debug mode and allowing the MCU to perform debug functions. 0 Normal operation 1 Freeze UC registers values
1 ODIS	Output Disable bit. The ODIS bit allows disabling the output pin when running any of the output modes with the exception of GPIO mode. 0 The output pin operates normally 1 If the selected Output Disable Input signal is asserted, the output pin goes to EDPOL for OPWFMB and OPWMB modes and to the complement of EDPOL for other output modes, but the Unified Channel continues to operate normally, i.e., it continues to produce FLAG and matches. When the selected Output Disable Input signal is negated, the output pin operates normally.
2-3 ODISSL	Output Disable select bits. The ODISSL[0:1] bits select one of the four output disable input signals, as shown in the following table. **Table 1-9. UC ODISSL selection** ODISSL[0:1] / input signal 00 / Output Disable Input 0 01 / Output Disable Input 1 10 / Output Disable Input 2 11 / Output Disable Input 3 1. Filter latency (the difference in time between the input and the response) is 3 clock edges. 2. The input signal is synchronized before arriving to the digital filter. 3. Not all devices support all counter buses. See the Device Configuration chapter for details. 4. Not all modes of operation are supported. See the configuration information for details. 5. b = adjust parameters for the mode of operation. Refer to “UC Modes of Operation” section for details.
4-5 ACKE3	Prescaler bits. The UCPRE[0:1] bits select the clock divider value for the internal prescaler of Unified Channel, as shown in the following table. **Table 1-10. UC Internal Prescaler clock divider** UCPRE[0:1] / Divide ratio 00 / 1 01 / 2 10 / 3 11 / 4 1. Filter latency (the difference in time between the input and the response) is 3 clock edges. 2. The input signal is synchronized before arriving to the digital filter. 3. Not all devices support all counter buses. See the Device Configuration chapter for details. 4. Not all modes of operation are supported. See the configuration information for details. 5. b = adjust parameters for the mode of operation. Refer to “UC Modes of Operation” section for details.
6	Prescaler Enable bit.

Table continues on the next page...

Table 1-8. eMIOS_Cn field descriptions (continued)

Field	Description
UCPREN	The UCPREN bit enables the prescaler counter. 0 Prescaler disabled (no clock) 1 Prescaler enabled
7 DMA	Direct Memory Access bit. The DMA bit selects if FLAG generation will be used as an interrupt or as a DMA request. 0 Flag/overrun assigned to Interrupt request. 1 Flag/overrun assigned to DMA request.
8 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
9-12 IF	Input Filter bits. The IF[0:3] bits control the programmable input filter, selecting the minimum input pulse width that can pass through the filter, as shown in the following table. For output modes, these bits have no meaning. **Table 1-11. UC Input Filter bits** IF[0:3] / Minimum input Pulse width [filter clock periods] 0000 / bypassed 0001 / 02 0010 / 04 0100 / 08 1000 / 16 All others / Reserved 1. Filter latency (the difference in time between the input and the response) is 3 clock edges. 2. The input signal is synchronized before arriving to the digital filter. 3. Not all devices support all counter buses. See the Device Configuration chapter for details. 4. Not all modes of operation are supported. See the configuration information for details. 5. b = adjust parameters for the mode of operation. Refer to “UC Modes of Operation” section for details.
13 FCK	Filter Clock select bit. The FCK bit selects the clock source for the programmable input filter. 0 Prescaled clock 1 eMIOS module clock
14 FEN	FLAG Enable bit. The FEN bit allows the Unified Channel FLAG bit to generate an interrupt signal or a DMA request signal (the type of signal to be generated is defined by the DMA bit). 0 Disable (FLAG does not generate an interrupt request) 1 Enable (FLAG generates an interrupt request)
15–17 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
18	Force Match A bit.

Table continues on the next page...

Table 1-8. eMIOS_Cn field descriptions (continued)

Field	Description
FORCMA	For output modes, the FORCMA bit is equivalent to a successful comparison on comparator A (except that the FLAG bit is not set). This bit is cleared by reset and is always read as zero. This bit is valid for every output operation mode which uses comparator A, otherwise it has no effect. For input modes, the FORCMA bit is not used and writing to it has no effect. 0 Has no effect. 1 Force a match at comparator A
19 FORCMB	Force Match B bit. For output modes, the FORCMB bit is equivalent to a successful comparison on comparator B (except that the FLAG bit is not set). This bit is cleared by reset and is always read as zero. This bit is valid for every output operation mode which uses comparator B, otherwise it has no effect. 0 Has not effect 1 Force a match at comparator B
20 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
21–22 BSL	Bus Select bits. The BSL[0:1] bits are used to select either one of the counter buses or the internal counter to be used by the Unified Channel , as shown in the following tables.

Table 1-12. UC BSL bits

BSL[0:1]	Selected bus
00	All channels: counter bus[A]
01	Channels 0 to 7: counter bus[B]
	Channels 8 to 15: counter bus[C]
	Channels 16 to 23: counter bus[D]
	Channels 24 to 31: counter bus[E]
10	Reserved
11	All channels: internal counter

1. Filter latency (the difference in time between the input and the response) is 3 clock edges.
2. The input signal is synchronized before arriving to the digital filter.
3. Not all devices support all counter buses. See the Device Configuration chapter for details.
4. Not all modes of operation are supported. See the configuration information for details.
5. b = adjust parameters for the mode of operation. Refer to “UC Modes of Operation” section for details.

23 EDSEL	Edge Selection bit. For input modes, the EDSEL bit selects whether the internal counter is triggered by both edges of a pulse or just by a single edge as defined by the EDPOL bit . When not shown in the mode of operation description, this bit has no effect. 0 – Single edge triggering defined by the EDPOL bit 1 – Both edges triggering{mts_ipi_slv_saic} For GPIO in mode, the EDSEL bit selects if a FLAG can be generated. 0 – A FLAG is generated as defined by the EDPOL bit

Table continues on the next page...

Table 1-8. eMIOS_Cn field descriptions (continued)

Field	Description
	1 – No FLAG is generated For SAOC mode, the EDSEL bit selects the behavior of the output flip-flop at each match. 0 – The EDPOL value is transferred to the output flip-flop 1 – The output flip-flop is toggled
24 EDPOL	Edge Polarity bit. For input modes, the EDPOL bit asserts which edge triggers either the internal counter or an input capture or a FLAG . When not shown in the mode of operation description, this bit has no effect. 0 – Trigger on a falling edge 1 – Trigger on a rising edge For output modes, the EDPOL bit is used to select the logic level on the output pin . 0 – A match on comparator A clears the output flip-flop, while a match on comparator B sets it 1 – A match on comparator A sets the output flip-flop, while a match on comparator B clears it
25-31 MODE	Mode selection bits. The MODE bits select the mode of operation of the Unified Channel, as shown in the following table. Values not listed are reserved. **NOTE**: If a reserved value is written to the mode bit field, the results are unpredictable.

Table 1-13. UC MODE bits

MODE[0:6]	Mode of operation
0000000	General purpose Input/Output mode (input)
0000001	General purpose Input/Output mode (output)
0000010	Single Action Input Capture
0000011	Single Action Output Compare
0000100	Input Pulse Width Measurement
0000101	Input Period Measurement
0000110	Double Action Output compare (with FLAG set on B match)
0000111	Double Action Output compare (with FLAG set on both match)
0001000	Pulse/Edge Accumulation (continuous)
0001001	Pulse/Edge Accumulation (single shot)
0001010	Pulse/Edge Counting (continuous)
0001011	Pulse/Edge Counting (single shot)
0001110	Windowed Programmable Time Accumulation
0001111	Reserved
001000b[-1]	Modulus Counter (Up counter with clear on match start)
001001b	Modulus Counter (Up counter with clear on match end)
00101bb[-1]	Modulus Counter (Up/Down counter)
00110b0[-1]	Output Pulse Width and Frequency Modulation (immediate update)
00110b1[-1]	Output Pulse Width and Frequency Modulation (next period update)

Table 1-8. eMIOS_Cn field descriptions

Field	Description
	Table 1-13. UC MODE bits (continued)

MODE[0:6]	Mode of operation
00111b0[-1]	Center Aligned Output Pulse Width Modulation (with trail edge dead-time)
00111b1[-1]	Center Aligned Output Pulse Width Modulation (with lead edge dead-time)
01000b0[-1]	Output Pulse Width Modulation (immediate update)
01000b1[-1]	Output Pulse Width Modulation (next period update)
0100100 through 0100101	Reserved
0100110	Output Pulse Width Modulation with Trigger
0100111 through 1001111	Reserved
101000b[-1]	Modulus Counter Buffered (Up counter)
101001b[-1]	Reserved
10101bb[-1]	Modulus Counter Buffered (Up/Down counter)
10110b0[-1]	Output Pulse Width and Frequency Modulation Buffered
10110b1[-1]	Reserved
10111b0[-1]	Center Aligned Output Pulse Width Modulation Buffered (with trail edge dead-time)
10111b1[-1]	Center Aligned Output Pulse Width Modulation Buffered (with lead edge dead-time)
11000b0[-1]	Output Pulse Width Modulation Buffered
1100001 through 1111111	Reserved

1. Filter latency (the difference in time between the input and the response) is 3 clock edges.
2. The input signal is synchronized before arriving to the digital filter.
3. Not all devices support all counter buses. See the Device Configuration chapter for details.
4. Not all modes of operation are supported. See the configuration information for details.
5. b = adjust parameters for the mode of operation. Refer to “UC Modes of Operation” section for details.

1. Filter latency (the difference in time between the input and the response) is 3 clock edges.
2. The input signal is synchronized before arriving to the digital filter.
3. Not all devices support all counter buses. See the Device Configuration chapter for details.
4. Not all modes of operation are supported. See the configuration information for details.
5. b = adjust parameters for the mode of operation. Refer to “UC Modes of Operation” section for details.

1.10 FEC interface selection

On this chip, the FEC interface depends on the settings of the RCR (in the FEC) and SIUL2_SCR0 (in the SIUL2), as shown in the following table.

Table 1-14. FEC interface selection

RCR[MII_MODE]	SIUL2_SCR0[FEC_MODE]	Interface used by the FEC
0	0 or 1	7-wire
1	0	RMII
	1	MII-Lite[1]

1. MII-Lite supported only on BGA packages. Full MII is not supported on this chip.

NOTE

It is required to setup the system clocks recommended in System clock frequency limitations before FEC is operational.

1.11 SENT Receiver (SRX)

1.11.1 Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CHn_CONFIG)

This register (CHn_CONFIG) provides the configurability for interrupt assertion for various events and error as explained in Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CHn_STATUS) and control bits for selection of various receive options that might be required by the user software. This register also configures the width of noise glitches (on the sensor input) to be filtered out by the input programmable filter.

User must program the contents of this register before enabling the channel by writing to the corresponding channel enable bit in the Channel Enable Register (SRX_CHNL_EN).

Address: 0h base + 68h offset + (16d × i), where i=0d to 1d

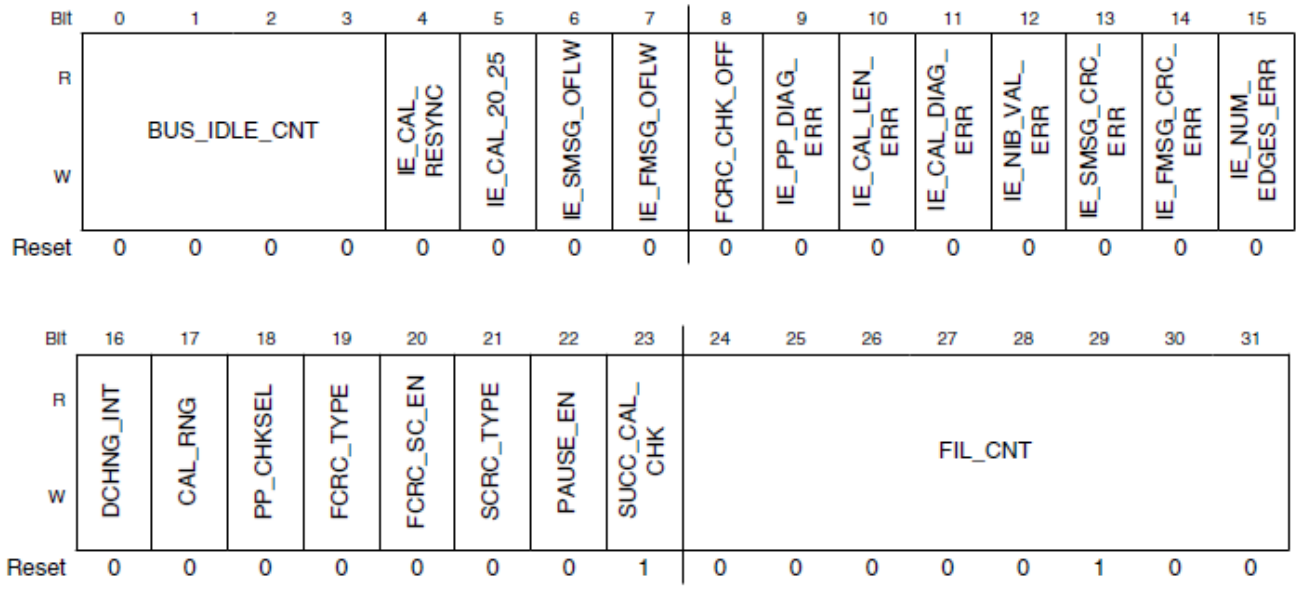


Table 1-15. SRX_CHn_CONFIG field descriptions

Field	Description
0–3 BUS_IDLE_CNT	Bus Idle Count. This register value defines the maximum allowable idle period on the sensor interface of this channel. The value is defined as follows: 0000 Disabled. 0001 127*2 Receiver Clock Tick Counts 0010 127*4 Receiver Clock Tick Counts 0100 127*8 Receiver Clock Tick Counts 1000 127*16 Receiver Clock Tick Counts
4 IE_CAL_ RESYNC	Successive Calibration Check Resynchronized Interrupt Enable: This bit enables interrupt assertion when Successive Calibration diagnosis has failed three times in case of "Option 2" being selected for Successive Calibration Check Method and the check resynchronizes the take the third message to be correct. 0 Interrupt is disabled 1 Interrupt is enabled
5 IE_CAL_20_25	Calibration Variation 20 - 25% Interrupt Enable. This bit enables interrupt assertion when corresponding status bit is set. 0 Interrupt is disabled 1 Interrupt is enabled
6 IE_SMSG_OFLW	Slow Serial Message Overflow Interrupt Enable. This bit enables interrupt assertion when overflow occurs on reception of Slow Serial Messages in corresponding channel 0 Interrupt is disabled 1 Interrupt is enabled

Table continues on the next page...

Table 1-15. SRX_CHn_CONFIG field descriptions (continued)

Field	Description
7 IE_FMSG_OFLW	Fast Message Overflow Interrupt Enable. This bit enables interrupt assertion when overflow occurs on reception of Fast Messages in corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
8 FCRC_CHK_ OFF	Fast Message CRC Check Off: This bit can be used to switch off CRC check in Fast Message 0 Check is enabled 1 Check is disabled/off
9 IE_PP_DIAG_ ERR	Ratio of calibration pulse length to message length varies by more than ±1.5625% between two frames. Valid for messages with pause pulse. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
10 IE_CAL_LEN_ ERR	Calibration pulse is wider than 56 ticks ±25%. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
11 IE_CAL_DIAG_ ERR	Successive Calibration pulses differ by more than ±1.56%. SAE spec defines it be 1.5625% but the accuracy of this check depends on the protocol clock. For instance for 62.5 MHz, this check will fail for 1.56% and passes for 1.55%. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
12 IE_NIB_VAL_ ERR	Any nibble data value <0 or >15. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
13 IE_SMSG_CRC_ ERR	Checksum error in Slow Serial Message. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
14 IE_FMSG_CRC_ ERR	Checksum error in Fast Message. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
15 IE_NUM_ EDGES_ERR	Checksum error in Fast Message. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
16 DCHNG_INT	Enable for Interrupt on Reception of Fast Message with Changed Data: This bit is used to enable storing and generating corresponding DMA or Message Ready Interrupt Request for Fast Message only when any of the data nibbles after different from previously received Fast Message on that

Table continues on the next page...

Table 1-15. SRX_CHn_CONFIG field descriptions (continued)

Field	Description
	channel. If a new Fast Message is received that has same values for all data nibbles as that of previous message then the current message will be discarded and will not be stored in the buffer, and neither Fast Message Ready Interrupt nor DMA request will be generated. 0 All Fast Messages will be received even if data nibbles are same 1 Only Fast Messages with differing values of data nibbles will be received
17 CAL_RNG	Valid Calibration Pulse Range Selection: This bit is used to control whether 20% variation or 25% variation in Calibration Pulse will be tolerated when detecting a Calibration Pulse 0 20% variation is acceptable 1 25% variation is acceptable
18 PP_CHKSEL	Pause Pulse Diagnostic Check Selection: This bit controls which diagnostic checks to run when messages are enabled to be received with pause pulses. 0 Both Successive Calibration Pulse Check and Pause Pulse Diagnostic are run 1 Only Pause Pulse Diagnostic is run. Successive Calibration Pulse Check is disabled.
19 FCRC_TYPE	Fast Message CRC Type. This indicates the type of CRC method to be used for CRC Diagnostic Check on the received fast messages 0 Recommended implementation (additional 0 data nibble XORed with the rest of the nibbles) 1 Legacy implementation (no additional 0 data nibble is XORed)
20 FCRC_SC_EN	Fast Message CRC Status and Communication Nibble Enable. This bit enables the inclusion of Status and Communication Nibble when CRC is calculated for Fast Messages. 0 Status and Communication Nibble not included in CRC calculation 1 Status and Communication Nibble is included in CRC calculation
21 SCRC_TYPE	Slow Serial Message CRC Type. This indicates the type of CRC method to be used for CRC Diagnostic Check on the received slow serial messages. 0 Recommended implementation (additional 0 data nibble XORed with the rest of the nibbles) 1 Legacy implementation (no additional 0 data nibble XORed)
22 PAUSE_EN	Pause Pulse Enable. Enables the channel receiver to detect a pause pulse. 0 Detection of Pause Pulse is disabled 1 Detection of Pause Pulse enabled
23 SUCC_CAL_ CHK	Successive Calibration Pulse Check Method. This bit indicates the method to be used for performing the successive calibration pulse check. Default value is 1. 0 Option 2 i.e. Low Latency Option as per SAE Specification 1 Option 1 i.e. Preferred but High Latency Option as per SAE Specification
24–31 FIL_CNT	Input Filter Sample Count. This indicates twice the number of Protocol Clock (of High Frequency Receiver Clock) cycles required for channel input from device's pad to remain stable before it is sampled as 0 or 1. This field defines the width of glitches to be filtered out by the input programmable filter on that channel. Only one bit should be set in this field. Output of the filter is equal to 2*FIL_CNT+4'. 0 No filtering. Input from device's pad is only synchronized 1 Non-Zero Filtering is enabled

1.11.2 Input programmable filter

The Input Programmable Filter ensures that only valid input pin transitions are received by the SENT Receiver module. The input programmable filter is an 8-bit programmable up counter that increments on the high frequency receiver clock. The sensor input signal from device's pad is synchronized by high frequency receiver clock. When a state change occurs in the sensor input signal, the 8-bit counter resets and starts counting up on the high frequency receiver clock. As long as the new state is stable on the pin, the counter remains incrementing. If a counter overflows occurs, the new pin value (i.e. sensor input) is validated and latched in the filtered output flop (also running on high frequency receiver clock). If the opposite edge appears on the sensor input before validation (i.e. overflow), the counter is reset and the input is not passed onto the filtered output. At the next pin transition, the counter starts counting again. Any pulse that is shorter than a full range of the masked counter is regarded as a glitch and it is not passed on to the filter output. The number of samples (or the match value of the up counter) is programmable by the user in the channel's configuration register. When zero is programmed into the registers, it bypasses the filter and the synchronized input is output as the filtered output. A non-zero value delays the filtered output by the number of clocks proportional to the programmed value in the Channel Configuration Register. A timing diagram of the input filter when FIL_CNT is 1, is shown in the following figure.

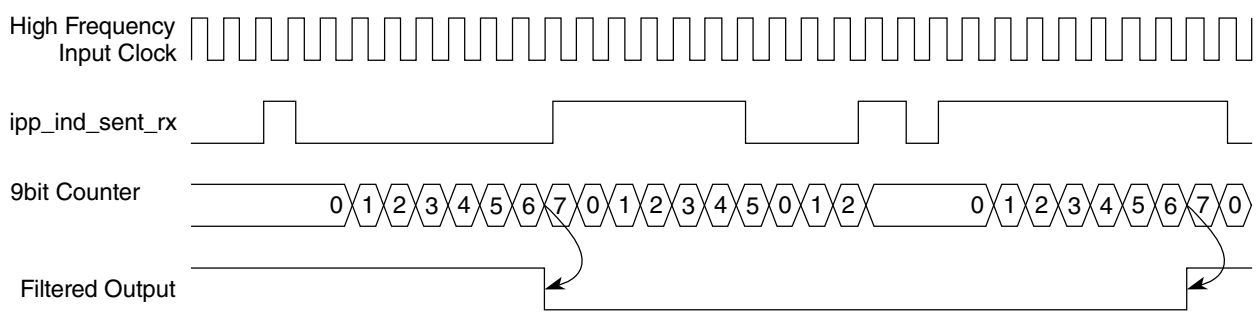


Figure 1-6. Input programmable filter timing diagram

1.11.3 Pause pulse diagnostic

The pause pulse diagnostic checks whether the ratio of calibration pulse length to full message length (including pause pulse) varies from one message to another by less than 1.5625% (as per spec). In order to have reduced latency of operation, message reception is not gated due to this check and no messages are discarded. However, when this check fails, a status bit is asserted to let the software know of this condition. The successive calibration pulse check can be configured to operate along with this check. This configurability is added to avoid conflicting results from the two checks.

There can be some inaccuracy in measuring pause pulse length as SAE specification specifies maximum clock drift/jitter up to maximum nibble length only and the frequency of high frequency receiver clock (protocol clock) used, allows clock compensation logic to ensure accuracy up to maximum nibble length. Hence there could be some inaccuracy in this check when sampling pause pulses that are larger than the maximum nibble pulse length.

This check will assert the PP_DIAG_ERR status bit on failure. This check does not run when reception of pause pulse is disabled.

When the Single Edge Nibble Transmission (SENT) Receiver (SRX) is configured to receive a pause pulse (Channel ‘n' Configuration Register - CHn_CONFIG[PAUSE_EN] = 1) and the length of the pause pulse from the sensor is in range of the valid calibration pulse length, the NUM_EDGES error can get asserted spuriously (Channel ‘n' Status Register - CHn_STATUS(NUM_EDGES_ERR] = 1) when there is any diagnostic error (other than number of expected edges error) or overflow in the incoming messages from the sensor.

Software can distinguish a spurious NUM_EDGES_ERR error from a real one by monitoring other error bits. The following tables will help distinguish between a false and real assertion of NUM_EDGES_ERR error and other errors. Software should handle the first error detected as per application needs and other bits can be evaluated based on these tables. Table 1 contains information due to erratum behavior and Table 2 below contains clarification of normal NUM_EDGES_ERR behavior.

Table 1-16. Behavior of NUM_EDGES_ERR

First Error Detected	Other error bits asserted	Cause for extra error bits getting asserted	Action
NIB_VAL_ERR	NUM_EDGES_ERR asserted twice	Upon detection of the first error, the state machine goes into a state where it waits for a calibration pulse, the first NUM_EDGES_ERR error is for the current message as the state machine does not detect an end of message. The second error comes when both the Pause pulse and the Calibration pulse are seen as back to back calibration pulses and no edges in between.	Ignore both NUM_EDGES_ERR error
FMSG_CRC_ERR	NUM_EDGES_ERR asserted twice	Upon detection of the first error, the state machine goes into a state where it waits for a calibration pulse, the first NUM_EDGES_ERR error is for the current message as	Ignore both NUM_EDGES_ERR errors

Table continues on the next page...

Table 1-16. Behavior of NUM_EDGES_ERR (continued)

		the state machine does not detect an end of message. The second error comes when both the Pause pulse and the Calibration pulse are seen as back to back calibration pulses and no edges in between.	
CAL_LEN_ERR	NUM_EDGES_ERR asserted once	Since the calibration pulse is not detected as a valid calibration pulse, the internal edges counter does not detect the end of one message and start of bad message (which has CAL_LEN_ERR); hence the NUM_EDGES_ERR gets asserted.	Ignore NUM_EDGES_ERR error

Resynchronization of SENT Receiver in case of any diagnostic error is shown in figure Figure 1-7.

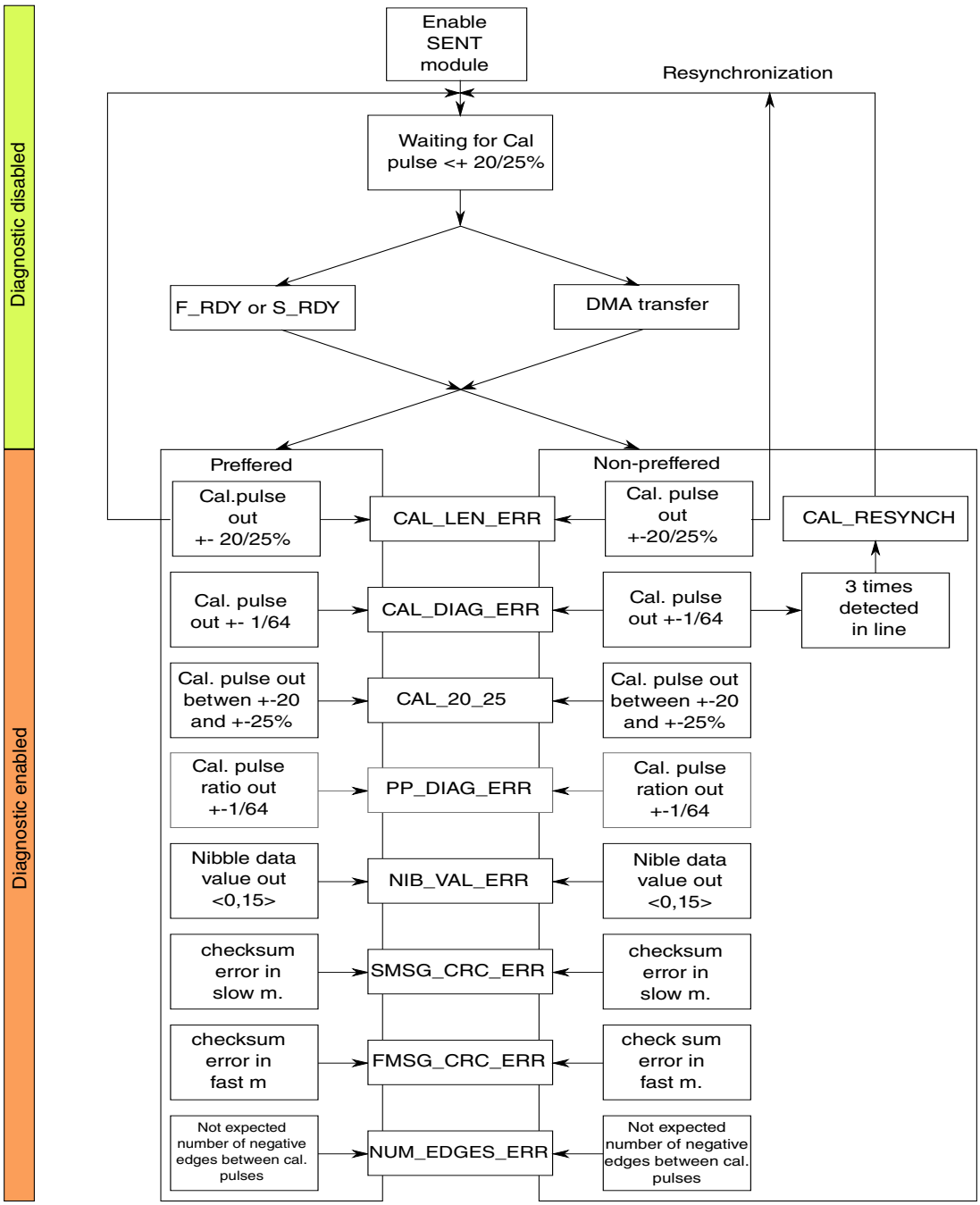


Figure 1-7. SENT receiver resynchronization

Table 1-17. Expected behavior, clarification of NUM_EDGES_ERR cases

First Error Detected	Other error bits asserted	Cause for extra error bits getting asserted	Action
NUM_EDGES_ERR (when edges are less than expected)	NIB_VAL_ERR is asserted	When the actual number of edges in the message are less than expected, then a pause pulse gets detected as a nibble since the state machine expects nibbles	Ignore the NIB_VAL_ERR

Table continues on the next page...

Table 1-17. Expected behavior, clarification of NUM_EDGES_ERR cases (continued)

		when actually there is a pause pulse present. This generates NIB_VAL_ERR.	
NUM_EDGES_ERR (when edges are more than expected)	NIB_VAL_ERR and PP_DIAG_ERR are asserted	When the actual number of edges in a message are more than expected, then after receiving the programmed number of data nibbles, the state machine expects a pause pulse. However, the pause pulse comes later and gets detected as a nibble and hence NIB_VAL_ERR is asserted. Since the message length is not correct, PP_DIAG_ERR is also asserted.	Ignore NIB_VAL_ERR and PP_DIAG_ERR

1.12 SIPI Error Register (SIPI_ERR)

This register contains the error bits for the last transaction(s) for all the channels. Communication errors are generated only for out of range address accesses, areas that are read only and where accesses do not lead to generation of transfer errors. Also, writing to read only registers will not generate errors.

Address: 0h base + B0h offset = B0h

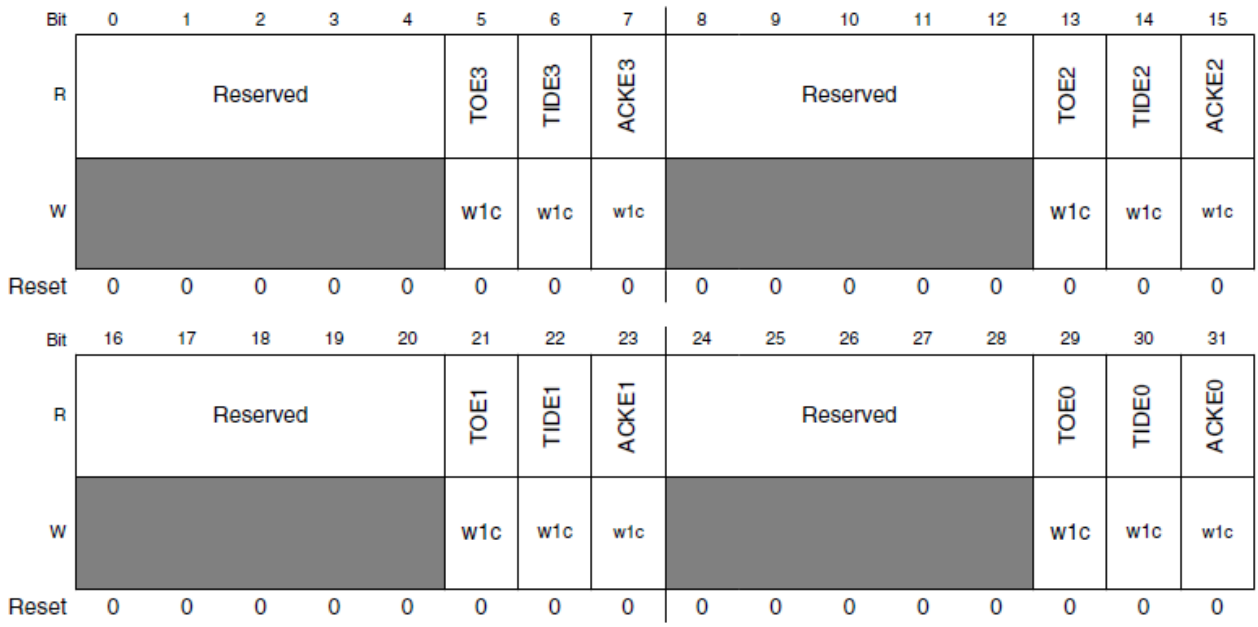


Table 1-18. SIPI_ERR field descriptions

Field	Description
0–4 Reserved	This field is reserved.
5 TOE3	Timeout Error for Channel 3. 0 Timeout error didn't occur 1 Timeout error occurred
6 TIDE3	Transaction ID Error for Channel 3. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
7 ACKE3	Acknowledge Error for Channel 3. 0 Acknowledge received is correct. 1 Acknowledge received is not correct.
8-12 Reserved	This field is reserved.
13 TOE2	Timeout Error for Channel 2. 0 Timeout error didn't occur 1 Timeout error occurred
14	Transaction ID Error for Channel 2.

Table continues on the next page...

Table 1-18. SIPI_ERR field descriptions (continued)

Field	Description
TIDE2	0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
15 ACKE2	Acknowledge Error for Channel 2. 0 Acknowledge received is correct. 1 Acknowledge received is not correct.
16-20 Reserved	This field is reserved.
21 TOE1	Timeout Error for Channel 1. 0 Timeout error didn't occur 1 Timeout error occurred
22 TIDE1	Transaction ID Error for Channel 1. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
23 ACKE1	Acknowledge Error for Channel 1. 0 Acknowledge received is correct. 1 Acknowledge received is not correct.
24-28 Reserved	This field is reserved.
29 TOE0	Timeout Error for Channel 0. 0 Timeout error didn't occur 1 Timeout error occurred
30 TIDE0	Transaction ID Error for Channel 0. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
31 ACKE0	Acknowledge Error for Channel 0. 0 Acknowledge received is correct. 1 Acknowledge received is not correct.

1.13 PLL configuration example

This is a typical configuration for 320 MHZ LFAST high-speed clock when the reference clock is 20 MHz.

1. Program the LFAST_PLLCR[FBDIV] for the Feedback Division factor, in this case 15 (divide by 16)
2. Program the LFAST_PLLCR[PREDIV] for the Division factor for PLL Reference Clock, in this case 0 (divide by 1)

3. Program the LFAST_PLLCR[SWPON] to 1, for switching the PLL ON

$$\text{PLL Clock} = \left(\frac{\text{clk_in}}{\text{d3g_pll_pre_div}}\right) \times (\text{d3g_pll_feedback_div} + 0.5) \quad \text{where d3g_pll_fraction_div_en=1}$$

Equation 1. PLL1 Output Frequency

The user should wait for:

1. The LFAST_PLLLSR[PLLDIS] =’0’, indicating that the PLL is enabled
2. Then, the LFAST_PLLLSR[PLDCR] =’1’, indicating that the PLL is locked and ready for use

$$\text{PLL Clock} = \left(\frac{\text{clk_in}}{\text{d3g_pll_pre_div}}\right) \times (\text{d3g_pll_feedback_div}) \quad \text{where d3g_pll_fraction_div_en=0}$$

Equation 2. PLL Output Frequency

1.14 LINFlexD

1.14.1 Introduction

The LINFlexD controller is designed to manage a high number of LIN messages efficiently with a minimum of CPU load. To reduce the CPU loading in Master mode, LINFlexD autonomously handles the LIN messages once software has triggered the header transmission, until the next header transmission request in transmitter mode or until checksum reception in the receiver mode.

The LINFlexD supports LIN protocol version 1.3, 2.0, 2.1 , and 2.2. It also consists of an 8-byte buffer for transmission/reception data.

The LINFlexD also provides support for some of the basic UART transfers of 8-bit, 9-bit, 16-bit, and 17-bit frames and also 12-bit data frame + parity reception in UART mode for MSC support.

The LINFlexD supports multi-channels and parametric DMA Tx/Rx interface in LIN/ UART operating mode.

1.14.2 LIN mode features

- Supports LIN protocol version 1.3, 2.0, 2.1, and 2.2
- Bit rates up to 20 Kbit/s (LIN protocol)
- Master/Slave mode

- Classic and Enhanced Checksum calculation and check
- Single 8-byte buffer or FIFO for Transmission/Reception
- Timeout management
- Identifier filters
- DMA interface
- Supports a maximum of 16 possible identifiers
- Master mode with autonomous message handling
- Extended frame mode for in-application programming purposes
- Wakeup event on dominant bit detection
- True LIN field state machine
- Advanced LIN error detection
- Header, response, and frame timeout
- Slave mode
  - Autonomous header handling
  - Autonomous transmit/receive data handling
- Identifier filters for autonomous message handling in Slave mode
- Separate clock for baud rate calculation
  - The relationship “(2/3)* LIN_CLK > PBRIDGEx_CLK > 1/3*LIN_CLK” should be maintained.

1.14.3 DMA Tx Enable Register (LINFlexD_DMATXE)

This register enables the DMA TX interface. This register can be written and read by software anytime.

The address offset depends on no_of_filters. Refer to chip configuration details to see the number of filters used in the device.

Address: 0h base + 98h offset = 98h

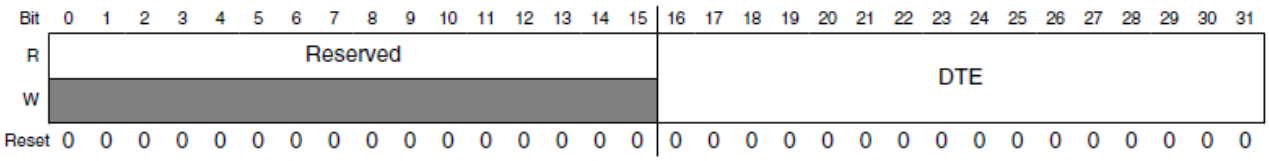


Table 1-19. LINFlexD_DMATXE field descriptions

Field	Description
0–15 Reserved	This field is reserved. Reserved **NOTE**: The number of reserved bits varies, and is equal to 32 - $2^{TX_CH_NUM}$. These lies from 0 to (31 - $2^{TX_CH_NUM}$). Refer to the chip configuration details for the value of TX_CH_NUM used in this device.
16–31 DTE	DMA Tx channel Y enable **NOTE**: The actual size of the register DMATXE depends on the value of the static parameter TX_CH_NUM. The size is [$2^{TX_CH_NUM}$ - 1 : 0]. The position of these bits are (32 - $2^{TX_CH_NUM}$) to 31. When DMATXE = 0x00000000, the DMA TX interface FSM is forced (soft reset) in Idle state. **NOTE**: Refer to the chip configuration details for the value of TX_CH_NUM used in this device. 0 DMA Tx channel Y disabled 1 DMA Tx channel Y enabled

1.14.4 DMA Rx Enable Register (LINFlexD_DMARXE)

This register enables the DMA RX interface. This register can be written and read by software any time.

The address offset depends on no_of_filters. Refer to device configuration chapter to see the number of filters used in the device.

Address: 0h base + 9Ch offset = 9Ch

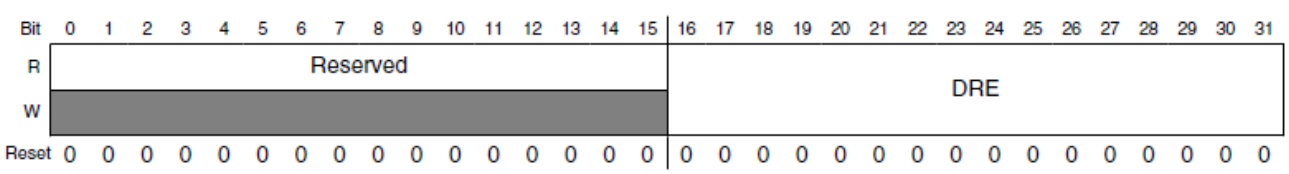


Table 1-20. LINFlexD_DMARXE field descriptions

Field	Description
0–15 Reserved	This field is reserved. Reserved

Table continues on the next page...

Table 1-20. LINFlexD_DMARXE field descriptions (continued)

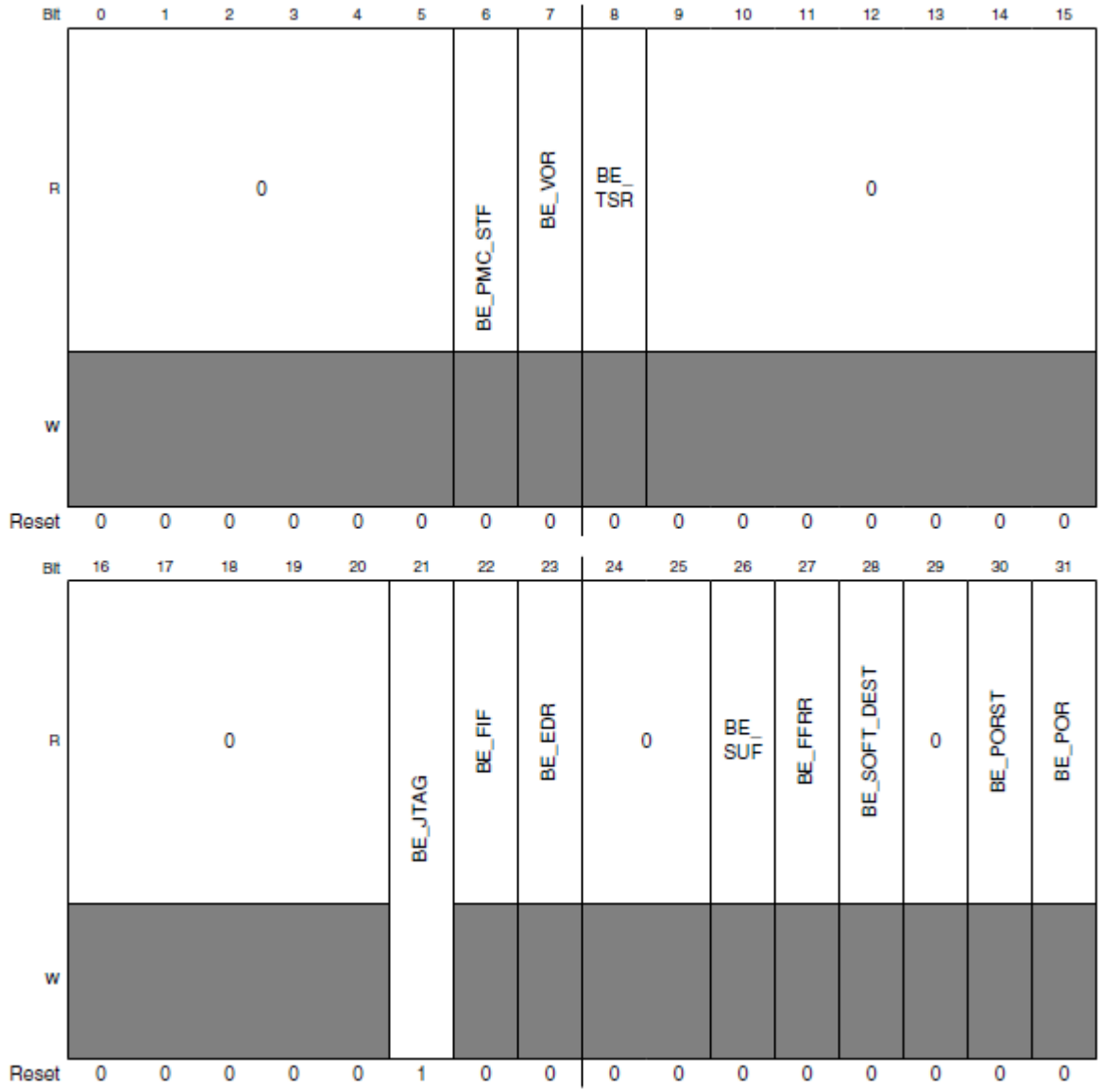
Field	Description
	NOTE: The number of reserved bits varies, and is equal to 32-DMA_RX_CH_NUM. Refer to the chip configuration details for the value of DMA_RX_CH_NUM used in this device.
16–31 DRE	DMA Rx channel Y enable. The number of DRE bits varies, and is equal to DMA_RX_CH_NUM. When DMARXE = 0x00000000, the DMA RX interface FSM is forced (soft reset) in idle state. **NOTE**: Refer to the chip configuration details for the value of DMA_RX_CH_NUM used in this device. 0 DMA Rx channel Y disabled 1 DMA Rx channel Y enabled

1.15 'Destructive' Bidirectional Reset Enable (MC_RGM_DBRE)

This register enables the generation of an external reset on 'destructive' reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

This register is reset only on power-on.

Address: 0h base + 30h offset = 30h



This register enables the generation of an external reset on 'destructive' reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

This register is reset only on power-on.

Table 1-21. MC_RGM_DBRE field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6	Bidirectional Reset Enables for PMC Selt Test Failure 'Destructive' Reset

Table continues on the next page...

Table 1-21. MC_RGM_DBRE field descriptions (continued)

Field	Description
BE_PMC_STF	0 RESET_B is asserted on a PMC Selt Test Failure 'destructive' reset event if the reset is enabled
7 BE_VOR	Bidirectional Reset Enables for Voltage Out of Range 'Destructive' Reset 0 RESET_B is asserted on a Voltage Out of Range 'destructive' reset event if the reset is enabled
8 BE_TSR	Bidirectional Reset Enables for Temperature sensor 'Destructive' Reset 0 RESET_B is asserted on a Temperature sensor 'destructive' reset event if the reset is enabled
9–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 BE_JTAG	Bidirectional Reset Enables for JTAG destructive reset 'Destructive' Reset 0 RESET_B is asserted on a JTAG 'destructive' reset event if the reset is enabled 1 RESET_B is not asserted on a JTAG destructive reset event
22 BE_FIF	Bidirectional Reset Enables for Flash Initialization Failure 'Destructive' Reset 0 RESET_B is asserted on a Flash Initialization Failure 'destructive' reset event if the reset is enabled
23 BE_EDR	Bidirectional Reset Enables for 'Functional' reset Escalation 'Destructive' Reset 0 RESET_B is asserted on a 'Functional' reset Escalation 'destructive' reset event if the reset is enabled
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 BE_SUF	Bidirectional Reset Enables for STCU Unrecoverable Fault 'Destructive' Reset 0 RESET_B is asserted on an STCU Unrecoverable Fault 'destructive' reset event if the reset is enabled
27 BE_FFRR	Bidirectional Reset Enables for FCCU Failure to React reset 'Destructive' Reset 0 RESET_B is asserted on an FCCU Failure to React reset 'destructive' reset event if the reset is enabled
28 BE_SOFT_DEST	Bidirectional Reset Enables for Software 'Destructive' Reset 0 RESET_B is asserted on a Software destructive reset 'destructive' reset event if the reset is enabled
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 BE_PORST	Bidirectional Reset Enables for External PowerOn reset 'Destructive' Reset 0 RESET_B is asserted on an External PowerOn reset 'destructive' reset event if the reset is enabled
31 BE_POR	Bidirectional Reset Enables for Power On 'Destructive' Reset 0 RESET_B is asserted on a Power On 'destructive' reset event if the reset is enabled

1.16 Power Management

1.16.1 Voltage Monitoring

The PMC senses the supply voltages continuously to keep the chip in safe operating range. Destructive or functional reset, depending on the PMC_RES register, or alternatively interrupts, are triggered when the voltage exceeds or falls below the limits defined by recommended electrical operating conditions. Please refer to the PMC_REE and PMC_RES register descriptions in the PMC chapter for more details.

1.16.2 ADC Interface

The ADC interface provides the PMC internal voltages for diagnostic purposes. The analog multiplexer and the buffer is inside the PMC. The SAR ADC0 converts the selected voltage, which can then be evaluated by the CPU. Refer to the PMC chapter, PMC_ADC_CS register description for details. See the Table 1-25.

1.17 Power Management Controller

1.17.1 PMC introduction

The power management controller (PMC) consists of two blocks: an analog power management block (and a digital controller for the analog block).

The PMC block contains all of the registers and digital logic to generate all of the enable signals, trim control bits, reset generation, and test mode logic for the analog block. The PMC digital logic also contains the controls for PMC analog outputs to be sensed with the ADC module.

In addition, the PMC digital logic contains the enable signals, trim bits and other registers for the temperature sensor. There are also several custom interfaces to various other blocks: The Reset Generation Module (RGM) block receives the LVD values during POR to use for phase transition conditions.

1.17.2 Reset Enable Control Register (PMC_REE)

This Reset Event Enable Register controls whether the voltage detect signal event will cause a reset or not. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

NOTE

This register is cleared on POR.

Address: 0h base + E0h offset = E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	LVD_PMC	0			LVD_SAR_ADC	0			LVD_IO	HVD_FLASH	LVD_FLASH	HVD_HV	HVD_CORE	LVD_CORE_HOT	LVD_CORE_COLD
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0	1*

* Notes:
- The reset value is chip-specific. See the chip-specific PMC information for the reset value.

Table 1-22. PMC_REE field descriptions

Field	Description
31–15	This field is reserved.

Table continues on the next page...

Table 1-22. PMC_REE field descriptions (continued)

Field	Description
Reserved	This read-only field is reserved and always has the value 0.
14 LVD_PMC	This bit enable/disable the generation of Reset. 0 Reset is disabled. NO reset either functional or destructive, will be generated if an event is detected on this supply 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply
13–11 Reserved	Reserved. Read only zero. This field is reserved. This read-only field is reserved and always has the value 0. 0 Reset is disabled, no reset will be generated even if this event occurred. 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply
10 LVD_SAR_ADC	Reset Enable/Disable Control This bit enable/disable the generation of Reset. **NOTE**: Reset value is defined by the DCF_PMC_REE_CTRL DCF client. 0 Reset is disabled, no reset will be generated even if this event occurred. 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply
9–7 Reserved	Reserved. Read only zero. This field is reserved. This read-only field is reserved and always has the value 0. 0 Reset is disabled, no reset will be generated even if this event occurred. 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply
6 LVD_IO	This bit enable/disable the generation of Reset. 0 Reset is Disabled. NO reset either functional or destructive, will be generated if an event is detected on this supply 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply
5 HVD_FLASH	Reset Enable/Disable Control This bit enable/disable the generation of Reset. **NOTE**: Reset value is defined by the DCF_PMC_REE_CTRL DCF client. 0 Reset is disabled, no reset will be generated even if this event occurred. 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply
4 LVD_FLASH	Reset Enable/Disable Control 0 Reset is Disabled. NO reset either functional or destructive, will be generated if an event is detected on this supply 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply
3	Reset Enable/Disable Control

Table continues on the next page...

Table 1-22. PMC_REE field descriptions (continued)

Field	Description
HVD_HV	This bit enable/disable the generation of Reset. **NOTE**: Reset value is defined by the DCF_PMC_REE_CTRL DCF client. 0 Reset is disabled, no reset will be generated even if this event occurred. 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply
2 HVD_CORE	Reset Enable/Disable Control This bit enable/disable the generation of Reset. **NOTE**: Reset value is defined by the DCF_PMC_REE_CTRL DCF client. 0 Reset is disabled, no reset will be generated even if this event occurred. 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply
1 LVD_CORE_HOT	Reset Enable/Disable Control This bit enable/disable the generation of Reset. **NOTE**: Reset value is defined by the DCF_PMC_REE_CTRL DCF client. 0 Reset is disabled, no reset will be generated even if this event occurred. 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply
0 LVD_CORE_COLD	Reset Enable/Disable Control This bit enable/disable the generation of Reset. NOTE: Reset value is defined by the DCF_PMC_REE_CTRL DCF client. 0 Reset is disabled, no reset will be generated even if there is event occurred on this supply. 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply.

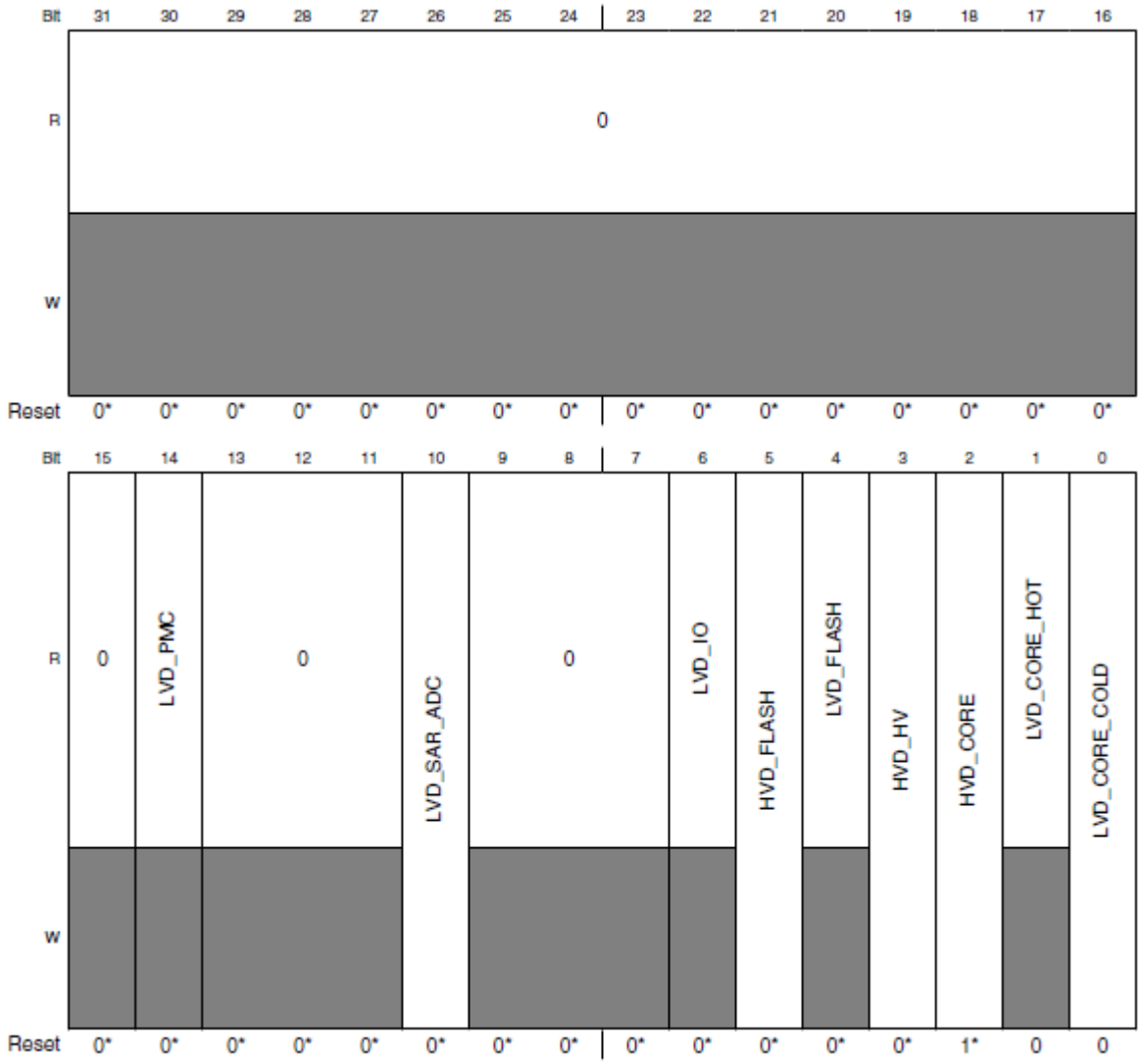
1.17.3 Reset Type (Destructive / Functional) Selection Register (PMC_RES)

This Reset Event Select Register controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enbaled in REE regsiter, the voltage detect event causes either a destructive or functional reset, when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence.

NOTE

This register is cleared on POR.

Address: 0h base + E4h offset = E4h



* Notes:

- The reset value is chip-specific. See the chip-specific PMC information for the reset value.

Table 1-23. PMC_RES field descriptions

Field	Description
31–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 LVD_PMC	Reset enabled/disabled. 0 Reset is Disabled. NO reset either functional or destructive, will be generated if an event is detected on this supply 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply
13–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

Table 1-23. PMC_RES field descriptions (continued)

Field	Description
	0 If the corresponding reset is enabled, then a Destructive Reset will be generated. 1 If the corresponding reset is enabled, then a Functional Reset will be generated.
10 LVD_SAR_ADC	This bit controls the enable/disable on the LVD SAR ADC supply. **NOTE**: Reset value is defined by the DCF_PMC_RES_CTRL DCF client. 0 If the corresponding reset is enabled, then a Destructive Reset will be generated. 1 If the corresponding reset is enabled, then a Functional Reset will be generated.
9–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0. 0 If the corresponding reset is enabled, then a Destructive Reset will be generated. 1 If the corresponding reset is enabled, then a Functional Reset will be generated.
6 LVD_IO	This bit enable/disable the generation of Reset. 0 If the corresponding reset is enabled, then a Destructive Reset will be generated. 1 If the corresponding reset is enabled, then a Functional Reset will be generated.
5 HVD_FLASH	This bit controls the enable/disable on the HVD FLASH supply. **NOTE**: Reset value is defined by the DCF_PMC_RES_CTRL DCF client. 0 If the corresponding reset is enabled, then a Destructive Reset will be generated. 1 If the corresponding reset is enabled, then a Functional Reset will be generated.
4 LVD_FLASH	This bit controls the enable/disable on the HVD FLASH supply. 0 If the corresponding reset is enabled, then a destructive Reset will be generated. 1 If the corresponding reset is enabled, then a Functional Reset will be generated.
3 HVD_HV	This bit controls the enable/disable on the VDD_HV_PMC supply. **NOTE**: Reset value is defined by the DCF_PMC_RES_CTRL DCF client. 0 If the corresponding reset is enabled, then a Destructive Reset will be generated. 1 If the corresponding reset is enabled, then a Functional Reset will be generated.
2 HVD_CORE	This bit controls the enable/disable on the HVD Core supply. **NOTE**: Reset value is defined by the DCF_PMC_RES_CTRL DCF client. 0 If the corresponding reset is enabled, then a Destructive Reset will be generated. 1 If the corresponding reset is enabled, then a Functional Reset will be generated.
1 LVD_CORE_ HOT	This bit enable/disable the generation of Reset. 0 If the corresponding reset is enabled, then a destructive Reset will be generated. 1 If the corresponding reset is enabled, then a Functional Reset will be generated.
0 LVD_CORE_ COLD	LVD Core Enable Disable Bit This bit controls the enable/disable on the LVD Core supply. **NOTE**: Reset value is defined by the DCF_PMC_RES_CTRL DCF client. 0 If the corresponding reset is enabled, then a Destructive Reset will be generated. 1 If the corresponding reset is enabled, then a Functional Reset will be generated

1.17.4 ADC Channel Select Register (PMC_ADC_CS)

This register contains the SRAM Brownout control and the analog channel mux of the analog PMC module.

Address: 0h base + E8h offset = E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0						STND_BY_RAM_BD_CLR	STND_BY_RAM_BR_DETCT	0		ADC_CHSE					
W								w1c								
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0	0	0	0	0	0

* Notes:

- The reset value is chip-specific. See the chip-specific PMC information for the reset value.

Table 1-24. PMC_ADC_CS field descriptions

Field	Description
31–10	This field is reserved.

Table continues on the next page...

Table 1-24. PMC_ADC_CS field descriptions (continued)

Field	Description
Reserved	This read-only field is reserved and always has the value 0.
9 STND_BY_ RAM_BO_CLR	Standby RAM Brownout clear bit. A write on this bit will generate one clock wide pulse. It is self clear pulse.
8 STND_BY_ RAM_BR_ DETCT	Standby RAM Brownout Detection register bit. This bit is W1C type. To clear this field, first write 1 to STND_BY_RAM_BO_CLR and then write 1 to this field (STND_BY_RAM_BR_DETCT). These two writes need to be separate write instructions.
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
ADC_CHSE	Analog-to-Digital Converter Channel Select. These bits are used to control the analog PMC's test mux going to the ADC. This is to probe the internal node voltages of the PMC for testing, calibration and characterization purposes by one of the ADCs of the SoC. To see what voltages are output from PMC in this ADC interface, please refer to Table 15. **NOTE**: While changing internal PMC signal exposed on ADC interface for diagnostics, it is required to change mux select lines to '000000' before changing to next valid combination. This is needed to avoid transient short circuit b/w internal nodes in PMC and a false LVD / HVD / POR assertion. This must be handled through software.

1.17.5 ADC Interface

ADC Interface is to probe the internal node voltages of PMC for testing, calibration and characterization purpose by one of the ADCs of the SoC. There will be a dedicated ADC channel for this interface.

PMC signals are buffered to ensure correct ADC sampling.

Node to be probed through ADC can be selected using bit "pmc_adc_ch_sel[5:0]".

Table 1-25. ADC Interface

S.No.	pmc_adc_ch_sel[5:0]	ADC Channel Output	Expected Value
0	6'b000000	ADC Channel OFF	
1	6’b011110	por085_c sense value	1.25V (CORE Supply)
2	6’b011101	por085_c reference value	0.915V
3	6’b011100	por098_c sense value	1.25V (CORE Supply)
4	6’b011011	por098_c reference value	1.01V
5	6'b011010	por085_c2 sense	1.25V (CORE SUPPLY)
6	6'b011001	por085_c2 reference value	0.905V

Table continues on the next page...

Table 1-25. ADC Interface (continued)

7	6'b011000	por098_c2 sense value	1.25V (CORE SUPPLY)
8	6'b010111	por098_c2 reference value	0.993V
9	6'b010110	lvd_core_cold sense	1.25V (CORE SUPPLY)
10	6'b010101	lvd_core_cold reference	1.24V
11	6'b010100	lvd_core_hot sense	1.25V (CORE SUPPLY)
12	6'b010011	lvd_core_hot reference	1.21V
13	6'b010010	hvd_core sense	1.25V (CORE SUPPLY)
14	6'b010001	hvd_core reference	1.43V
15	6'b010000	por260_c divider tap point	2.55V
16	6'b001111	por260_c reference	1.2V
17	6'b001110	por260_c2 divider tap point	2.55V
18	6'b001101	por260_c2 reference	1.2V
19	6'b001100	lvd_pmc divider tap point	1.98V
20	6'b001011	lvd_pmc reference	1.2V
21	6'b001010	hvd600 divider tap point	1.13V
22	6'b001001	hvd600 reference	1.2V
23	6b'001000	lvd_flash divider tap point	1.31V
24	6'b000111	lvd_flash reference	1.2V
25	6'b000110	hvd_flash divider tap point	1.13V
26	6'b000101	hvd_flash reference	1.2V
27	6'b000100	lvd_io divider tap point	1.97V
28	6'b000011	lvd_io reference	1.2V
29	6'b000010	Reserved	
30	6'b000001	Reserved	
31	6'b100000	Reserved	
32	6'b111111	Reserved	
33	6'b111110	Reserved	
34	6'b111101	Reserved	
35	6'b111100	Reserved	
36	6'b111011	Reserved	
37	6'b111010	lvd_buddy reference	1.05V
38	6'b111001	lvd_buddy sense	1.2V(BUDDY Device Supply)
39	6'b111000	lvd_sar_adc divider tap point	2.25V
40	6'b110111	lvd_sar_adc reference	1.2V
41	6'b110110	hvd_sar_adc divider tap point	1.13V
42	6'b110101	hvd_sar_adc reference	1.2V
43	6'b110100	bandgap voltage with only curve trimming	1.205V
44	6'b110011	nwellbias regulator output	1.25V
45	6'b110010	nwellbias regulator reference	1.25V
59	6'b100100	ioseg3_sense	VDD_HV_IO Segment3

Table continues on the next page...

Table 1-25. ADC Interface (continued)

60	6'b100011	ioseg2_sense	VDD_HV_IO Segment2
61	6'b100010	vrefh_sd_adc	5V
62	6'b100001	vrefh_sd_adc	5V
63	6'b100000	vss_sd_adc	0
	Others	Reserved	

1.18 Power Control Unit (MC_PCU)

1.18.1 Introduction

The function of this module is simply to map the PMC digital interface IP into its address space.

The MC_PCU module plays a different role in this microcontroller than it does in other microcontrollers. Instead of directly providing an interface to configure and control power distribution to various segments ("power domains") of the chip, the MC_PCU acts as an integration module.

This microcontroller has a single power domain, Power Domain 0, which is managed by the Power Management Controller (PMC). The 16K address space consists of a single MC_PCU power domain status register and registers that configure and control the PMC.

1.18.2 Functional description

Please see the PMC chapter for specific details on the PMC control registers.

1.19 NXMC block connections

To provide Nexus data trace messaging from bus masters on the master ports of the crossbar (XBAR) that do not inherently have a trace client, a Nexus Crossbar Multi-Master Client (NXMC) monitors traffic into the master port of the XBAR. Two NXMCs are implemented: one for the fast masters and one for the slow masters. Trace messages transmitted over the Nexus interface (or stored in trace memory) include which NXMC generated the message as well as an identifier for the pre-concentrator source of the message. In addition to XBAR bus master trace capability, the NXMC also allows for in-

circuit trace capability that can be used to trace the Sequence Processing Unit (SPU) counter values. The two external hardware watchpoint triggers of each NXMC are used to generate timestamps (four total—two from each NXMC) for events from the SPU.[1]

The following figure shows a block diagram illustrating the NXMC integration.

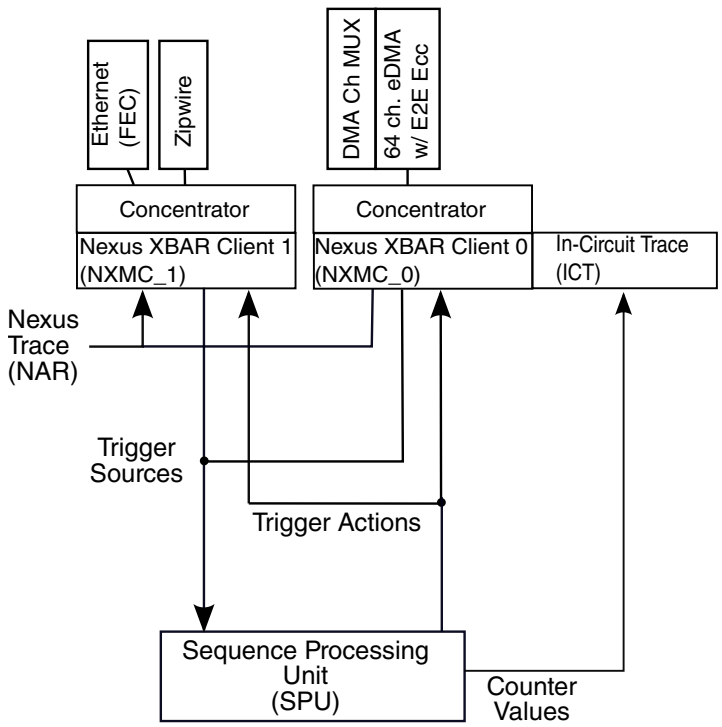


Figure 1-8. NXMC integration block diagram

The values of the trace master ID or in-circuit client ID, along with the Nexus message source identifier for the different clients are shown in the following table.[2]

Table 1-26. NXMC source and master client IDs

Client		Nexus trace source ID (SRC)	Trace master ID or In-circuit trace client ID (MASTER)
In-Circuit Trace	SPU	0b1011	0b0000
XBAR Client 0 Concentrator	eDMA Controller	0b1100	0b0011
(fast masters)	Reserved		All other values
XBAR Client 1 Concentrator	Ethernet	0b1101	0b0100
(slow masters)	Zipwire		0b1000
	Reserved		All other values

1. To provide Nexus data trace messaging from bus masters on the master ports of the crossbar (XBAR) that do not inherently have a trace client, a Nexus Crossbar Multi- Master Client (NXMC) monitors traffic into the master port of the XBAR. Two NXMCs are implemented: one for the fast masters and one for the slow masters.
2. NXMC clients are connected between E2E-ECC master client and AXBS. Therefore any corrupted data, correctable (single bit error) or in-correctable data (multi bit error), is traced without any trace error information. ECC errors are captured in MEMU.

1.20 Power

There are two types of voltage supervisors on the MPC5746R, Low Voltage Detect (LVD) and High Voltage Detect (HVD) monitors. Safety relevant voltages (recommended operating voltages), are supervised for voltage levels that are out of correct operating voltage ranges. Since any voltage running outside of the correct operating voltage range has the potential to disable the failure indication mechanisms of the MCU, the indication of these voltage variations can be used to cause a direct transition of the MCU into the safe state (reset assertion) (see "Power Management Controller (PMC)" and "Power Control Unit (MC_PCU)" chapters for details on power monitoring).

1.21 Fault Collection and Control Unit (FCCU)

1.21.1 Available FCCU output signals

The following table shows the available FCCU output signals implemented on this chip.

Table 1-27. FCCU output signals

FCCU output signal name	Chip-top signal name
EOUT[0]	ERROR0
EOUT[1]	ERROR1

NOTE

The FCCU EOUT signals can toggle after LBIST execution during reset.

1.21.2 FCCU failure inputs from other modules

The following table shows the different failure input signals of the FCCU, how they can be tested, and their default configuration after reset.

NOTE

All the faults shown in the following table require a long functional reset.

NOTE

FCCU Fault Channels 22, 23, and 24 may be asserted after reset. FCCU Fault Channels 51, 52, 69 through 84, 92, and 93 may be asserted during the execution of MBIST and LBIST. Before running MBIST or LBIST, these channels should be configured to NO REACTION. If any of these channels become asserted during MBIST or LBIST testing, the fault should simply be cleared and no further action should be taken.

Table 1-28. FCCU Fault Inputs

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
0	No Reaction	Temp sensor	Using PMC_TS_FIR register to set the fault at source + Via FCCU Fake bit	Temperature Sensor Event Triggered by PMC_ESR_TD[TEMP1_2, TEMP1_0, TEMP0_2, TEMP0_0] and PMC_TS_FIR[TSNS_FAULT] . For details, see the descriptions of the PMC_ESR_TD and PMC_TS_FIR registers.
1	No Reaction	PMC	Using PMC FIR register to set the fault at source + Via FCCU Fake bit	LVD event to FCCU Triggerred by PMC_LVD_HVD_EVENT_STATUS[LVD_PMC, LVD_SAR_ADC, LVD_IO, LVD_FLASH, LVD_CORE_HOT, LVD_CORE_COLD] and PMC_FIR[LVD_FAULT]. For details, see the descriptions of the PMC_LVD_HVD_EVENT_STATUS and PMC_FIR registers.
2	No Reaction	PMC	Using PMC FIR register to set the fault at source + Via FCCU Fake bit	HVD Event to FCCU Triggered by PMC_LVD_HVD_EVENT_STATUS[HVD_FLASH, HVD_HV, HVD_CORE]. For details, see the descriptions of the PMC_LVD_HVD_EVENT_STATUS register.
3	No Reaction	PMC	Using PMC FIR register to set the fault at source + Via FCCU Fake bit	PMC Self Test Failure Event PMC hard/soft self-test failed: LVD/HVD **NOTE:** When this occurs, single LVD or HVD self-tests should be executed to identify which detection mechanism failed in the SOC: LVD or HVD. For details, see the description of the PMC_SELF_TEST_UM_VD_REG register as well as the flag bits associated to NCF3 (PMC_SELF_TEST_UM_VD_REG [ST_RESULT]). Self test bit definition indicates all valid values for PMC_SELF_TEST_UM_VD_REG[VD_ST_CTRL]

Table continues on the next page...

Table 1-28. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
				while executing single self-tests. LVD-HVD self test mechanism provides details about the available PMC self-tests
4	No Reaction	SSCM	write wrong data to address + Via FCCU Fake bit to set the fault at error path	See SSCM_STATUS[CERS] field description.
5	No Reaction	SSCM	write wrong data to address + Via FCCU Fake bit to set the fault at error path	See SSCM_STATUS[CER] field description.
6	long reset	STCU	SW check of BIST MISR + Via FCCU Fake bit to set the fault at source	Unrecoverable Fault of STCU. Set when STCU2_ERR_STAT[UFSF] is set.
7	No Reaction	STCU	SW check of BIST MISR + Via FCCU Fake bit to set fault at source	Recoverable Fault of STCU. Set when STCU2_ERR_STAT[RFSF] is set.
8	No Reaction	STCU	STCU activation during user application run Via FCCU Fake bit to set the fault at source	The fault in case the self-test was wrongly triggered without STCU programming. This fault is set when the self-test was triggered without STCU2_RUN[RUN] bit programming.
9	No Reaction	Checker Shell	Via FCCU Fake bit	out of lockstep fault detected for miscellaneous safety core output
10	No Reaction	Checker Shell	Via FCCU Fake bit to set the fault at error source	out of lockstep fault detected for safety Core dtcm array interface signal
11	No Reaction	Checker Shell	Via FCCU Fake bit to set the fault at error source	out of lockstep fault detected for safety Core I Cache array interface signals
12	Reserved	Reserved		Reserved
13	No Reaction	Checker Shell	Via FCCU Fake bit to set the fault at error source	out of lockstep fault detected for safety Core Amba interface signals
14	No Reaction	DMA Checker Shell	Via FCCU Fake bit to set the fault at error source	out of lockstep fault detected for DMA
15	No Reaction	DMC Checker	Via FCCU Fake bit to set the fault at error source	out of lockstep fault detected for core0 TCM DMC
16	RESERVED	RESERVED		RESERVED
17	No Reaction	SWT3	Emulate fault using functionality + Via FCCU Fake bit to set the fault at error path	Software watchdog timer reset request. This input must be mapped only to reset, and must not be mapped to an alarm.
18	RESERVED	RESERVED		RESERVED
19	Long Reset	SWT1	Emulate fault using functionality (SW test by not triggering SWT during startup) + Via FCCU Fake bit to set the fault at error path	Software watchdog timer reset request. This input must be mapped only to reset, and must not be mapped to an alarm.

Table continues on the next page...

Table 1-28. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
20	No Reaction	SWT0	Emulate fault using functionality (SW test by not triggering SWT during startup) + Via FCCU Fake bit to set the fault at error path	Software watchdog timer interrupt
21	No Reaction	SWT0	Emulate fault using functionality (SW test by not triggering SWT during startup) +Via FCCU Fake bit to set the fault at error path	Software watchdog timer reset request. This input must be mapped only to reset, and must not be mapped to an alarm.
22	No Reaction	MEMU	Injection of RAM ECC error via ECC bit access and later reading + Via FCCU Fake bit to set the fault at error path+ LBIST	System RAM Correctable Error detected
23	No Reaction	MEMU	Injection of RAM ECC error via ECC bit access and later reading + Via FCCU Fake bit to set the fault at error path + LBIST	System RAM Un-Correctable Error detected
24	No Reaction	MEMU	Via FCCU Fake bit to set the fault at error path + LBIST	System RAM error Overflow
25	No Reaction	MEMU + eTPU Memory Error Correction Interrupt	Via FCCU Fake bit to set the fault at error path + LBIST	Peripheral RAM Correctable error detected. Set eTPU_MESR_MECR[MECIE] to enable this error.
26	No Reaction	MEMU + eTPU Global Exception	Via FCCU Fake bit to set the fault at error path + LBIST	Peripheral RAM UN-Correctable error detected. Set eTPU_MESR_MECR[MECIE] to enable this error.
27	No Reaction	MEMU	Via FCCU Fake bit to set the fault at error path + LBIST	Periph RAM error Overflow
28	No Reaction	MEMU	Reading preprogrammed correctable ECC error from UTEST flash memory + Via FCCU Fake bit to set the fault at error path + LBSIT	Flash Correctable Error detected NCF correlated to C55FMC_MCR[SBC] bit **NOTE:** From the MEMU perspective this is indicated by MEMU_FLASH_CERR_STSn. For details, see the description of the C55FMC_MCR and MEMU_FLASH_CERR_STSn registers.
29	No Reaction	MEMU	Reading preprogrammed uncorrectable ECC error from UTEST flash memory + Via FCCU Fake bit to set the fault at error path + LBIST	Flash Un-Correctable Error detected NCF correlated to C55FMC_MCR[EER] bit **NOTE:** From the MEMU perspective this is indicated by MEMU_FLASH_UNCERR_STSn. For

Table continues on the next page...

Table 1-28. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
				details, see the description of the C55FMC_MCR and MEMU_FLASH_UNCERR_STS registers.
30	No Reaction	MEMU	Via FCCU Fake bit to set the fault at error path + LBIST	Flash Error Overflow For details on conditions which trigger this NCF, see section Handling overflows (Multiple error reporting). **NOTE:** From the MEMU perspective this is indicated by MEMU_FLASH_OFLWn. For details, see the description of the MEMU_FLASH_OFLWn register.
31	RESERVED	RESERVED		RESERVED
32	No Reaction	PLL0	Forced loss by PLL reconfiguration + Via FCCU Fake bit to set the fault at error path	Loss of Lock detected for NON-FM PLL **NOTE:** This source should be configured to generate a reset instead of an interrupt if the source is used as the system clock.
33	No Reaction	PLL1	Forced loss by PLL reconfiguration + Via FCCU Fake bit to set the fault at error path	Loss of Lock detected for FM PLL **NOTE:** This source should be configured to generate a reset instead of an interrupt if the source is used as the system clock.
34	No Reaction	CMU_PLL	Test by misconfiguration + Via FCCU Fake bit to set the fault at error path	OLR Event from crystal clock monitor circuit of cmu_pll **NOTE:** This source should be configured to generate a reset instead of an interrupt if the source is used as the system clock.
35	No Reaction	CMU_PLL	Test by misconfiguration + Via FCCU Fake bit to set the fault at error path	FLL/FHH event from PLL Clock Monitor circuit. This fault is set when CMU_0 detects an FHH or FLL event **NOTE:** To identify which CMU module raised this NCF, read the CMU_ISR[FHHI/FLLI] bit fields from all relevant CMU instances. For details, see the description of the CMU_ISR register. For available CMU's, see sections Clock input sources and CMU registers and field availability.
36	No Reaction	System clock CMU	Test by misconfiguration + Via FCCU Fake bit to set the fault at error path	FLL/FHH event from System clock monitor circuit.

Table continues on the next page...

Table 1-28. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
				This fault is set when one of CMU_1 to CMU_3 detects an FHH or FLL event. **NOTE:** To identify which CMU module raised this NCF, read the CMU_ISR[FHHI/FLLI] bit fields from all relevant CMU instances. For details, see the description of the CMU_ISR register. For available CMU's, see sections Clock input sources and CMU registers and field availability.
37	No Reaction	Other CMU	Emulate fault using functionality + Via FCCU Fake bit to set the fault at error path	FLL/FHH event from other clock monitor circuit. This fault is set when one of CMU_4 to CMU_12 detects an FHH or FLL event. **NOTE:** To identify which CMU module raised this NCF, read the CMU_ISR[FHHI/FLLI] bit fields from all relevant CMU instances. For details, see the description of the CMU_ISR register. For available CMU's, see sections Clock input sources and CMU registers and field availability.
38	No Reaction	Periph AXBS	Using XBIC_EIR register to set the fault at source+Via FCCU Fake bit to set the fault at error path	Peripheral Crossbar alarm to FCCU
39	No Reaction	Comp AXBS	Using XBIC_EIR register to set the fault at source+Via FCCU Fake bit to set the fault at error path	Computational Crossbar alarm to FCCU
40	No Reaction	Flash	Via FCCU Fake bit to set the fault at error source + LBIST	alarm indicating the flash detected an error in the ECC correction logic through an EDC
41	No Reaction	Flash Controller	Via FCCU Fake bit to set the fault at error path + LBIST	alarm indicating the flash controller detected an error in the address ECC manipulation logic through an EDC
42	No Reaction	Flash	Via FCCU Fake bit to set the fault at error path + LBIST	alarm indicating the Flash Array Controller detected an Encode Address Error, such as multiple address lines selected or no address lines selected **NOTE:** Address Encode Errors detected within Flash Array [FCCU channel 42] will also assert the P-Flash [FCCU channel 43]
43	No Reaction	Flash Controller	Via FCCU Fake bit to set the fault at error path + LBIST	alarm indicating the flash controller detected a transaction monitor mismatch when compared to the flash safety feedback outputs

Table continues on the next page...

Table 1-28. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
44	No Reaction	Flash Controller	Via FCCU Fake bit to set the fault at error path + LBIST	alarm indicating the flash controller detected a transaction monitor mismatch when compared to the on-chip calibration RAM safety feedback outputs
45	No Reaction	Flash Controller	Via FCCU Fake bit to set the fault at error path + LBIST	alarm indicating the flash controller detected a transaction monitor mismatch in the pseudo-replicated calibration evaluation hardware
46	RESERVED	RESERVED		RESERVED
47	No Reaction	Platform Gasket	Via FCCU Fake bit to set the fault at error path	2:1 bridging gasket transaction monitor alarm
48	No Reaction	Platform Gasket	Via FCCU Fake bit to set the fault at error path	1:2 bridging gasket transaction monitor alarm
49	No Reaction	SMPU	Via FCCU Fake bit to set the fault at error path + LBIST	Computational MPU alarm to the fault collection unit An access error has happened due to restrictions specified by the MPU region descriptors. This is in addition to the MPU responding to the requesting master with an error response.
50	No Reaction	SMPU	Via FCCU Fake bit to set the fault at error path + LBIST	Peripheral MPU alarm to the fault collection unit An access error has happened due to restrictions specified by the MPU region descriptors. This is in addition to the MPU responding to the requesting master with an error response.
51	No Reaction	Decorated memory Controller	No fake faults supported	AIPS decorated storage transaction monitor error: this monitor basically ensure that the Decorated Memory Controller is not modifying regular AHB transactions when a non-decorated storage instruction is performed. If a non-decorated storage instruction is executed, the DMC should pass the transaction through unaltered.
52	No Reaction	Decorated memory Controller	No fake faults supported	AIPS decorated storage transaction monitor error: this monitor basically ensure that the Decorated Memory Controller is not modifying regular AHB transactions when a non-decorated storage instruction is performed. If a non-decorated storage instruction is executed, the DMC should pass the transaction through unaltered.
53	No Reaction	PRAM Controller	Via FCCU Fake bit to set the fault at error path	PRAM Controller/AHB bus transaction monitor error output
54	RESERVED	RESERVED		RESERVED

Table continues on the next page...

Table 1-28. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
55	No Reaction	Core	Willfully cause an appropriate exception (for example, access a non-existing address) + Via FCCU Fake bit to set the fault at error path	machine check has occurred
56	No Reaction	Core	Willfully cause an appropriate exception (for example, access a non-existing address) + Via FCCU Fake bit to set the fault at error path	machine check has occurred
57	No Reaction	SOC	Disabling Checker core while safety core is on+ Via FCCU Fake bit to set the fault at error path	Lock Step Disabled
58	No Reaction	RGM	RGM “safe mode” request + Via FCCU Fake bit to set the fault at error path + LBIST	Safe Mode request from RGM
59	RESERVED	RESERVED		-
60	No Reaction	IOMUX	External activation of EIN. External entity has to pull the error input low. + Via FCCU Fake bit to set the fault at error path	Error Input
61	RESERVED	RESERVED		RESERVED
62	No Reaction	DCI	Via FCCU Fake bit to set the fault at error path + LBIST	Debug Fault Condition to MCU • If any JTAG_aux_tms is high from DCI • If Jtag_rst_b is low from DCI • If JTAG mode is high from DCI • If nex_rst_b is high from DCI
63	No Reaction	SAR ADC0/1/2/3	VIA SAR-ADC STCR2[SERR] bit to set the fault at error source + Via FCCU Fake bit to set the fault at error path	Non Critical fault from ALL SARs
64	No Reaction	SAR ADC0/1/2/3	VIA SAR-ADC STCR2[SERR] bit to set the fault at error source + Via FCCU Fake bit to set the fault at error path	Critical Fault from All SARs
69	No Reaction	CoreA Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data0 feedback check alarm
70	No Reaction	CoreA Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data1 feedback check alarm

Table continues on the next page...

Table 1-28. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
71	No Reaction	CoreA Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data2 feedback check alarm
72	No Reaction	CoreA Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data3 feedback check alarm
73	No Reaction	CoreA Instruction Tag	Via FCCU Fake bit to set the fault at error path	IC tag feedback check alarm
74	No Reaction	CoreA DMEM Controller	Via FCCU Fake bit to set the fault at error path	DTCM0 feedback check alarm
75	No Reaction	CoreA DMEM Controller	Via FCCU Fake bit to set the fault at error path	DTCM1 feedback check alarm
76	No Reaction	CoreA IMEM Controller	Via FCCU Fake bit to set the fault at error path	ITCM feedback check alarm
77	No Reaction	CoreB Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data0 feedback check alarm
78	No Reaction	CoreB Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data1 feedback check alarm
79	No Reaction	CoreB Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data2 feedback check alarm
80	No Reaction	CoreB Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data3 feedback check alarm
81	No Reaction	CoreB Instruction Tag	Via FCCU Fake bit to set the fault at error path	IC tag feedback check alarm
82	No Reaction	CoreB DMEM Controller	Via FCCU Fake bit to set the fault at error path	DTCM0 feedback check alarm
83	No Reaction	CoreB DMEM Controller	Via FCCU Fake bit to set the fault at error path	DTCM1 feedback check alarm
84	No Reaction	CoreB IMEM Controller	Via FCCU Fake bit to set the fault at error path	ITCM feedback check alarm
85	No Reaction	CoreA/0 interrupt controller	This alarm can be triggered by enabling the INTC_MMR and setting the INTC_TIMER value to be larger than the INTC_LAT value (which itself needs to be non-zero). + Via FCCU Fake bit to set the fault at error path	INTC monitor error
86	No Reaction	CoreB/1 Interrupt controller	This alarm can be triggered by enabling the INTC_MMR and setting the INTC_TIMER value	INTC monitor error

Table continues on the next page...

Table 1-28. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
			to be larger than the INTC_LAT value (which itself needs to be non-zero). + Via FCCU Fake bit to set the fault at error path	
87	No Reaction	Decorated memory Controller	Via FCCU Fake bit to set the fault at error path	PRAM decorated storage transaction monitor error
88	No Reaction	Computational SMPU	Via FCCU Fake bit to set the fault at error path + LBIST	MPU alarm to the fault collection unit An internal error has happened in the operation of the MPU. Example: MPU erroneously blocked transactions without a real permissions violation will assert this NCF.
89	No Reaction	peripheral SMPU	Via FCCU Fake bit to set the fault at error path + LBIST	MPU transaction monitor error output An internal error has happened in the operation of the MPU. Example: MPU erroneously blocked transactions without a real permissions violation will assert this NCF.
90	No Reaction	Core0/1 IMEM MBIST	Via FCCU Fake bit to set the fault at error path	Mbist multi bit error of IMEM (Core0 or Core1)
91	No Reaction	Flash	Via FCCU Fake bit to set the fault at error path + LBIST	Flash Read Voltage error or reference current error
92	No Reaction	Core0/1 DMEM MBIST	Via FCCU Fake bit to set the fault at error path	Mbist multi bit error of DMEM (Core0 or Core1)
93	No Reaction	Core0/1 ICACHE/TAG MBIST	Via FCCU Fake bit to set the fault at error path	Mbist multi bit error of ICACHE(Core0 or Core1)
94	No Reaction	SRAM MBIST	Via FCCU Fake bit to set the fault at error path	Mbist Multibit error of SRAM
95	No reaction	All Peripheral RAM MBIST	Via FCCU Fake bit to set the fault at error path	Mbist multi bit error from all peripheral RAM

1.22 Self-Test Control Unit (STCU2)

1.22.1 STCU2 chip-specific module information

This section describes the chip-specific implementation of the STCU2 module.

NOTE

Before running either online or offline self-test, user needs to mask the core LVD, but should enable the core LVDs before running the application via SW through the PMC_REE register.

1.22.2 STCU2 Off-Line MBIST Status Medium Register (STCU2_MBSM)

The STCU2_MBSM register includes the results corresponding to the execution of the selected Off-Line MBIST in the range NMCUT = 32-63.

The size of the register depends on the number of BISTed RAMs/ROMs.

Address: 0h base + 48h offset = 48h

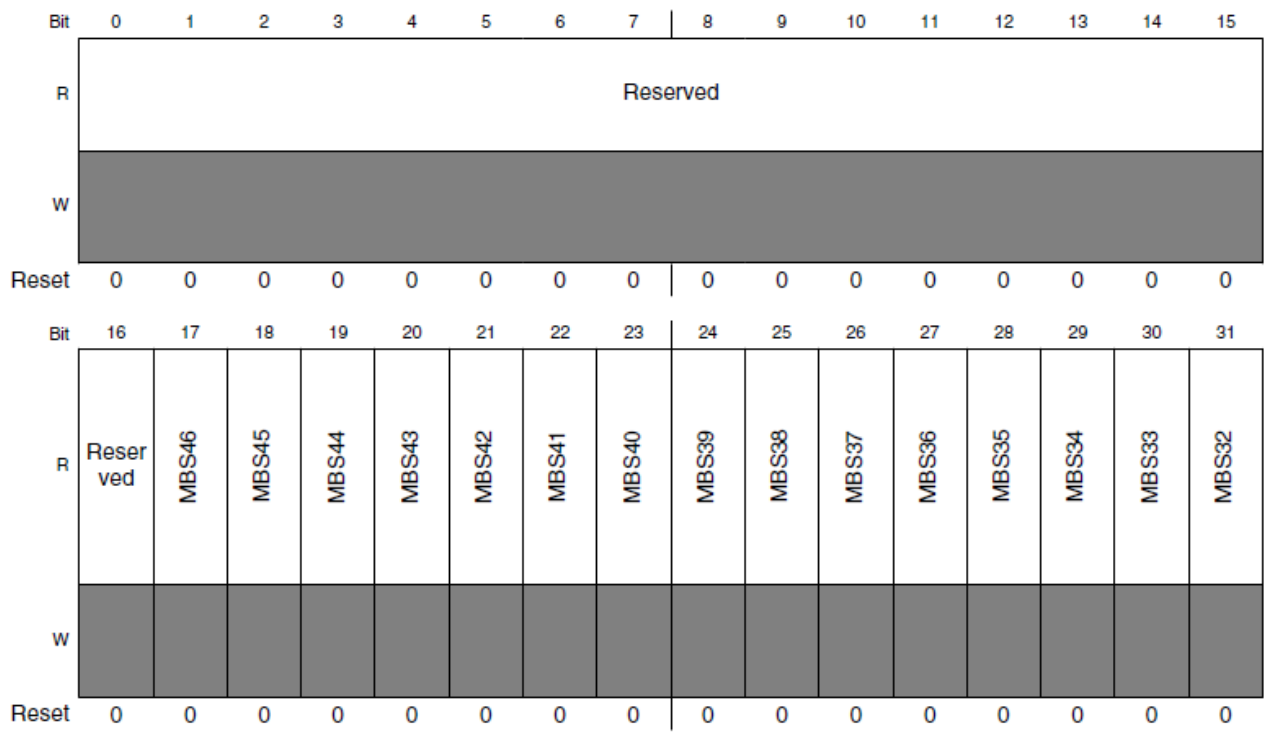


Table 1-29. MBSM field descriptions

Field	Description
0–16 Reserved	This field is reserved.
17 MBS46	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE**: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

Table 1-29. MBSM field descriptions (continued)

Field	Description
18 MBS45	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE**: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
19 MBS44	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE**: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
20 MBS43	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE**: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBS42	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE**: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBS41	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE**: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBS40	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE**: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBS39	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE**: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBS38	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE**: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

Table 1-29. MBSM field descriptions (continued)

Field	Description
26 MBS37	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE**: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBS36	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE**: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBS35	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE**: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBS34	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE**: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBS33	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE**: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBS32	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE**: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

1.23 PMC protected registers

Table 1-30. Protected PMC registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
TS_IER	32	A4h	32
REE_TD	32	ACh	32

Table continues on the next page...

Table 1-30. Protected PMC registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
RES_TD	32	B0h	32
CTL_TD	32	B4h	32
TS_FIR	32	C0h	32
FIR	32	D4h	32
PMC_VD_INTR_ENB	32	D8h	32
PMC_REE	32	E0h	32
PMC_RES	32	E4h	32
ADC_CS	32	E8h	32

1.24 BAR execution protection

The BAR (Boot Assist ROM) is ROM-based code, mapped within the AIPS space, which is used during FSL production phase. The BAR code makes it possible to upload data into the system RAM by reading the data directly from a GPIO port. During FSL Production test, the boot CPU1 processor starts executing from the BAR code. After FSL Production test, the CPU1 always starts executing the BAF code. After JDP/FSL Production lifecycle, it is not possible to accidentally or maliciously jump to the BAR code, since the access to this memory area is considered illegal.

The BAR ROM code hardware is designed to always generate an Illegal Peripheral Access Error when it is accessed. This is the same error that is caused by accessing a 'reserved' location in any peripheral.

1.25 Password and Device Security Module (PASS)

1.25.1 Password and Device Security Module (PASS) chip-specific module information

The PASS module is used to implement password-based read and write protection for flash blocks. Up to four levels of password protection can be implemented for each flash block.

NOTE

The registers discussed in this section are initialized via DCF records, so the reset values depend on user settings.

Each password group defined in the DCF records has a set of four LOCK*n* registers association with it—PASS_LOCK*n*_PG*n*. The bits of these registers are associated with specific flash blocks. The mapping is shown in the attached Flash Block Assignment spreadsheet.

NOTE

Password Challenge passwords and JTAG passwords that are all 0's or all 1's are invalid.

NOTE

For availability of RL0, RL1, and RL2 (Flash Region Lock bit fields) definitions in Password Group n Lock n Status Registers for the device, please refer to the Flash_Block_Assignment sheet in the pdf attachments.

The following table shows the the Debug Interface Access scenarios on this chip.

Table 1-31. Debug Interface Enable Truth Table

Input Conditions					Outcome
Lifecycle	**Production Disable mode**	**DCF Censorship**	**Debug Enable (LOCK3[DBL] bit)**	**JTAG Password**	**Debug Interface Enable**
Customer Delivery	Don't Care	Don't Care	Don't Care	Don't Care	Enabled
OEM Production	Don't Care	Not censored	Don't Care	Don't Care	Enabled
		Censored	Unlocked (DBL=0)	Don't Care	Enabled
			Locked (DBL=1)	Matched	Enabled
				Not Matched	Blocked
In Field	Off	Not censored	Don't Care	Don't Care	Enabled
		Censored	Unlocked (DBL=0)	Don't Care	Enabled
			Locked (DBL=1)	Matched	Enabled
				Not Matched	Blocked
	On	Don't Care	Don't Care	Don't Care	Blocked
Failure Analysis	Don't Care	Not censored	Don't Care	Don't Care	Enabled
		Censored	Unlocked (DBL=0)	Don't Care	Enabled
			Locked (DBL=1)	Matched	Enabled
				Not Matched	Blocked

The following table summarizes the scenarios of read lock (RL) region protection on this chip.

Table 1-32. Flash memory read protection truth table

Inputs				Outputs	
Life Cycle	**Censorship DCF client**	**Debug Interface**	**LOCK3[RLx]**	**Read Protected RL[0] (UTest)**	**Read Protected RL[3:1] (other regions)**
Customer Delivery	Don't care	Don't care	Don't care	Readable	Readable
OEM Production or In Field	0x55AA	Don't care	Don't care	Readable	Readable
	Any other value	Blocked	Don't care	Readable	Readable
		Enabled	Locked	Read Blocked	Read Blocked
			Unlocked	Readable	Readable
Failure Analysis	0x55AA	Don't care	Don't care	Readable	Readable
	Any other value	Don't care	Locked	Readable	Read Blocked
			Unlocked	Readable	Readable

The following table indicates byte-to-byte correspondence between UTEST DCF address and JTAG_PASSWORD.

Table 1-33. JTAG_PASSWORD register bit correspondence

UTEST Address Offset	Register Bit Position (MSB is 255)
0x00	JTAG_PASSWORD[7:0]
0x01	JTAG_PASSWORD[15:8]
...	...
0x1E	JTAG_PASSWORD[247:240]
0x1F	JTAG_PASSWORD[255:248]

1.25.2 Life Cycle Status Register (PASS_LCSTAT)

The Life Cycle Status Register reflects the production status and DCF Censorship client status of the device. It is possible to mature the device (move the device life cycle forward), but never to revert the life cycle to an earlier state.

The reset value of the LCSTAT register LIFE field derives from an external life cycle management module (like SSCM). It is a 3-bit life cycle. This register simply displays the life cycle input from external blocks.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CNS	JUN	1	0												
W																
Reset	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0													LIFE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

* Notes:
- CNS field: Reset value depends on the Censorship DCF client

Table 1-34. PASS_LCSTAT field description

Field	Description
0 CNS	This field indicates whether the DCF Censorship client has been written with a DCF record to 0x55AA or if the DCF Censorship client contains any other value than 0x55AA. The bit can be set (or cleared) via DCF Record in UTest Flash. **NOTE**: If no DCF record has been written to affect the value of the CNS bit, its reset value will be 1. 1 DCF Censorship Client value is anything other than 0x55AA. 0 DCF Censorship client value is 0x55AA.
1 JUN	This field indicates whether the device has been temporarily uncensored. Successful transmission of the JTAG password sets this bit to '1', otherwise it will be 0. Unsuccessful transmission clears this bit.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
3-28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29-31 LIFE	The life cycle setting is manipulated externally (SSCM). Pass displays a 3-bit input life cycle 000 Failure Analysis 001 Reserved 010 OEM Production 011 Customer Delivery 100 Reserved 101 Reserved

Table 1-34. PASS_LCSTAT field description

Field	Description
	110 Freescale Production 111 In Field

1.25.3 Debug Disable (DD)

Debug Disable is an optional feature that, once the life cycle is In Field overrides all other debug controls and disables the debug interface. The Debug Disable feature has a higher priority than Censorship, JTAG password and the PASS LOCK3[DBL] bit.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, C-5, CodeTest, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and µVision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

©
2017
NXP B.V.

Document Number MPC5746RRMAD
Revision 3, 06/2017

MPC5746R Reference Manual Addendum

Updates to the MPC5746R Reference Manual, Revision 6

Document Number: MPC5746RRMAD
Rev. 2, 10/2016

Contents

Section number **Title** **Page**

Chapter 1
Addendum for Rev 6 of MPC5746R

1.1 Overview....13

1.2 Device Configuration Format (DCF) Records....19

1.2.1 DCF clients....19

1.2.2 DCF_UTEST_Miscellaneous DCF client....19

1.2.3 DCF_PMC_REE_TD DCF client....21

1.2.4 DCF_IVPR0 DCF client....21

1.2.5 DCF_IVPR1 DCF client....22

1.2.6 DCF_SELFTEST_CONFIG_IPS_1 DCF client....23

1.2.7 DCF_SELFTEST_CONFIG_IPS_2 DCF client....24

1.2.8 DCF_SELFTEST_CONFIG_IPS_3 DCF client....26

1.2.9 DCF_SELFTEST_CONFIG_IPS_4 DCF client....27

1.2.10 DCF_ERROR0_PAD_SELECT DCF client....27

1.2.11 DCF_ERROR1_PAD_SELECT DCF client....29

1.3 Local memory features....30

1.4 INTC interrupt sources....30

1.5 Lockstep....43

1.5.1 Initialization....44

1.5.2 Errors....44

1.6 LINFlex clocking....45

1.7 Clock configuration....45

1.8 CMU Measurement Duration Register (CMU_MDR)....47

1.9 System Clock Dividers....47

1.10 Auxiliary Clock Generation....47

1.11 eTPU chapter....48

1.12 BCTU Introduction....48

Section number	Title	Page

1.13 BCTU memory map....49

1.14 Status Register (DSPI_SR)....49

1.15 Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 1)....50

1.16 PHASE2[FUNC] Phase: flash initial configuration....50

1.17 PHASE3[FUNC] Phase: device configuration monitoring....50

1.18 Module status during reset sequence....51

1.19 Overview....52

1.20 BAF NMI event handling....53

1.21 System Status and Configuration Module (SSCM) chip-specific module information....53

1.22 ECC error monitoring....53

1.23 Power Management Controller (PMC)....54

1.24 Supply Concept....54

1.25 Chip-specific PMC information....55

1.26 PMC register values defined by DCF clients....56

1.27 Temperature Event Status register (PMC_ESR_TD)....56

1.28 LVD-HVD self test mechanism....56

1.28.1 Default mode....57

1.28.2 Software-triggered self test....57

1.28.3 Single voltage detect test....57

1.29 Core Status (MC_ME_CS)....58

1.30 Module Configuration Register (JTAGM_MCR)....58

1.31 Module Configuration Register (JTAGM_MCR)....59

1.32 Data Out Register 0/1/2/3 (JTAGM_DOR0/1/2/3)....59

1.33 Concurrent overflow register (OFLW) implementation....60

1.34 FCCU chapter....60

1.35 AUTOLOCK_VALUE for register write access and DCF_COMPLETION value for DCF write completion....60

1.36 Password and Device Security Module (PASS) chip-specific module information....61

1.37 Flash erase counter and tamper detection....61

Chapter 2

Section number	Title	Page

Enhanced Time Processing Unit (eTPU)

2.1 Enhanced Time Processing Unit (eTPU) chip-specific module information 63

2.1.1 Configuration 63

2.1.2 Device-specific information 63

2.1.3 Register reset values 64

2.1.4 Memory 64

2.1.5 Clocking 64

2.1.6 Debug 65

2.1.7 STAC bus 65

2.1.8 Input connections 65

2.1.9 Output connections 66

2.2 Introduction 66

2.2.1 Overview 67

2.2.2 Features 74

2.2.3 Modes of operation 81

2.3 Detailed signal descriptions 83

2.3.1 eTPU channel output signals 83

2.3.2 eTPU channel input signals 83

2.3.3 Time Base Clock Signal — TCRCLK 84

2.3.4 eTPU channel output disable signals 84

2.4 Memory map/register definition 85

2.5 Memory map 85

2.6 System Configuration memory map and registers 90

2.6.1 eTPU Module Configuration Register (eTPU_SCR_MCR) 92

2.6.2 eTPU Coherent Dual-Parameter Controller Register (eTPU_SCR_CDCR) 96

2.6.3 eTPU MISC Compare Register (eTPU_SCR_MISCCMPR) 98

2.6.4 eTPU SCM Off-range Data Register (eTPU_SCR_SCMOFFDATAR) 98

2.6.5 eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) 99

2.6.6 eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2) 103

Section number	Title	Page
2.7	Timebase memory map and registers	107
2.7.1	eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1)	108
2.7.2	eTPU Engine 1 Time Base 1 (TCR1) Visibility Register (eTPU_TBR_TB1R_ENG1)	112
2.7.3	eTPU Engine 1 Time Base 2 (TCR2) Visibility Register (eTPU_TBR_TB2R_ENG1)	112
2.7.4	eTPU Engine 1 STAC Configuration Register (eTPU_TBR_STAC_ENG1)	113
2.7.5	eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2)	115
2.7.6	eTPU Engine 2 Time Base 1 (TCR1) Visibility Register (eTPU_TBR_TB1R_ENG2)	119
2.7.7	eTPU Engine 2 Time Base 2 (TCR2) Visibility Register (eTPU_TBR_TB2R_ENG2)	119
2.7.8	eTPU Engine 2 STAC Configuration Register (eTPU_TBR_STAC_ENG2)	120
2.8	Engine memory map and registers	122
2.8.1	eTPU Watchdog Timer Register for Engine 1 (eTPU_ER_WDTR_ENG1)	123
2.8.2	eTPU Idle Register for Engine 1 (eTPU_ER_IDLER_ENG1)	124
2.8.3	eTPU Watchdog Timer Register for Engine 2 (eTPU_ER_WDTR_ENG2)	125
2.8.4	eTPU Idle Register for Engine 2 (eTPU_ER_IDLER_ENG2)	126
2.9	Latency Monitor memory map and registers	127
2.9.1	eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM*n*CR_ENG1)	128
2.9.2	eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM*n*MR_ENG1)	130
2.9.3	eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM*n*CR_ENG2)	131
2.9.4	eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM*n*MR_ENG2)	132
2.10	Memory Error Support memory map and registers	133
2.10.1	eTPU Memory Error Control Register (eTPU_MESR_MECR)	134
2.10.2	eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR)	137
2.10.3	eTPU Data Error Injection Data Pattern Register (eTPU_MESR_DEIDPR)	138
2.10.4	eTPU Data Error Injection Parity Pattern Register (eTPU_MESR_DEIPPR)	138
2.10.5	eTPU Data Error Report Address Register (eTPU_MESR_DERAR)	140
2.10.6	eTPU Data Error Report Data Register (eTPU_MESR_DERDR)	141
2.10.7	eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR)	142
2.10.8	eTPU Memory Error Status Register (eTPU_MESR_MESR)	144
2.10.9	eTPU Code Error Injection Address Register (eTPU_MESR_CEIAR)	147

2.10.10 eTPU Code Error Injection Data Pattern Register (eTPU_MESR_CEIDPR)..... 148
2.10.11 eTPU Code Error Injection Parity Pattern Register (eTPU_MESR_CEIPPR)..... 148
2.10.12 eTPU Code Error Report Address Register (eTPU_MESR_CERAR)..... 149
2.10.13 eTPU Code Error Report Data Register (eTPU_MESR_CERDR)..... 150
2.10.14 eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR)..... 151
2.10.15 eTPU Code Error Fix Register (eTPU_MESR_CEFR)..... 152
2.11 Global Channel memory map and registers..... 153
2.11.1 eTPU Channel Interrupt Status Register for Engine 1 (eTPU_GCR_CISR_ENG1)..... 154
2.11.2 eTPU Channel Interrupt Status Register for Engine 2 (eTPU_GCR_CISR_ENG2)..... 154
2.11.3 eTPU Channel Data Transfer Request Status Register for Engine 1 (eTPU_GCR_CDTRSR_ENG1)..... 155
2.11.4 eTPU Channel Data Transfer Request Status Register for Engine 2 (eTPU_GCR_CDTRSR_ENG2)..... 156
2.11.5 eTPU Channel Interrupt Overflow Status Register for Engine 1 (eTPU_GCR_CIOSR_ENG1)..... 156
2.11.6 eTPU Channel Interrupt Overflow Status Register for Engine 2 (eTPU_GCR_CIOSR_ENG2)..... 157
2.11.7 eTPU Channel Data Transfer Request Overflow Status Register for Engine 1 (eTPU_GCR_CDTROSR_ENG1)..... 158
2.11.8 eTPU Channel Data Transfer Request Overflow Status Register for Engine 2 (eTPU_GCR_CDTROSR_ENG2)..... 158
2.11.9 eTPU Channel Interrupt Enable Register for Engine 1 (eTPU_GCR_CIER_ENG1)..... 159
2.11.10 eTPU Channel Interrupt Enable Register for Engine 2 (eTPU_GCR_CIER_ENG2)..... 159
2.11.11 eTPU Channel Data Transfer Request Enable Register for Engine 1 (eTPU_GCR_CDTRER_ENG1)..... 160
2.11.12 eTPU Channel Data Transfer Request Enable Register for Engine 2 (eTPU_GCR_CDTRER_ENG2)..... 160
2.11.13 eTPU Watchdog Status Register for Engine 1 (eTPU_GCR_WDSR_ENG1)..... 161
2.11.14 eTPU Watchdog Status Register for Engine 2 (eTPU_GCR_WDSR_ENG2)..... 162
2.11.15 eTPU Channel Pending Service Status Register for Engine 1 (eTPU_GCR_CPSSR_ENG1)..... 162
2.11.16 eTPU Channel Pending Service Status Register for Engine 2 (eTPU_GCR_CPSSR_ENG2)..... 163
2.11.17 eTPU Channel Service Status Register for Engine 1 (eTPU_GCR_CSSR_ENG1)..... 164
2.11.18 eTPU Channel Service Status Register for Engine 2 (eTPU_GCR_CSSR_ENG2)..... 164
2.12 Channel Configuration Control memory map and registers..... 165
2.12.1 eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1)..... 175

2.12.2 eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C*n*SCR_ENG1)....178
2.12.3 eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C*n*HSRR_ENG1)....180
2.12.4 eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2)....181
2.12.5 eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C*n*SCR_ENG2)....184
2.12.6 eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C*n*HSRR_ENG2)....186
2.13 Shared Subsystem Integration memory map and registers....187
2.13.1 eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR*n*)....188
2.14 Functional description....188
2.14.1 Functions and threads....189
2.14.2 Host interface....204
2.14.3 Scheduler....213
2.14.4 Parameter sharing and coherency....221
2.14.5 Enhanced Channels....227
2.14.6 Time bases....279
2.14.7 EAC - eTPU Angle Counter....288
2.14.8 Microengine....311
2.14.9 Microinstruction Set....330
2.14.10 Shared Subsystems support....364
2.14.11 Development support features....367
2.14.12 Safety features....375
2.14.13 Performance monitoring features....384
2.15 Initialization/application information....387
2.15.1 Configuration sequence....387
2.15.2 Reset options....389
2.15.3 Multiple parameter coherency methods....389
2.15.4 Programming Hints and Caveats....391
2.15.5 Estimating Worst Case Latency....392
2.15.6 Endianness....407
2.15.7 Memory Error Service Procedures....408

2.16 Appendices....411
2.16.1 Microcycle and I/O timing....411
2.16.2 Channel Mode Summary....416
2.16.3 MISC Algorithm....419

Chapter 3
Fault Collection and Control Unit (FCCU)

3.1 FCCU chip-specific module information....421
3.1.1 FCCU fault recovery after NMI reaction....421
3.1.2 FCCU chip-specific register reset values....421
3.1.3 FCCU_CFG register event bit values by source (N and C)....422
3.1.4 Available FCCU output signals....423
3.1.5 Error signal flow diagram....423
3.1.6 FCCU failure inputs from other modules....425
3.1.7 Lockstep Status....433
3.1.8 FOSU Count Value....434
3.1.9 FCCU_CFG[FCCU_SET_AFTER_RESET]....434
3.2 Introduction....434
3.2.1 Acronyms and abbreviations....435
3.3 Main features....436
3.4 Block diagram....437
3.5 Signal description....438
3.5.1 Signals....438
3.6 Put FCCU in Configuration or Normal state....440
3.6.1 Introduction....440
3.6.2 About changing states....440
3.6.3 Put FCCU in Configuration state....440
3.6.4 Put FCCU in Normal state....441
3.7 Run operations....441
3.7.1 Introduction....441

3.7.2 About running operations....442
3.7.3 Run an operation....442
3.8 Lock and unlock the configuration....442
3.8.1 Introduction....442
3.8.2 About locking the configuration....442
3.8.3 Temporarily lock the configuration....443
3.8.4 Temporarily unlock the configuration....443
3.8.5 Permanently lock the configuration....443
3.9 Functional description....443
3.9.1 Definitions....443
3.9.2 FSM description....444
3.9.3 Self-checking capabilities....447
3.9.4 Fault priority scheme and nesting....448
3.9.5 Fault recovery....449
3.9.6 NMI/WKPU interface....451
3.9.7 STCU interface....452
3.9.8 Nonvolatile memory interface....453
3.9.9 EOUT interface....454
3.9.10 Error signal flow....461
3.10 Register descriptions....461
3.10.1 Control (FCCU_CTRL)....464
3.10.2 Control Key (FCCU_CTRLK)....466
3.10.3 Configuration (FCCU_CFG)....467
3.10.4 Noncritical Fault Configuration (FCCU_NCF_CFG*n*)....471
3.10.5 Noncritical Fault State Configuration (FCCU_NCFS_CFG*n*)....472
3.10.6 Noncritical Fault Status (FCCU_NCF_S*n*)....473
3.10.7 Noncritical Fault Key (FCCU_NCFK)....474
3.10.8 Noncritical Fault Enable (FCCU_NCF_E*n*)....475
3.10.9 Noncritical Fault Timeout Enable (FCCU_NCF_TOE*n*)....476

3.10.10 Noncritical Fault Timeout (FCCU_NCF_TO)....477
3.10.11 Configuration-State Timer Interval (FCCU_CFG_TO)....478
3.10.12 IO Control (FCCU_EINOUT)....479
3.10.13 Status (FCCU_STAT)....481
3.10.14 NA Freeze Status (FCCU_N2AF_STATUS)....483
3.10.15 AF Freeze Status (FCCU_A2FF_STATUS)....484
3.10.16 NF Freeze Status (FCCU_N2FF_STATUS)....485
3.10.17 FA Freeze Status (FCCU_F2A_STATUS)....486
3.10.18 SC Freeze Status (FCCU_SCFS)....487
3.10.19 Noncritical Fault Fake (FCCU_NCFF)....489
3.10.20 IRQ Status (FCCU_IRQ_STAT)....489
3.10.21 IRQ Enable (FCCU_IRQ_EN)....491
3.10.22 XTMR (FCCU_XTMR)....492
3.10.23 Mode Controller Status (FCCU_MCS)....493
3.10.24 Transient Configuration Lock (FCCU_TRANS_LOCK)....495
3.10.25 Permanent Configuration Lock (FCCU_PERMNT_LOCK)....496
3.10.26 Delta T (FCCU_DELTA_T)....496
3.10.27 IRQ Alarm Enable (FCCU_IRQ_ALARM_EN*n*)....497
3.10.28 NMI Enable (FCCU_NMI_EN*n*)....498
3.10.29 Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN*n*)....499
3.10.30 Configuration registers....500
3.10.31 FCCU_CFG register bit value sources (N and C) by event....501
3.11 FCCU Output Supervision Unit....502
3.12 Use cases and limitations....504
3.12.1 Configuration guidelines....504
3.12.2 Recommendations to configure FCCU....505

Chapter 1
Addendum for Rev 6 of MPC5746R

1.1 Overview

This addendum describes corrections or updates to the MPC5746R Reference Manual. The currently available version of the MPC5746R Reference Manual is Revision 6.

Navigation tip: When navigating this PDF file using the hyperlinks, pressing the ALT and "left arrow" keys simultaneously will return you to the previously viewed page.

Table 1-1. MPC5746RRM Rev6 Addendum

Location	Description
Chapter 6, Device Configuration Format (DCF) Records	
Page 134	• Added row to DCF clients table with description "Software Tamper Region Override Disable". Modified following row's DCF Address. • See DCF clients.
Page 137	• Added recommendation to program DCF_UTEST_Miscellaneous DCF client only one time. • Updated DCF_UTEST_Miscellaneous DCF client figure and field descriptions: Changed reserved fields at 15, 19, and 24:26 from RO to RW and changed corresponding descriptions. • Added note to DCF_UTEST_Miscellaneous[OSC_EN] field description. • See DCF_UTEST_Miscellaneous DCF client.
Page 140	• Updated DCF_PMC_REE_TD DCF client figure and field descriptions: Changed reserved field at 26:27 from RO to RW and changed corresponding description. • See DCF_PMC_REE_TD DCF client.
Page 141	• In the DCF_IVPR0 DCF client description, added two notes. • Changed the position of the Vector Base field in the diagram and changed the corresponding offsets in the field description table. • See DCF_IVPR0 DCF client.
Page 141	• In the DCF_IVPR1 DCF client description, replaced the first note and added a third note regarding IVPR1_CPU1 and BAF. • Changed the position of the Vector Base field in the diagram and changed the corresponding offsets in the field description table. • See DCF_IVPR1 DCF client.

Table continues on the next page...

Table 1-1. MPC5746RRM Rev6 Addendum (continued)

Location	Description
Page 142	• In description for DCF_SELFTEST_CONFIG_IPS_1 DCF client, added statements about how the register is reset and when it is loaded by the SSCM. • Updated figure and field descriptions: Changed reserved fields at 1:9, 12:14, 15, 16:18, and 20:31 from RO to RW and changed corresponding descriptions. • Moved the "Reserved" bit that is to be set when running LBIST from the bit 0 position to the bit 31 position. The bit 0 position is now "Reserved" and is read/write zero always. • See DCF_SELFTEST_CONFIG_IPS_1 DCF client.

Page 143	• Updated DCF_SELFTEST_CONFIG_IPS_2 DCF client figure and descriptions:
	  • In description, added statements about how the register is reset and when it is loaded by the SSCM.
	  • Changed reserved fields at 0:1 and 17:19 from RO to RW and changed corresponding descriptions.
	  • Changed the order of the System_MUX_Divider_*n*_Division_Value fields in the diagram and field description table.
	  • Changed width of Aux_MUX_0_Divider_4_Division_Value and Aux_MUX_0_Divider_3_Division_Value fields to three bits.
	• See DCF_SELFTEST_CONFIG_IPS_2 DCF client.

Page 144	• In description for DCF_SELFTEST_CONFIG_IPS_3 DCF client, added statements about how the register is reset and when it is loaded by the SSCM. • Updated figure and field descriptions: Changed reserved fields at 0:4 and 14:16 from RO to RW and changed corresponding descriptions. • See DCF_SELFTEST_CONFIG_IPS_3 DCF client.
Page 145	• In description for DCF_SELFTEST_CONFIG_IPS_4 DCF client, added statements about how the register is reset and when it is loaded by the SSCM. • Updated figure and field descriptions: Changed reserved field at 0:7 from RO to RW and changed corresponding description. • See DCF_SELFTEST_CONFIG_IPS_4 DCF client.
Page 146	• Updated DCF_ERROR0_PAD_SELECT DCF client figure and field descriptions: Changed reserved fields at 0:1, 4:5, 9, and 16:27 from RO to RW and changed corresponding descriptions. • See DCF_ERROR0_PAD_SELECT DCF client.
Page 147	• Updated DCF_ERROR1_PAD_SELECT DCF client figure and field descriptions: Changed reserved fields at 0:1, 4:5, 9, and 16:27 from RO to RW and changed corresponding descriptions. • See DCF_ERROR1_PAD_SELECT DCF client.
Chapter 7, e200z4d Core Complex Overview	
Page 155	• Added the following note: "IMEM and DMEM must be initialized, regardless of whether they are used." • See Local memory features
Chapter 17, Interrupt Controller (INTC)	

Page 397	• In the "Interrupt sources" table, made the following changes to align to the terms used in the eTPU chapter.
	  • Changed "eTPU Engine 0 and 1" to "eTPU Engine 1 and 2".
	  • Changed "ETPU_MCRAB[MGEA,MGEB,ILFA,ILFB,SCMMISF]" to "eTPU_SCR_MCR[MGE1,MGE2,ILF1,ILF2,SCMMISF]".
	  • Changed "eTPU Engine 0" to "eTPU Engine 1".

Table continues on the next page...

Table 1-1. MPC5746RRM Rev6 Addendum (continued)

Location	Description
	• Changed "eTPU Engine 1" to "eTPU Engine 2". • Changed ETPU_CISR_0 to eTPU_GCR_CISR_ENG1. • Changed ETPU_CISR_1 to eTPU_GCR_CISR_ENG2. • See INTC interrupt sources.
Chapter 18, Enhanced Direct Memory Access (eDMA)	
Page 605	• Added Lockstep section. • See Lockstep.
Chapter 22, Clocking	
Page 659	• Removed both notes at end of "LINFlex clocking" section. • See LINFlex clocking.
Chapter 23, Dual PLL Digital Interface (PLLDIG)	

Page 665	• Updated the clock configuration equations:
	  • The equation "PLL1 PHI Output Frequency" is now two equations, depending upon the value of PLLDIG_PLL1FD[FDEN].
	  • The equation "PLL1 VCO Frequency" is now two equations, depending upon the value of PLLDIG_PLL1FD[FDEN].
	• See Clock configuration.

Chapter 24, Clock Monitor Unit (CMU)	
Page 695	• Added a note in the CMU_MDR register field description for the MD field. • See CMU Measurement Duration Register (CMU_MDR).
Chapter 25, Clock Generation Module (MC_CGM)	
Page 751	• Added note about system clock divider configuation. • See System Clock Dividers.
Page 758	• Added procedures to avoid undefined system behavior to end of section. • See Auxiliary Clock Generation.
Chapter 43, Enhanced Time Processing Unit (eTPU)	
Page 1319	• Numerous updates have been made throughout this chapter, which is included in its entirety. See eTPU chapter. • See Enhanced Time Processing Unit (eTPU) changes for the list of changes in the eTPU chapter.
Chapter 46, Body Cross-triggering Unit (BCTU)	
Page 1781	• In the "Introduction" section, changed "The BCTU has 32 trigger inputs" to "Up to 32 trigger inputs are available." • See BCTU Introduction.
Page 1787	• In the BCTU "Memory Map" section, deleted the BCTU_SFTRGR3 register. • See BCTU memory map.
Chapter 48, De-serial Serial Peripheral Interface (DSPI)	
Page 1991	• In the Status Register (DSPI_SR), changed TXRXS bitfield access from W1C to RO. • See Status Register (DSPI_SR)
Page 2043	• In the "Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 1)" section, added a note regarding operation in MTFE Enabled Mode (configuration of frame size and masking of extra bits during POP operation). • See Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 1)
Chapter 55, Reset and Boot	

Table continues on the next page...

Table 1-1. MPC5746RRM Rev6 Addendum (continued)

Location	Description
Page 2481	• Changed "All processes that need to be done in PHASE2[DEST] are completed" to "All processes that need to be done in PHASE2[FUNC] are completed". • See *PHASE2[FUNC]* Phase: flash initial configuration.
Page 2481	• Changed both occurrences of "shortened external 'functional' reset event" to "shortened 'functional' reset event". • See *PHASE3[FUNC]* Phase: device configuration monitoring.
Page 2483	• In the DEVICE CONFIG row, changed RESET to ON in the P1 and P2 [DEST] columns and changed HOLD to LOAD in the P3[FUNC] column. • See Module status during reset sequence.
Page 2483	• Added footnote (6) about ADC reset to the "other modules including cores" table entry. • Corrected footnote numbering throughout table. • See Module status during reset sequence.
Chapter 56, Reset Generation Module (MC_RGM)	
Page 2486	• Edited the two lists of reset sources on left side of "MC_RGM Block Diagram". • See Figure 1-3.
Chapter 57, Boot Assist Flash (BAF)	
Page 2555	• The section [BAF NMI event handling] has been moved and its content completely replaced/corrected. The previous revision incorrectly described the workaround for an issue in which the device may latch pending FCCU NMI events generated during a device reset. • See BAF NMI event handling.
Chapter 58, System Status and Configuration Module (SSCM)	
Page 2576	• Added a new paragraph regarding discernment between the boot location and other valid RCHW locations. • See System Status and Configuration Module (SSCM) chip-specific module information.
Page 2584	• Added a Note regarding the case of a multibit ECC error on the DCF start record during DCF Scanning. • See ECC error monitoring.
Chapter 59, Power Management	
Page 2590	• Changed "within the PMU" to "within the PMC". • See Power Management Controller (PMC).
Page 2591	• Added supplies VSS_HV_ADV_SAR, VSS_HV_ADV_SD, VDD_HV_IO_JTAG, VDD_HV_IO_FEC, VDD_HV_IO_MSC, VDD_LV_BD, VDD_HV_IO_BD, and VDDSTBY to the "Supply concept" diagram. • See Figure 1-4.
Chapter 60, Power Management Controller (PMC)	
Page 2601	• Changed title of chapter from "Power Management Controller digital interface (PMC_dig)" to "Power Management Controller (PMC)".
Page 2601	• Changed the title of Section 60.1 from "PMC_dig chip-specific module information" to "Chip-specific PMC information". • Changed "The Power Management Unit (PMU)" to "The PMC". • Changed "within the PMU" to "within the PMC". • See Chip-specific PMC information.

Table continues on the next page...

Table 1-1. MPC5746RRM Rev6 Addendum (continued)

Location	Description

Page 2601	• In the "PMC_dig chip-specific module information" section:
	  • In the "PMC register values defined by DCF clients" subsection:
	  • Renamed this subsection (was "PMC registers whose reset values can be defined by DCF clients").
	  • Change the existing sentence to "The following PMC registers are defined by the corresponding DCF clients:".
	  • Added the sentence "Refer to Table 6-5, DCF Clients".
	  • Added a note to verify the PMC_REE register value against the DCF_PMC_REE_CTRL DCF client value after any reset, and to update it if the verification fails.
	  • Removed the "PMC registers with chip-specific reset values" subsection.
	• See PMC register values defined by DCF clients.

Pages 2607-2608	• Added a note in the PMC_ESR_TD register field descriptions for TEMP1_2 and TEMP0_2. • See Temperature Event Status register (PMC_ESR_TD).

Page 2632	• Throughout the "LVD-HVD self test mechanism" section/subsections:
	  • Changed all instances of "VD_UTST[field_name]" to "PMC_SELF_TEST_UM_VD_REG[field_name]".
	  • Changed instance of "VD_ST_CTRL[ST_MODE]" to "PMC_SELF_TEST_UM_VD_REG[ST_MODE]".
	• See LVD-HVD self test mechanism.

Chapter 62, Mode Entry Module (MC_ME)	
Page 2724	• Updated the field names in the "MC_ME_CS field descriptions" table to match the implementation. • See Core Status (MC_ME_CS)
Chapter 70, JTAG Master (JTAGM)	
Page 3057	• In "JTAGM_MCR" text, added a note regarding an unexpected interrupt that will occur if the Idle Interrupt is enabled after the end of frame transfer. • See Module Configuration Register (JTAGM_MCR)
Page 3058	• In "JTAGM_MCR field descriptions", changed the TCKSEL 010 field value to Reserved. • See Module Configuration Register (JTAGM_MCR)
Pages 3061-3062	• In the "Data Out Register 0/1/2/3 (JTAGM_DOR0/1/2/3)" sections, added the stipulation that writing to these registers when DTM = 0 may cause unintended results. • See Data Out Register 0/1/2/3 (JTAGM_DOR0/1/2/3)
Chapter 77, Memory Error Management Unit (MEMU)	

Page 3273	• In the "Corresponding error sources for system RAM OFLW0 register" table:
	  • Added the following note to the "Core0 Instruction TCM error" row: "This error source could be triggered if Core0 IMEM is not initialized before Core0 is started."
	  • Added the following note to the "Core1 Instruction TCM error" row: "This error source could be triggered after reset."
	• See Table 1-15

Chapter 78, Fault Collection and Control Unit (FCCU)	

Table continues on the next page...

Table 1-1. MPC5746RRM Rev6 Addendum (continued)

Location	Description
Page 3305	• Numerous updates have been made throughout the chapter, which is included in its entirety. See FCCU chapter. • See Fault Collection and Control Unit (FCCU) changes for the list of changes in the FCCU chapter.
Chapter 79, Self-Test Control Unit (STCU2)	

Page 3392	• Updated the "AUTOLOCK_VALUE for register write access via STCU2_SKC" section:
	  • Renamed this section.
	  • Replaced this section with new content.
	  • Added DCF_COMPLETION value of 16.
	  • Updated AUTOLOCK_VALUE to 1024.
	• See AUTOLOCK_VALUE for register write access and DCF_COMPLETION value for DCF write completion.

Chapter 82, Password and Device Security Module (PASS)	
Page 3520	• Added the "JTAG_PASSWORD register bit correspondence" table and its introductory sentence. • See Password and Device Security Module (PASS) chip-specific module information.
Chapter 83, Tamper Detection Module (TDM)	
Page 3560	• Added paragraph "When attempting to update the TDM diary and erase flash blocks..." • See Flash erase counter and tamper detection.
Attachment: Flash_Block_Assignment.xlsx	
"LOCK Register Def." tab	• In the 'LOCK0 Register' table, changed the "Register Bit Name" of bit number 1 from "Reserved" to "ATSL".
"LOCK Register Def." tab	• In the 'LOCK1 Register' table, updated cells Y24-AE24 and Y25-AE25.
Attachment: IO_Signal_Description_and_Input_Multiplexing_Tables.xlsx	
"IO Signal Table" tab, cell T1500	• Changed the Pad State During Reset of PJ[14] from "Output/Low" to "Input/Pull-up".
Attachment: Memory_Map_Tables.xlsx	
"NVM and overlay memory map" tab	• In the "Large Flash Blocks - no overlay" section, updated the rows with start addresses 0x010C0000 - 0x01340000.
"UTEST memory map" tab	• Corrected the UTEST NVM addresses for Temperature Sensor 2 Calibration coefficients K1-K4, locations 0x004000C4 - 0x004000CF.

1.2 Device Configuration Format (DCF) Records

1.2.1 DCF clients

Table 1-2. DCF Clients

DCF CS[14:0]	DCF Address [16:2]	DCF Client Description	Reset Value of DCF Client	IPS Read Direct Access	DCF Client Special Strategy	Flash Special Strategy	DCF Order Dependency	UTEST Flash Memory Space
Tamper Detect								
000_0000_0001_0000	0000000_0000000	Diary Base Address	0xFFFFFFFF	no	Write Once + Write ‘0’ only	None	None	8
000_0000_0001_0000	0000000_0000001	Tamper Region Override	0x00	no	Write ‘1’ only	None	None	8
000_0000_0001_0000	0000000_0000010	Software Tamper Region Override Disable	0x00000000	no	Write Once + Write '1' only	None	None	8
000_0000_0001_0000	0000000_0000011-0000111	Reserved						

1.2.2 DCF_UTEST_Miscellaneous DCF client

The DCF_UTEST_Miscellaneous register controls miscellaneous functionalities.

The DCF_UTEST_Miscellaneous DCF client represents the chip-specific device configuration. Therefore, we recommend that you program this client only one time, to avoid undesirable chip behavior.

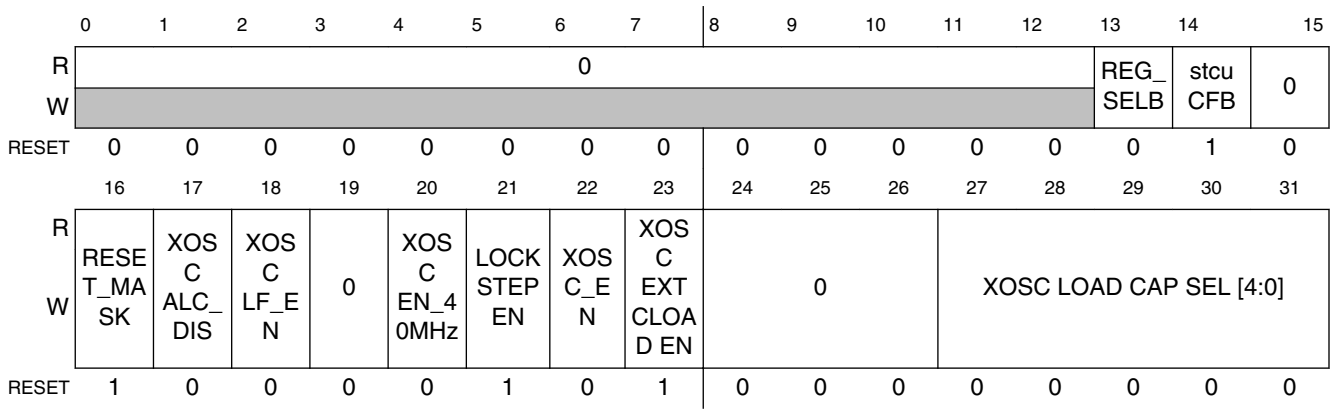


Table 1-3. DCF_UTEST_Miscellaneous Field Description

Field	Name	Description
0-12	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13	REG_SELB	If this bit is set to 0, internal regulation will be selected. If not external regulation is selected.
14	stcu CFB	If this is set to 1, critical fault during offline LBIST run will generate a destructive reset.
15	Reserved	This field is reserved. This read-write field must always be written with the value 0.
16	RESET_MASK	Active Low reset Mask bit. 1- RESET PIN enabled, 0-RESET Pin Disabled Level Controller
17	XOSC ALC_DIS	Active high to disable Automatic Level Controller
18	XOSC LF_EN	Active high when oscillator is used for low frequency (~8MHz)
19	Reserved	This field is reserved. This read-write field must always be written with the value 0.
20	XOSC EN_40MHz	Selects 20MHz against 40MHz XOSC. Set this bit to enable the 40Mhz mode.
21	LOCKSTEP EN	Enables Safety Core to be in lockstep with Checker Core. Value 0 = Lockstep disabled, Value 1 = Lockstep enabled
22	XOSC_EN	Enables XOSC. **NOTE:** If XOSC is enabled via this field, then VDD_HV_IO_JTAG must be within the operating range before the RESET pin is released.
23	XOSC EXT CLOAD EN	Selects XOSC internal capacitor (ok for 20-40MHz operation) or external (required for 4-16MHz operation). Set to 1 if external capacitors are used.
24-26	Reserved	This field is reserved. This read-write field must always be written with the value 0.
27-31	XOSC LOAD CAP SEL [4:0]	5 Trim bits to program the load capacitance at crystal pins

1.2.3 DCF_PMC_REE_TD DCF client

The DCF_PMC_REE_TD register controls whether the temperature detect signal event causes a reset.

If the desired flag bit is enabled, the temperature detect event causes a reset to be generated when the selected temperature passes the trigger event.

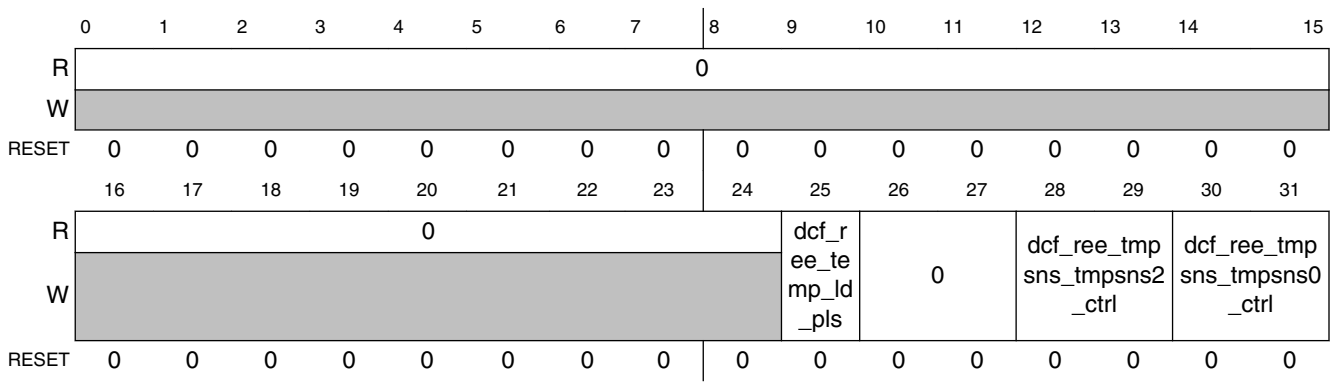


Table 1-4. DCF_PMC_REE_TD Field Description

Field	Name	Description
0-24	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25	dcf_ree_temp_ld_pls	Set this bit to 1 so that values uploaded becomes effective.
26-27	Reserved	This field is reserved. This read-write field must always be written with the value 0.
28-29	dcf_ree_tmpsns_tmpsns2_ctrl	Defines whether a temperature detect assertion generates a system reset. This is associated with the temperature for the temperature sensor point (150C). • 00: Temperature sensor detect event does not cause system reset. • 01: Temperature sensor 0 detect causes system reset. • 10: Temperature sensor 1 detect causes system reset. • 11: Temperature sensor 0 and 1 detect causes system reset.
30-31	dcf_ree_tmpsns_tmpsns0_ctrl	Defines whether a temperature detect assertion generates a system reset. This is associated with the temperature for the cold temperature sensor,when the temperature exceeds its corresponding threshold. • 00: Temperature sensor detect event does not cause system reset. • 01: Temperature sensor 0 detect causes system reset. • 10: Temperature sensor 1 detect causes system reset. • 11: Temperature sensor 0 and 1 detect causes system reset.

1.2.4 DCF_IVPR0 DCF client

The DCF_IVPR0 is the reset interrupt vector prefix register DCF client for Main Core_0.

NOTE

- The DCF client value is applied as the core IVPR reset value if the core is reset and PFLASH_PFCR3[BAF_DIS] is written with 1.
- The value in this DCF client is left-shifted by 8 bits before programmed to the Core_1 IVPR.

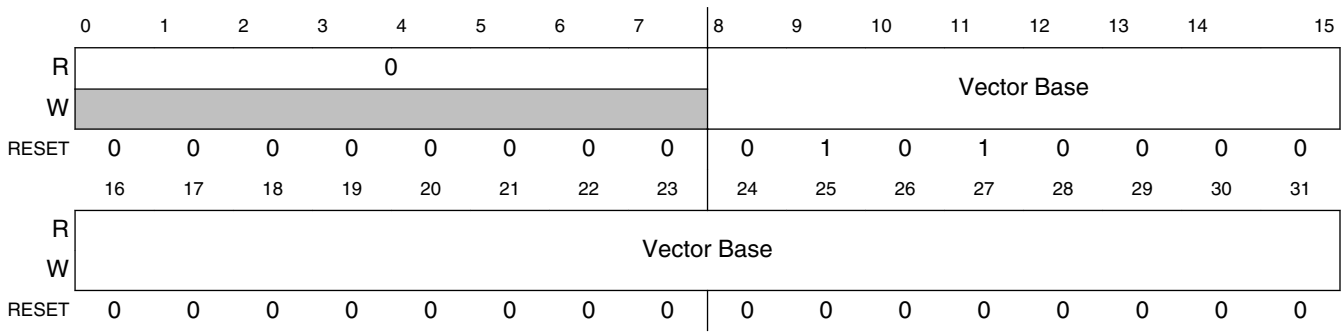


Table 1-5. DCF_IVPR0 Field Description

Field	Name	Description
0-7	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8-31	Vector Base	This field is used to define the base location of the vector table, aligned to a 256 byte boundary. This field provides the high-order 24 bits of the location of all interrupt handlers (unless hardware vectoring is used). The vector offset value appropriate for the type of exception being processed is concatenated with the IVPR Vector Base to form the address of the handler in memory. For hardware-vectored interrupts, the value of the supplied interrupt vector is logically OR'ed with the low order bits of the Vec Base Field to determine the interrupt handler address. **NOTE:** dcf_ivpr[31:8] -> core.ivpr[8:31]

1.2.5 DCF_IVPR1 DCF client

The DCF_IVPR1 is the reset interrupt vector prefix register DCF client for Main Core_1.

NOTE

- The DCF client value is applied as the core IVPR reset value if the core is reset and PFLASH_PFCR3[BAF_DIS] is written with 1.

- The value in this DCF client is left-shifted by 8 bits before programmed to the Core_1 IVPR.
- IVPR1_CPU1 - BAF set IVPR1 for CPU1 DCF Client can be used to apply an IVPR reset value for core1 after POR.

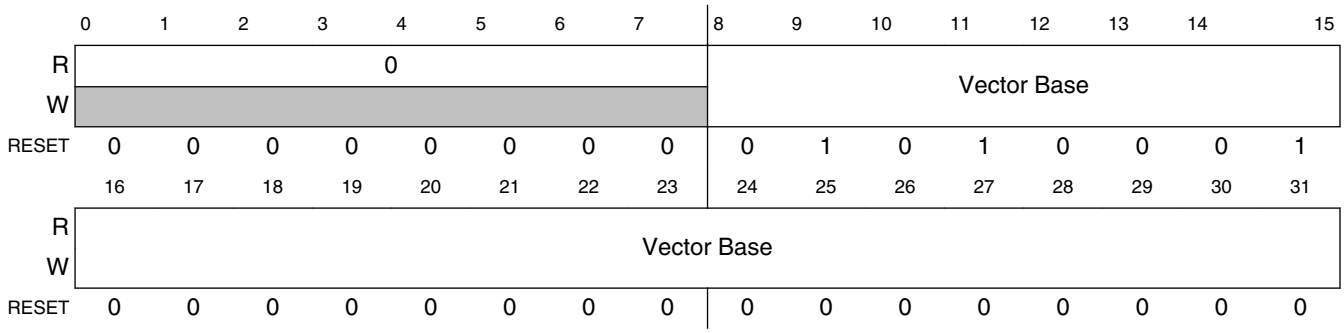


Table 1-6. DCF_IVPR1 Field Description

Field	Name	Description
0-7	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8-31	Vector Base	This field is used to define the base location of the vector table, aligned to a 256 byte boundary. This field provides the high-order 24 bits of the location of all interrupt handlers (unless hardware vectoring is used). The vector offset value appropriate for the type of exception being processed is concatenated with the IVPR Vector Base to form the address of the handler in memory. For hardware-vectored interrupts, the value of the supplied interrupt vector is logically OR'ed with the low order bits of the Vec Base Field to determine the interrupt handler address. **NOTE:** dcf_ivpr[31:8] -> core.ivpr[8:31]

1.2.6 DCF_SELFTEST_CONFIG_IPS_1 DCF client

The DCF_SELFTEST_CONFIG_IPS_1 register controls the fault Model for LBIST and selects which PLL would be used for LBIST testing if any.

This register is reset by destructive reset only. The DCF client is loaded by the SSCM only if the DCF client has been previously reset.

Address: 0x11_1111_1110_0000 base + 0x0 offset

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0										PLL_LOCK1	PLL_LOCK0	0			1
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			config[1]	0											0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1-7. DCF_SELFTEST_CONFIG_IPS_1 Field Description

Field	Name	Description
0-9	Reserved	This field is reserved. This read-write field must always be written with the value 0.
10	PLL_LOCK1	Program this bit to "1" if PLL1 is to be used for Selftest. Only one of bits 20 and 21 should be programmed "1".
11	PLL_LOCK0	Program this bit to "1" if PLL0 is to be used for Selftest . Only one of bits 20 and 21 should be programmed "1".
12-14	Reserved	This field is reserved. This read-write field must always be written with the value 0.
15	Reserved	This field is reserved. This read-write field must always be written with the value 1.
16-18	Reserved	This field is reserved. This read-write field must always be written with the value 0.
19	config[1]	Program this bit to "1" if stuck-At LBIST is required to be run. In this mode PLL is locked at 200 Mhz but all partitions are run at 50 Mhz at 4 seperate edges of 200 Mhz so as to reduce the peak power.
20-30	Reserved	This field is reserved. This read-write field must always be written with the value 0.
31	Reserved	This field is reserved. Set this bit when running LBIST.

1.2.7 DCF_SELFTEST_CONFIG_IPS_2 DCF client

The DCF_SELFTEST_CONFIG_IPS_2 register controls the division values for the dividers.

The division value set by these fields is the bit value + 1. (0b000 sets the division value to 1, 0b001 sets the division value to 2, and so on.)

This register is reset by destructive reset only. The DCF client is loaded by the SSCM only if the DCF client has been previously reset.

Address: 0x11_1111_1110_0000 base + 0x100 offset

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	0		Aux_MUX_0_Divider_4_Division_Value			Aux_MUX_0_Divider_3_Division_Value			Aux_MUX_0_Divider_2_Division_Value			Aux_MUX_0_Divider_1_Division_Value			Aux_MUX_0_Divider_0_Division_Value	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	(Aux_MUX_0_Divider_0_Division_Value)		0			System_MUX_Divider_0_Division_Value			System_MUX_Divider_1_Division_Value			System_MUX_Divider_2_Division_Value			System_MUX_Divider_3_Division_Value	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1-8. DCF_SELFTEST_CONFIG_IPS_2 Field Description

Field	Name	Description
0-1	Reserved	This field is reserved. This read-write field must always be written with the value 0.
2-4	Aux_MUX_0_Divider_4_Division_Value	Sets the division value of the divider.
5-7	Aux_MUX_0_Divider_3_Division_Value	Sets the division value of the divider.
8-10	Aux_MUX_0_Divider_2_Division_Value	Sets the division value of the divider.
11-13	Aux_MUX_0_Divider_1_Division_Value	Sets the division value of the divider.
14-16	Aux_MUX_0_Divider_0_Division_Value	Sets the division value of the divider.
17-19	Reserved	This field is reserved. This read-write field must always be written with the value 0.
20-22	System_MUX_Divider_0_Division_Value	Sets the division value of the divider.
23-25	System_MUX_Divider_1_Division_Value	Sets the division value of the divider.
26-28	System_MUX_Divider_2_Division_Value	Sets the division value of the divider.
29-31	System_MUX_Divider_3_Division_Value	Sets the division value of the divider.

1.2.8 DCF_SELFTEST_CONFIG_IPS_3 DCF client

The DCF_SELFTEST_CONFIG_IPS_3 register controls the division values for the dividers.

The division value set by these fields is the bit value + 1. (0b000 sets the division value to 1, 0b001 sets the division value to 2, and so on.)

This register is reset by destructive reset only. The DCF client is loaded by the SSCM only if the DCF client has been previously reset.

Address: 0x11_1111_1110_0000 base + 0x1000 offset

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	0					Aux_Mux_10_Divider_0_Division_Value			Aux_Mux_9_Divider_0_Division_Value			Aux_Mux_8_Divider_0_Division_Value			0	
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	0	Aux_Mux_6_Divider_0_Division_Value			Aux_Mux_5_Divider_1_Division_Value			Aux_Mux_5_Divider_0_Division_Value			Aux_Mux_2_Divider_0_Division_Value			Aux_Mux_1_Divider_0_Division_Value		
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1-9. DCF_SELFTEST_CONFIG_IPS_3 Field Description

Field	Name	Description
0-4	Reserved	This field is reserved. This read-write field must always be written with the value 0.
5-7	Aux_Mux_10_Divider_0_Division_Value	Sets the division value of the divider.
8-10	Aux_Mux_9_Divider_0_Division_Value	Sets the division value of the divider.
11-13	Aux_Mux_8_Divider_0_Division_Value	Sets the division value of the divider.
14-16	Reserved	This field is reserved. This read-write field must always be written with the value 0.
17-19	Aux_Mux_6_Divider_0_Division_Value	Sets the division value of the divider.
20-22	Aux_Mux_5_Divider_1_Division_Value	Sets the division value of the divider.
23-25	Aux_Mux_5_Divider_0_Division_Value	Sets the division value of the divider.
26-28	Aux_Mux_2_Divider_0_Division_Value	Sets the division value of the divider.
29-31	Aux_Mux_1_Divider_0_Division_Value	Sets the division value of the divider.

1.2.9 DCF_SELFTEST_CONFIG_IPS_4 DCF client

The DCF_SELFTEST_CONFIG_IPS_4 register controls the LBIST shift length for each partition.

This register is reset by destructive reset only. The DCF client is loaded by the SSCM only if the DCF client has been previously reset.

Address: 0x11_1111_1110_0000 base + 0x1100 offset

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	0								LBIST_W1_Shift_Length						LBIST_W0_Shift_Length	
RESET	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	(LBIST_W0_Shift_Length)				LBIST_L1_Shift_Length						LBIST_L0_Shift_Length					
RESET	0	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1

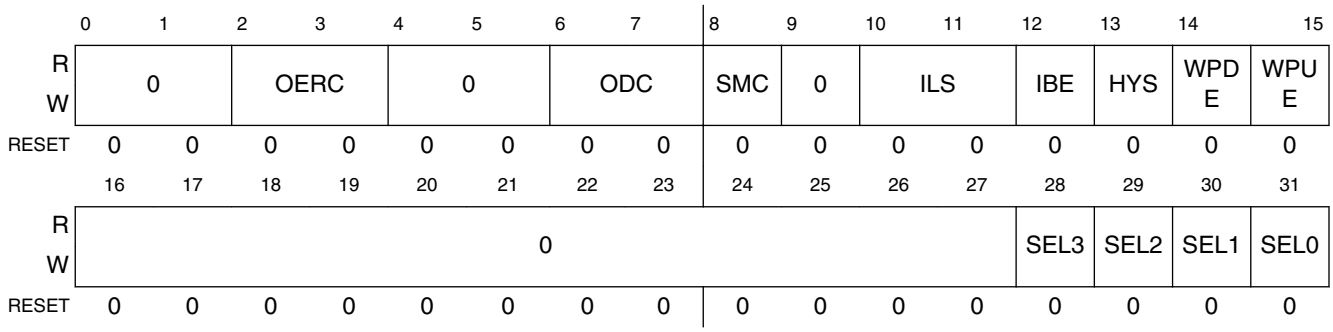
Table 1-10. DCF_SELFTEST_CONFIG_IPS_4 Field Description

Field	Name	Description
0-7	Reserved	This field is reserved. This read-write field must always be written with the value 0.
8-13	LBIST_W1_Shift_Length	Sets the LBIST shift length of the W1 partition.
14-19	LBIST_W0_Shift_Length	Sets the LBIST shift length of the W0 partition.
20-25	LBIST_L1_Shift_Length	Sets the LBIST shift length of the L1 partition.
26-31	LBIST_L0_Shift_Length	Sets the LBIST shift length of the L0 partition.

1.2.10 DCF_ERROR0_PAD_SELECT DCF client

The DCF_ERROR0_PAD_SELECT_CONTROL register selects either the functional muxing or the EOUT functionality on pads PJ[11], PJ[1], PI[0], and PH[0]. This is done at boot time according to the SELn bits of this DCF client:

- If ERROR0 OUT is selected (i.e., the SELn bits are set), the PAD controls of the pad are governed by this DCF client.
- If functional muxing is selected (i.e., the SELn bits are not set), the PAD controls are governed by the SIUL2_MSCRn register.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	0		OERC		0		ODC		SMC	0	ILS		IBE	HYS	WPDE	WPUE
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	0												SEL3	SEL2	SEL1	SEL0
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1-11. DCF_ERROR0_PAD_SELECT_CONTROL Field Descriptions

Field	Name	Description
0-1	Reserved	This field is reserved. This read-write field must always be written with the value 0.
2-3	OERC	Sets the Output Edge Rate Control when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
4-5	Reserved	This field is reserved. This read-write field must always be written with the value 0.
6-7	ODC	Sets the Output Drive Control when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
8	SMC	Sets the Safe Mode Control when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
9	Reserved	This field is reserved. This read-write field must always be written with the value 0.
10-11	ILS	Sets the Input Level Selection when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
12	IBE	Sets the Input Buffer Enable when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
13	HYS	Sets the Input Hysteresis when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
14	WPDE	Sets the Weak Pulldown Enable when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
15	WPUE	Sets the Weak Pullup Enable when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
16-27	Reserved	This field is reserved. This read-write field must always be written with the value 0.
28	SEL3	Enable ERROR0 OUT on PJ[11] irrespective of SIUL2_MSCRn[SSS] settings and override the pad setting as defined in this register.
29	SEL2	Enable ERROR0 OUT on PJ[1] irrespective of SIUL2_MSCRn[SSS] settings and override the pad setting as defined in this register
30	SEL1	Enable ERROR0 OUT on PI[0] irrespective of SIUL2_MSCRn[SSS] settings and override the pad setting as defined in this register
31	SEL0	Enable ERROR0 OUT on PH[0] irrespective of SIUL2_MSCRn[SSS] settings and override the pad setting as defined in this register

There is no hardware restriction on the programming of the SELn bits. All the four pads can be simultaneously be programmed as ERROR0 OUT Pins.

1.2.11 DCF_ERROR1_PAD_SELECT DCF client

The DCF_ERROR1_PAD_SELECT_CONTROL register selects either the functional muxing or the EOUT functionality on pads PJ[0], PI[1], and PH[3]. This is done at boot time according to the SELn bits of this DCF client:

- If ERROR1 OUT is selected (i.e., the SELn bits are set), the PAD controls of the pad are governed by this DCF client.
- If functional muxing is selected (i.e., the SELn bits are not set), the PAD controls are governed by the SIUL2_MSCR register.

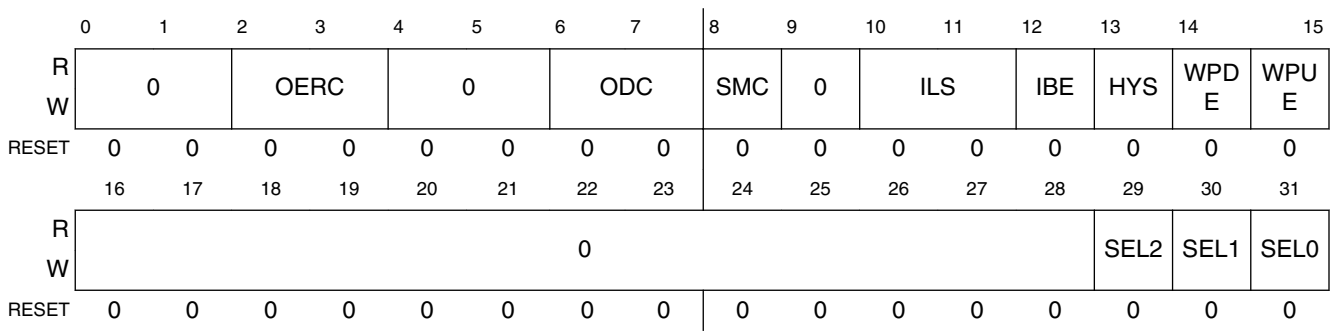


Table 1-12. DCF_ERROR1_PAD_SELECT_CONTROL Field Descriptions

Field	Name	Description
0-1	Reserved	This field is reserved. This read-write field must always be written with the value 0.
2-3	OERC	Sets the Output Edge Rate Control when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
4-5	Reserved	This field is reserved. This read-write field must always be written with the value 0.
6-7	ODC	Sets the Output Drive Control when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
8	SMC	Sets the Safe Mode Control when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
9	Reserved	This field is reserved. This read-write field must always be written with the value 0.
10-11	ILS	Sets the Input Level Selection when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
12	IBE	Sets the Input Buffer Enable when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
13	HYS	Sets the Input Hysteresis when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
14	WPDE	Sets the Weak Pulldown Enable when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.

Table continues on the next page...

Table 1-12. DCF_ERROR1_PAD_SELECT_CONTROL Field Descriptions (continued)

Field	Name	Description
15	WPUE	Sets the Weak Pullup Enable when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
16-28	Reserved	This field is reserved. This read-write field must always be written with the value 0.
29	SEL2	Enable ERROR1 OUT on PJ[0] irrespective of SIUL2_MSCRn[SSS] settings and override the pad setting as defined in this register.
30	SEL1	Enable ERROR1 OUT on PI[1] irrespective of SIUL2_MSCRn[SSS] settings and override the pad setting as defined in this register.
31	SEL0	Enable ERROR1 OUT on PH[3] irrespective of SIUL2_MSCRn[SSS] settings and override the pad setting as defined in this register.

There is no hardware restriction on the programming of the SELn bits. All three pads can be simultaneously be programmed as ERROR1 OUT Pins.

1.3 Local memory features

The features of the Local Instruction and Data Memories are as follows:

- Local instruction and data memories (IMEM, DMEM) with shared AHB 2.v6 64-bit slave interface
  - e200z425n3—16 KB IMEM, 32 KB DMEM
  - e200z424—No IMEM, No DMEM
- Support for zero wait-state access relative to cache hit timings
- Support for multi-bit (SECDED) ECC with correction/scrubbing capability
- 4-entry ECC R-M-W Store Buffer for DMEM
- 32-bit address bus plus attributes and control
- Separate uni-directional 64-bit read data bus and (for DMEM only) 64-bit write data bus

NOTE

IMEM and DMEM must be initialized, regardless of whether they are used.

1.4 INTC interrupt sources

The following table defines the interrupt sources for the INTC on this chip. All IRQ numbers not specifically listed are not used.

Table 1-13. Interrupt sources

IRQ #	Offset	Source description	Source name
0	1000	Software settable flag 0	INTC_SSCIR0[CLR0]
1	1004	Software settable flag 1	INTC_SSCIR0[CLR1]
2	1008	Software settable flag 2	INTC_SSCIR0[CLR2]
3	100C	Software settable flag 3	INTC_SSCIR0[CLR3]
4	1010	Software settable flag 4	INTC_SSCIR0[CLR4]
5	1014	Software settable flag 5	INTC_SSCIR0[CLR5]
6	1018	Software settable flag 6	INTC_SSCIR0[CLR6]
7	101C	Software settable flag 7	INTC_SSCIR0[CLR7]
8	1020	Software settable flag 8	INTC_SSCIR0[CLR8]
9	1024	Software settable flag 9	INTC_SSCIR0[CLR9]
10	1028	Software settable flag 10	INTC_SSCIR0[CLR10]
11	102C	Software settable flag 11	INTC_SSCIR0[CLR11]
12	1030	Software settable flag 12	INTC_SSCIR0[CLR12]
13	1034	Software settable flag 13	INTC_SSCIR0[CLR13]
14	1038	Software settable flag 14	INTC_SSCIR0[CLR14]
15	103C	Software settable flag 15	INTC_SSCIR0[CLR15]
16	1040	Software settable flag 16	INTC_SSCIR0[CLR16]
17	1044	Software settable flag 17	INTC_SSCIR0[CLR17]
18	1048	Software settable flag 18	INTC_SSCIR0[CLR18]
19	104C	Software settable flag 19	INTC_SSCIR0[CLR19]
20	1050	Software settable flag 20	INTC_SSCIR0[CLR20]
21	1054	Software settable flag 21	INTC_SSCIR0[CLR21]
22	1058	Software settable flag 22	INTC_SSCIR0[CLR22]
23	105C	Software settable flag 23	INTC_SSCIR0[CLR23]
24	1060	Software settable flag 24	INTC_SSCIR0[CLR24]
25	1064	Software settable flag 25	INTC_SSCIR0[CLR25]
26	1068	Software settable flag 26	INTC_SSCIR0[CLR26]
27	106C	Software settable flag 27	INTC_SSCIR0[CLR27]
28	1070	Software settable flag 28	INTC_SSCIR0[CLR28]
29	1074	Software settable flag 29	INTC_SSCIR0[CLR29]
30	1078	Software settable flag 30	INTC_SSCIR0[CLR30]
31	107C	Software settable flag 31	INTC_SSCIR0[CLR31]
32	1080	Platform watchdog timer0	SWT_0_IR[TIF]
33	1084	Platform watchdog timer1	SWT_1_IR[TIF]
35	108C	Platform watchdog timer3	SWT_3_IR[TIF]
36	1090	Platform periodic timer 0_0 (STM)	STM_0_CIR0[CIF]
37	1094	Platform periodic timer 0_1 (STM)	STM_0_CIR1[CIF]
38	1098	Platform periodic timer 0_2 (STM)	STM_0_CIR2[CIF]
39	109A	Platform periodic timer 0_3 (STM)	STM_0_CIR3[CIF]

Table continues on the next page...

Table 1-13. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
40	10A0	Platform periodic timer 1_0 (STM)	STM_1_CIR0[CIF]
41	10A4	Platform periodic timer 1_1 (STM)	STM_1_CIR1[CIF]
42	10A8	Platform periodic timer 1_2 (STM)	STM_1_CIR2[CIF]
43	10AC	Platform periodic timer 1_3 (STM)	STM_1_CIR3[CIF]
52	10D0	eDMA Combined Error 127 - 0	eDMA Channel Error Flags
53	10D4	eDMA Channel 0	DMA_INTL[INT0]
54	10D8	eDMA Channel 1	DMA_INTL[INT1]
55	10DC	eDMA Channel 2	DMA_INTL[INT2]
56	10E0	eDMA Channel 3	DMA_INTL[INT3]
57	10E4	eDMA Channel 4	DMA_INTL[INT4]
58	10E8	eDMA Channel 5	DMA_INTL[INT5]
59	10EC	eDMA Channel 6	DMA_INTL[INT6]
60	10F0	eDMA Channel 7	DMA_INTL[INT7]
61	10F4	eDMA Channel 8	DMA_INTL[INT8]
62	10F8	eDMA Channel 9	DMA_INTL[INT9]
63	10FC	eDMA Channel 10	DMA_INTL[INT10]
64	1100	eDMA Channel 11	DMA_INTL[INT11]
65	1104	eDMA Channel 12	DMA_INTL[INT12]
66	1108	eDMA Channel 13	DMA_INTL[INT13]
67	110C	eDMA Channel 14	DMA_INTL[INT14]
68	1110	eDMA Channel 15	DMA_INTL[INT15]
69	1114	eDMA Channel 16	DMA_INTL[INT16]
70	1118	eDMA Channel 17	DMA_INTL[INT17]
71	111C	eDMA Channel 18	DMA_INTL[INT18]
72	1120	eDMA Channel 19	DMA_INTL[INT19]
73	1124	eDMA Channel 20	DMA_INTL[INT20]
74	1128	eDMA Channel 21	DMA_INTL[INT21]
75	112C	eDMA Channel 22	DMA_INTL[INT22]
76	1130	eDMA Channel 23	DMA_INTL[INT23]
77	1134	eDMA Channel 24	DMA_INTL[INT24]
78	1138	eDMA Channel 25	DMA_INTL[INT25]
79	113C	eDMA Channel 26	DMA_INTL[INT26]
80	1140	eDMA Channel 27	DMA_INTL[INT27]
81	1144	eDMA Channel 28	DMA_INTL[INT28]
82	1148	eDMA Channel 29	DMA_INTL[INT29]
83	114C	eDMA Channel 30	DMA_INTL[INT30]
84	1150	eDMA Channel 31	DMA_INTL[INT31]
85	1154	eDMA Channel 32	DMA_INTH[INT32]
86	1158	eDMA Channel 33	DMA_INTH[INT33]

Table continues on the next page...

Table 1-13. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
87	115C	eDMA Channel 34	DMA_INTH[INT34]
88	1160	eDMA Channel 35	DMA_INTH[INT35]
89	1164	eDMA Channel 36	DMA_INTH[INT36]
90	1168	eDMA Channel 37	DMA_INTH[INT37]
91	116C	eDMA Channel 38	DMA_INTH[INT38]
92	1170	eDMA Channel 39	DMA_INTH[INT39]
93	1174	eDMA Channel 40	DMA_INTH[INT40]
94	1178	eDMA Channel 41	DMA_INTH[INT41]
95	117C	eDMA Channel 42	DMA_INTH[INT42]
96	1180	eDMA Channel 43	DMA_INTH[INT43]
97	1184	eDMA Channel 44	DMA_INTH[INT44]
98	1188	eDMA Channel 45	DMA_INTH[INT45]
99	118C	eDMA Channel 46	DMA_INTH[INT46]
100	1190	eDMA Channel 47	DMA_INTH[INT47]
101	1194	eDMA Channel 48	DMA_INTH[INT48]
102	1198	eDMA Channel 49	DMA_INTH[INT49]
103	119C	eDMA Channel 50	DMA_INTH[INT50]
104	11A0	eDMA Channel 51	DMA_INTH[INT51]
105	11A4	eDMA Channel 52	DMA_INTH[INT52]
106	11A8	eDMA Channel 53	DMA_INTH[INT53]
107	11AC	eDMA Channel 54	DMA_INTH[INT54]
108	11B0	eDMA Channel 55	DMA_INTH[INT55]
109	11B4	eDMA Channel 56	DMA_INTH[INT56]
110	11B8	eDMA Channel 57	DMA_INTH[INT57]
111	11BC	eDMA Channel 58	DMA_INTH[INT58]
112	11C0	eDMA Channel 59	DMA_INTH[INT59]
113	11C4	eDMA Channel 60	DMA_INTH[INT60]
114	11C8	eDMA Channel 61	DMA_INTH[INT61]
115	11CC	eDMA Channel 62	DMA_INTH[INT62]
116	11D0	eDMA Channel 63	DMA_INTH[INT63]
185	12E4	Flash controller Prog/Erase/Suspend IRQ_0	MCR[DONE]
218	1368	Ethernet_0_0	EIR[TXF]
219	136C	Ethernet_0_1	EIR[RXF]
220	1370	Ethernet_0_2	EIR[HBERR] EIR[BABR] EIR[BABT] EIR[GRA] EIR[TXB] EIR[RXB] EIR[MII] EIR[EBERR] EIR[LC] EIR[RL] EIR[UN]
226	1388	Periodic Interrupt Timer (PIT0)	PIT_0_TFLG0[TIF]
227	138C	Periodic Interrupt Timer (PIT1)	PIT_0_TFLG1[TIF]
228	1390	Periodic Interrupt Timer (PIT2)	PIT_0_TFLG2[TIF]
229	1394	Periodic Interrupt Timer (PIT3)	PIT_0_TFLG3[TIF]

Table continues on the next page...

Table 1-13. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
230	1398	Periodic Interrupt Timer (PIT4)	PIT_0_TFLG4[TIF]
231	139C	Periodic Interrupt Timer (PIT5)	PIT_0_TFLG5[TIF]
232	13A0	Periodic Interrupt Timer (PIT6)	PIT_0_TFLG6[TIF]
233	13A4	Periodic Interrupt Timer (PIT7)	PIT_0_TFLG7[TIF]
239	13BC	PIT_RTI	PIT_0_RTI_TFLG[TIF]
240	13C0	PIT_64_Lower	PIT_1_TFLG0[TIF]
241	13C4	PIT_64_Upper	PIT_1_TFLG1[TIF]
242	13C8	XOSC counter	XOSC
243	13CC	SIU External Interrupt_0	SIU External Interrupt_0
244	13D0	SIU External Interrupt_1	SIU External Interrupt_1
251	13EC	MC_ME 0	ME_IS[I_SAFE]
252	13F0	MC_ME 1	ME_IS[I_MTC]
253	13F4	MC_ME 2	ME_IS[I_IMODE]
254	13F8	MC_ME 3	ME_IS[I_ICONF]
255	13FC	MC_RGM 0	MC_RGM Functional and destructive reset alternate event interrupt
259	140C	DSPI0_0	DSPI_0_SR[TFUF] | DSPI_0_SR[RFOF]| DSPI_0_SR[TFIWF]
260	1410	DSPI0_1	DSPI_0_SR[EOQF]
261	1414	DSPI0_2	DSPI_0_SR[TFFF]
262	1418	DSPI0_3	DSPI_0_SR[TCF]
263	141C	DSPI0_4	DSPI_0_SR[RFDF]
264	1420	DSPI0_5	DSPI_0_SR[CMD_TCF]
265	1424	DSPI0_6	DSPI_0_SR[CMDFFF]
266	1428	DSPI0_7	DSPI_0_SR[SPEF]
268	1430	DSPI1_0	DSPI_1_SR[TFUF] | DSPI_1_SR[RFOF]| DSPI_1_SR[TFIWF]
269	1434	DSPI1_1	DSPI_1_SR[EOQF]
270	1438	DSPI1_2	DSPI_1_SR[TFFF]
271	143C	DSPI1_3	DSPI_1_SR[TCF]
272	1440	DSPI1_4	DSPI_1_SR[RFDF]
273	1444	DSPI1_5	DSPI_1_SR[CMD_TCF]
274	1448	DSPI1_6	DSPI_1_SR[CMDFFF]
275	144C	DSPI1_7	DSPI_1_SR[SPEF]
277	1454	DSPI2_0	DSPI_2_SR[TFUF] | DSPI_2_SR[RFOF]| DSPI_2_SR[TFIWF]
278	1458	DSPI2_1	DSPI_2_SR[EOQF]
279	145C	DSPI2_2	DSPI_2_SR[TFFF]
280	1460	DSPI2_3	DSPI_2_SR[TCF]
281	1464	DSPI2_4	DSPI_2_SR[RFDF]
282	1468	DSPI2_5	DSPI_2_SR[CMD_TCF]

Table continues on the next page...

Table 1-13. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
283	146C	DSPI2_6	DSPI_2_SR[CMDFFF]
284	1470	DSPI2_7	DSPI_2_SR[SPEF]
286	1478	DSPI3_0	DSPI_3_SR[TFUF] | DSPI_3_SR[RFOF]| DSPI_3_SR[TFIWF]
287	147C	DSPI3_1	DSPI_3_SR[EOQF]
288	1480	DSPI3_2	DSPI_3_SR[TFFF]
289	1484	DSPI3_3	DSPI_3_SR[TCF]
290	1488	DSPI3_4	DSPI_3_SR[RFDF]
291	148C	DSPI3_5	DSPI_3_SR[CMD_TCF]
292	1490	DSPI3_6	DSPI_3_SR[CMDFFF]
293	1494	DSPI3_7	DSPI_3_SR[SPEF]
295	149C	DSPI4_0	DSPI_4_SR[TFUF] | DSPI_4_SR[RFOF]| DSPI_4_SR[TFIWF]
296	14A0	DSPI4_1	DSPI_4_SR[EOQF]
297	14A4	DSPI4_2	DSPI_4_SR[TFFF]
298	14A8	DSPI4_3	DSPI_4_SR[TCF]
299	14AC	DSPI4_4	DSPI_4_SR[RFDF]
300	14B0	DSPI4_5	DSPI_4_SR[CMD_TCF]
301	14B4	DSPI4_6	DSPI_4_SR[CMDFFF]
302	14B8	DSPI4_7	DSPI_4_SR[SPEF]
304	14C0	DSPI_M0_0	DSPI_M0_SR[TFUF] | DSPI_M0_SR[RFOF]| DSPI_M0_SR[TFIWF]
305	14C4	DSPI_M0_1	DSPI_M0_SR[EOQF]
306	14C8	DSPI_M0_2	DSPI_M0_SR[TFFF]
307	14CC	DSPI_M0_3	DSPI_M0_SR[TCF]
308	14D0	DSPI_M0_4	DSPI_M0_SR[RFDF]
309	14D4	DSPI_M0_5	DSPI_M0_SR[SPITCF] | DSPI_M0_SR[CMD_TCF]
310	14D8	DSPI_M0_6	DSPI_M0_SR[DSITCF] | DSPI_M0_SR[CMDFFF]
311	14DC	DSPI_M0_7	DSPI_M0_SR[SPEF] | DSPI_M0_SR[DPEF]
312	14E0	DSPI_M0_8	DSPI_M0_SR[DDIF]
313	14E4	DSPI_M1_0	DSPI_M1_SR[TFUF] | DSPI_M1_SR[RFOF]| DSPI_M1_SR[TFIWF]
314	14E8	DSPI_M1_1	DSPI_M1_SR[EOQF]
315	14EC	DSPI_M1_2	DSPI_M1_SR[TFFF]
316	14F0	DSPI_M1_3	DSPI_M1_SR[TCF]
317	14F4	DSPI_M1_4	DSPI_M1_SR[RFDF]
318	14F8	DSPI_M1_5	DSPI_M1_SR[SPITCF] | DSPI_M1_SR[CMD_TCF]
319	14FC	DSPI_M1_6	DSPI_M1_SR[DSITCF] | DSPI_M1_SR[CMDFFF]

Table continues on the next page...

Table 1-13. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
320	1500	DSPI_M1_7	DSPI_M1_SR[SPEF] | DSPI_M1_SR[DPEF]
321	1504	DSPI_M1_8	DSPI_M1_SR[DDIF]
376	15E0	Linflex/eSCI0_0	LINFlex_0_RXI
377	15E4	Linflex/eSCI0_1	LINFlex_0_TXI
378	15E8	Linflex/eSCI0_2	LINFlex_0_ERR
380	15F0	Linflex/eSCI1_0	LINFlex_1_RXI
381	15F4	Linflex/eSCI1_1	LINFlex_1_TXI
382	15F8	Linflex/eSCI1_2	LINFlex_1_ERR
384	1600	Linflex/eSCI2_0	LINFlex_2_RXI
385	1604	Linflex/eSCI2_1	LINFlex_2_TXI
386	1608	Linflex/eSCI2_2	LINFlex_2_ERR
388	1610	Linflex/eSCI3_0	LINFlex_3_RXI
389	1614	Linflex/eSCI3_1	LINFlex_3_TXI
390	1618	Linflex/eSCI3_2	LINFlex_3_ERR
392	1620	Linflex/eSCI_M0_0	LINFlex_M0_RXI
393	1624	Linflex/eSCI_M0_1	LINFlex_M0_TXI
394	1628	Linflex/eSCI_M0_2	LINFlex_M0_ERR
396	1630	Linflex/eSCI_M1_0	LINFlex_M1_RXI
397	1634	Linflex/eSCI_M1_1	LINFlex_M1_TXI
398	1638	Linflex/eSCI_M1_2	LINFlex_M1_ERR
477	1774	Power Monitor Unit	GR_S[VD15] | GR_S[VD14] | GR_S[VD13] | GR_S[VD12] | GR_S[VD10] | GR_S[VD9] | GR_S[VD7] | GR_S[VD4] | GR_S[VD3]
478	1778	Power management Unit (temp sensor)	EPR_TD[TEMP_3] | EPR_TD[TEMP_2] | EPR_TD[TEMP_0]
480	1780	PLL_0_0	PLL0SR[LOLF]
482	1788	PLL_1_0	PLL1SR[LOLF]
484	1790	PLL_0_1	PLL0SR[EXTPDF]
485	1794	PLL_1_1	PLL1SR[EXTPDF]
488	17A0	FCCU_0	FCCU_IRQ_STAT[ALRM_STAT]
489	17A4	FCCU_1	FCU_IRQ_STAT[CFG_TO_STAT]
490	17A8	FCCU_2	FCCU_SCFS[RCCS0]
491	17AC	FCCU_3	FCCU_SCFS[RCCS1]
494	17B8	STCU2_0	STCU_RUNSW[LBIE]
495	17BC	STCU2_1	STCU_RUNSW[MBIE]
528	1840	SAR_ADC0 Combined interrupt	SAR_ADC_0_EOCTU (end of CTU injected conversion) SAR_ADC_0_JEOC (end of injected conversion of each channel) SAR_ADC_0_JECH (end of injected chain)

Table continues on the next page...

Table 1-13. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
			SAR_ADC_0_EOC (end of normal conversion of each channel) SAR_ADC_0_ECH (end of normal chain)
529	1844	SAR_ADC0_ER	SAR_ADC0_ER
530	1848	SAR_ADC0_WD	SAR_ADC0_WD
532	1850	SAR_ADC1 Combined interrupt	SAR_ADC_1_EOCTU (end of CTU injected conversion) SAR_ADC_1_JEOC (end of injected conversion of each channel) SAR_ADC_1_JECH (end of injected chain) SAR_ADC_1_EOC (end of normal conversion of each channel) SAR_ADC_1_ECH (end of normal chain)
533	1854	SAR_ADC1_ER	SAR_ADC1_ER
534	1858	SAR_ADC1_WD	SAR_ADC1_WD
536	1860	SAR_ADC2 Combined interrupt	SAR_ADC_2_EOCTU (end of CTU injected conversion) SAR_ADC_2_JEOC (end of injected conversion of each channel) SAR_ADC_2_JECH (end of injected chain) SAR_ADC_2_EOC (end of normal conversion of each channel) SAR_ADC_2_ECH (end of normal chain)
537	1864	SAR_ADC2_ER	SAR_ADC2_ER
538	1868	SAR_ADC2_WD	SAR_ADC2_WD
540	1870	SAR_ADC3 Combined interrupt	SAR_ADC_3_EOCTU (end of CTU injected conversion) SAR_ADC_3_JEOC (end of injected conversion of each channel) SAR_ADC_3_JECH (end of injected chain) SAR_ADC_3_EOC (end of normal conversion of each channel) SAR_ADC_3_ECH (end of normal chain)
541	1874	SAR_ADC3_ER	SAR_ADC3_ER
542	1878	SAR_ADC3_WD	SAR_ADC3_WD
544	1880	SD_ADC0	SD_ADC0
545	1884	SD_ADC1	SD_ADC1
546	1888	SD_ADC2	SD_ADC2
558	18B8	SENT_COMBINED_FAST_0	SENT_0_FMSG_RDY[0] | SENT_0_FMSG_RDY[1] | SENT_0_FMSG_RDY[2] | SENT_0_FMSG_RDY[3] | SENT_0_FMSG_RDY[4]

Table continues on the next page...

Table 1-13. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
559	18BC	SENT_COMBINED_SLOW_0	SENT_0_SMSG_RDY[0] | SENT_0_SMSG_RDY[1] | SENT_0_SMSG_RDY[2] | SENT_0_SMSG_RDY[3] | SENT_0_SMSG_RDY[4]
560	18C0	SENT_COMBINED_ERR_0	SENT_0_GBL_STATUS[FMDU] | SENT_0_GBL_STATUS[SMDU] | SENT_0_CHn_STATUS[4:7] | SENT_0_CHn_STATUS[9:15]
561	18C4	SENT_COMBINED_FAST_1	SENT_1_FMSG_RDY[0] | SENT_1_FMSG_RDY[1] | SENT_1_FMSG_RDY[2] | SENT_1_FMSG_RDY[3] | SENT_1_FMSG_RDY[4]
562	18C8	SENT_COMBINED_SLOW_1	SENT_1_SMSG_RDY[0] | SENT_1_SMSG_RDY[1] | SENT_1_SMSG_RDY[2] | SENT_1_SMSG_RDY[3] | SENT_1_SMSG_RDY[4]
563	18CC	SENT_COMBINED_ERR_1	SENT_1_GBL_STATUS[FMDU] | SENT_1_GBL_STATUS[SMDU] | SENT_1_CHn_STATUS[4:7] | SENT_1_CHn_STATUS[9:15]
564	18D0	SENT_0_CH0_FAST	SENT_0_FMSG_RDY[0]
565	18D4	SENT_0_CH0_SLOW	SENT_0_SMSG_RDY[0]
566	18D8	SENT_0_CH0_ERR	SENT_0_CH0_STATUS[4:7] | SENT_0_CH0_STATUS[9:15]
567	18DC	SENT_0_CH1_FAST	SENT_0_FMSG_RDY[1]
568	18E0	SENT_0_CH1_SLOW	SENT_0_SMSG_RDY[1]
569	18E4	SENT_0_CH1_ERR	SENT_0_CH1_STATUS[4:7] | SENT_0_CH1_STATUS[9:15]
570	18E8	SENT_0_CH2_FAST	SENT_0_FMSG_RDY[2]
571	18EC	SENT_0_CH2_SLOW	SENT_0_SMSG_RDY[2]
572	18F0	SENT_0_CH2_ERR	SENT_0_CH2_STATUS[4:7] | SENT_0_CH3_STATUS[9:15]
579	190C	SENT_1_CH0_FAST	SENT_1_FMSG_RDY[0]
580	1910	SENT_1_CH0_SLOW	SENT_1_SMSG_RDY[0]
581	1914	SENT_1_CH0_ERR	SENT_1_CH2_STATUS[4:7] | SENT_1_CH2_STATUS[9:15]
582	1918	SENT_1_CH1_FAST	SENT_1_FMSG_RDY[1]
583	191C	SENT_1_CH1_SLOW	SENT_1_SMSG_RDY[1]
584	1920	SENT_1_CH1_ERR	SENT_1_CH1_STATUS[4:7] | SENT_1_CH1_STATUS[9:15]
585	1924	SENT_1_CH2_FAST	SENT_1_FMSG_RDY[2]
586	1928	SENT_1_CH2_SLOW	SENT_1_SMSG_RDY[2]

Table continues on the next page...

Table 1-13. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
587	192C	SENT_1_CH2_ERR	SENT_1_CH3_STATUS[4:7] | SENT_1_CH3_STATUS[9:15]
654	1A38	SIPI_0	SIPI_ERR[TOEn] | SIPI_ERR[TIDEn] | SIPI_ERR[ACKEn]
655	1A3C	SIPI_1	SIPI_SR[GCRCE)
656	1A40	SIPI_2	SIPI_CSR0[RAR] | SIPI_CSR0[ACKR]
657	1A44	SIPI_3	SIPI_CSR1[RAR] | SIPI_CSR1[ACKR]
658	1A48	SIPI_4	SIPI_CSR2[RAR] | SIPI_CSR2[ACKR]
659	1A4C	SIPI_5	SIPI_CSR3[RAR] | SIPI_CSR3[ACKR]
660	1A50	SIPI_6	SIPI_SR[TE0] | SIPI_SR[TE1] |SIPI_SR[TE2] | SIPI_SR[TE3] |SIPI_SR[MCR]
661	1A54	LFAST0_0	LFAST_0_TISR[TXPNGF] | LFAST_0_TISR[TXUNSF] | LFAST_0_TISR[TXICLCF] | LFAST_0_TISR[TXDTF]
662	1A58	LFAST0_1	LFAST_0_TISR[TXIEF] | LFAST_0_TISR[TXOVF]
663	1A5C	LFAST0_2	LFAST_0_RISR[RXCTSF] | LFAST_0_RISR[RXDF] | LFAST_0_RISR[RXUNSF]
664	1A60	LFAST0_3	LFAST_0_RISR[RXUOF] | LFAST_0_RISR[RXMNF] | LFAST_0_RISR[RXMXF] | LFAST_0_RISR[RXUFF] | LFAST_0_RISR[RXOFF] | LFAST_0_RISR[RXSZF] | LFAST_0_RISR[RXICF] | LFAST_0_RISR[RXLCEF]
665	1A64	LFAST0_4	LFAST_0_RIISR[ICPFF] | LFAST_0_RIISR[ICPSF] | LFAST_0_RIISR[ICPRF] | LFAST_0_RIISR[ICTOF] | LFAST_0_RIISR[ICLPF] | LFAST_0_RIISR[ICCTF] | LFAST_0_RIISR[ICTDF] | LFAST_0_RIISR[ICTEF] | LFAST_0_RIISR[ICRFF] | LFAST_0_RIISR[ICRSF] | LFAST_0_RIISR[ICTFF] | LFAST_0_RIISR[ICTSF] | LFAST_0_RIISR[ICPOFF] | LFAST_0_RIISR[ICPONF]
674	1A88	JTAGM	JTAGM_SR[SPU_INT] | JTAGM_SR[Idle]
675	1A8C	JDC	JDC_MSR[JIN_INT] | JDC_MSR[JOUT_INT]
677	1A94	FlexCAN_0_0	CAN0_ESR1[BOFFINT] | CAN0_ESR1[BOFFDONEINT]
678	1A98	FlexCAN_0_1	CAN0_ESR1[ERRINT] | CAN0_ESR1[ERRINT_FAST]

Table continues on the next page...

Table 1-13. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
679	1A9C	FlexCAN_0_5	CAN0_IFLAG1[BUF15:BUF0]
680	1AA0	FlexCAN_0_2	CAN0_IFLAG1[BUF31:BUF16]
681	1AA4	FlexCAN_0_3	CAN0_IFLAG2[BUF63:BUF32]
682	1AA8	FlexCAN_0_4	CAN0_IFLAG3[BUF95:BUF64]
683	1AAC	FlexCAN_1_0	CAN1_ESR1[BOFFINT] | CAN1_ESR1[BOFFDONEINT]
684	1AB0	FlexCAN_1_1	CAN1_ESR1[ERRINT] | CAN1_ESR1[ERRINT_FAST]
685	1AB4	FlexCAN_1_5	CAN1_IFLAG1[BUF15:BUF0]
686	1AB8	FlexCAN_1_2	CAN1_IFLAG1[BUF31:BUF16]
687	1ABC	FlexCAN_1_3	CAN1_IFLAG2[BUF63:BUF32]
688	1AC0	FlexCAN_1_4	CAN1_IFLAG3[BUF95:BUF64]
689	1AC4	FlexCAN_2_0	CAN2_ESR1[BOFFINT]
690	1AC8	FlexCAN_2_1	CAN2_ESR1[ERRINT]
691	1ACC	FlexCAN_2_5	CAN2_IFLAG1[BUF7:BUF0]
692	1AD0	FlexCAN_2_2	CAN2_IFLAG1[BUF15:BUF8]
693	1AD4	FlexCAN_2_3	CAN2_IFLAG1[BUF31:BUF16]
694	1AD8	FlexCAN_2_4	CAN2_IFLAG2[BUF63:BUF32]
695	1ADC	FlexCAN_3_0	CAN3_ESR1[BOFFINT]
696	1AE0	FlexCAN_3_1	CAN3_ESR1[ERRINT]
697	1AE4	FlexCAN_3_5	CAN3_IFLAG1[BUF7:BUF0]
698	1AE8	FlexCAN_3_2	CAN3_IFLAG1[BUF15:BUF8]
699	1AEC	FlexCAN_3_3	CAN3_IFLAG1[BUF31:BUF16]
700	1AF0	FlexCAN_3_4	CAN3_IFLAG2[BUF63:BUF32]
706	1B08	eMIOS_0 channel 8	EMIOS_0_GFR[F8]
707	1B0C	eMIOS_0 channel 9	EMIOS_0_GFR[F9]
708	1B10	eMIOS_0 channel 10	EMIOS_0_GFR[F10]
709	1B14	eMIOS_0 channel 11	EMIOS_0_GFR[F11]
710	1B18	eMIOS_0 channel 12	EMIOS_0_GFR[F12]
711	1B1C	eMIOS_0 channel 13	EMIOS_0_GFR[F13]
712	1B20	eMIOS_0 channel 14	EMIOS_0_GFR[F14]
713	1B24	eMIOS_0 channel 15	EMIOS_0_GFR[F15]
714	1B28	eMIOS_0 channel 16	EMIOS_0_GFR[F16]
715	1B2C	eMIOS_0 channel 17	EMIOS_0_GFR[F17]
716	1B30	eMIOS_0 channel 18	EMIOS_0_GFR[F18]
717	1B34	eMIOS_0 channel 19	EMIOS_0_GFR[F19]
718	1B38	eMIOS_0 channel 20	EMIOS_0_GFR[F20]
719	1B3C	eMIOS_0 channel 21	EMIOS_0_GFR[F21]
720	1B40	eMIOS_0 channel 22	EMIOS_0_GFR[F22]
721	1B44	eMIOS_0 channel 23	EMIOS_0_GFR[F23]

Table continues on the next page...

Table 1-13. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
722	1B48	eMIOS_1 channel 8	EMIOS_1_GFR[F8]
723	1B4C	eMIOS_1 channel 9	EMIOS_1_GFR[F9]
724	1B50	eMIOS_1 channel 10	EMIOS_1_GFR[F10]
725	1B54	eMIOS_1 channel 11	EMIOS_1_GFR[F11]
726	1B58	eMIOS_1 channel 12	EMIOS_1_GFR[F12]
727	1B5C	eMIOS_1 channel 13	EMIOS_1_GFR[F13]
728	1B60	eMIOS_1 channel 14	EMIOS_1_GFR[F14]
729	1B64	eMIOS_1 channel 15	EMIOS_1_GFR[F15]
730	1B68	eMIOS_1 channel 16	EMIOS_1_GFR[F16]
731	1B6C	eMIOS_1 channel 17	EMIOS_1_GFR[F17]
732	1B70	eMIOS_1 channel 18	EMIOS_1_GFR[F18]
733	1B74	eMIOS_1 channel 19	EMIOS_1_GFR[F19]
734	1B78	eMIOS_1 channel 20	EMIOS_1_GFR[F20]
735	1B7C	eMIOS_1 channel 21	EMIOS_1_GFR[F21]
736	1B80	eMIOS_1 channel 22	EMIOS_1_GFR[F22]
737	1B84	eMIOS_1 channel 23	EMIOS_1_GFR[F23]
738	1B88	eTPU Engine 1 and 2 Global Exception	eTPU_SCR_MCR[MGE1,MGE2,ILF1,ILF2,SCMMISF]
739	1B8C	eTPU Engine 1 Channel 0	eTPU_GCR_CISR_ENG1[CIS0]
740	1B90	eTPU Engine 1 Channel 1	eTPU_GCR_CISR_ENG1[CIS1]
741	1B94	eTPU Engine 1 Channel 2	eTPU_GCR_CISR_ENG1[CIS2]
742	1B98	eTPU Engine 1 Channel 3	eTPU_GCR_CISR_ENG1[CIS3]
743	1B9C	eTPU Engine 1 Channel 4	eTPU_GCR_CISR_ENG1[CIS4]
744	1BA0	eTPU Engine 1 Channel 5	eTPU_GCR_CISR_ENG1[CIS5]
745	1BA4	eTPU Engine 1 Channel 6	eTPU_GCR_CISR_ENG1[CIS6]
746	1BA8	eTPU Engine 1 Channel 7	eTPU_GCR_CISR_ENG1[CIS7]
747	1BAC	eTPU Engine 1 Channel 8	eTPU_GCR_CISR_ENG1[CIS8]
748	1BB0	eTPU Engine 1 Channel 9	eTPU_GCR_CISR_ENG1[CIS9]
749	1BB4	eTPU Engine 1 Channel 10	eTPU_GCR_CISR_ENG1[CIS10]
750	1BB8	eTPU Engine 1 Channel 11	eTPU_GCR_CISR_ENG1[CIS11]
751	1BBC	eTPU Engine 1 Channel 12	eTPU_GCR_CISR_ENG1[CIS12]
752	1BC0	eTPU Engine 1 Channel 13	eTPU_GCR_CISR_ENG1[CIS13]
753	1BC4	eTPU Engine 1 Channel 14	eTPU_GCR_CISR_ENG1[CIS14]
754	1BC8	eTPU Engine 1 Channel 15	eTPU_GCR_CISR_ENG1[CIS15]
755	1BCC	eTPU Engine 1 Channel 16	eTPU_GCR_CISR_ENG1[CIS16]
756	1BD0	eTPU Engine 1 Channel 17	eTPU_GCR_CISR_ENG1[CIS17]
757	1BD4	eTPU Engine 1 Channel 18	eTPU_GCR_CISR_ENG1[CIS18]
758	1BD8	eTPU Engine 1 Channel 19	eTPU_GCR_CISR_ENG1[CIS19]
759	1BDC	eTPU Engine 1 Channel 20	eTPU_GCR_CISR_ENG1[CIS20]
760	1BE0	eTPU Engine 1 Channel 21	eTPU_GCR_CISR_ENG1[CIS21]

Table continues on the next page...

Table 1-13. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
761	1BE4	eTPU Engine 1 Channel 22	eTPU_GCR_CISR_ENG1[CIS22]
762	1BE8	eTPU Engine 1 Channel 23	eTPU_GCR_CISR_ENG1[CIS23]
763	1BEC	eTPU Engine 1 Channel 24	eTPU_GCR_CISR_ENG1[CIS24]
764	1BF0	eTPU Engine 1 Channel 25	eTPU_GCR_CISR_ENG1[CIS25]
765	1BF4	eTPU Engine 1 Channel 26	eTPU_GCR_CISR_ENG1[CIS26]
766	1BF8	eTPU Engine 1 Channel 27	eTPU_GCR_CISR_ENG1[CIS27]
767	1BFC	eTPU Engine 1 Channel 28	eTPU_GCR_CISR_ENG1[CIS28]
768	1C00	eTPU Engine 1 Channel 29	eTPU_GCR_CISR_ENG1[CIS29]
769	1C04	eTPU Engine 1 Channel 30	eTPU_GCR_CISR_ENG1[CIS30]
770	1C08	eTPU Engine 1 Channel 31	eTPU_GCR_CISR_ENG1[CIS31]
771	1C0C	eTPU Engine 2 Channel 0	eTPU_GCR_CISR_ENG2[CIS0]
772	1C10	eTPU Engine 2 Channel 1	eTPU_GCR_CISR_ENG2[CIS1]
773	1C14	eTPU Engine 2 Channel 2	eTPU_GCR_CISR_ENG2[CIS2]
774	1C18	eTPU Engine 2 Channel 3	eTPU_GCR_CISR_ENG2[CIS3]
775	1C1C	eTPU Engine 2 Channel 4	eTPU_GCR_CISR_ENG2[CIS4]
776	1C20	eTPU Engine 2 Channel 5	eTPU_GCR_CISR_ENG2[CIS5]
777	1C24	eTPU Engine 2 Channel 6	eTPU_GCR_CISR_ENG2[CIS6]
778	1C28	eTPU Engine 2 Channel 7	eTPU_GCR_CISR_ENG2[CIS7]
779	1C2C	eTPU Engine 2 Channel 8	eTPU_GCR_CISR_ENG2[CIS8]
780	1C30	eTPU Engine 2 Channel 9	eTPU_GCR_CISR_ENG2[CIS9]
781	1C34	eTPU Engine 2 Channel 10	eTPU_GCR_CISR_ENG2[CIS10]
782	1C38	eTPU Engine 2 Channel 11	eTPU_GCR_CISR_ENG2[CIS11]
783	1C3C	eTPU Engine 2 Channel 12	eTPU_GCR_CISR_ENG2[CIS12]
784	1C40	eTPU Engine 2 Channel 13	eTPU_GCR_CISR_ENG2[CIS13]
785	1C44	eTPU Engine 2 Channel 14	eTPU_GCR_CISR_ENG2[CIS14]
786	1C48	eTPU Engine 2 Channel 15	eTPU_GCR_CISR_ENG2[CIS15]
787	1C4C	eTPU Engine 2 Channel 16	eTPU_GCR_CISR_ENG2[CIS16]
788	1C50	eTPU Engine 2 Channel 17	eTPU_GCR_CISR_ENG2[CIS17]
789	1C54	eTPU Engine 2 Channel 18	eTPU_GCR_CISR_ENG2[CIS18]
790	1C58	eTPU Engine 2 Channel 19	eTPU_GCR_CISR_ENG2[CIS19]
791	1C5C	eTPU Engine 2 Channel 20	eTPU_GCR_CISR_ENG2[CIS20]
792	1C60	eTPU Engine 2 Channel 21	eTPU_GCR_CISR_ENG2[CIS21]
793	1C64	eTPU Engine 2 Channel 22	eTPU_GCR_CISR_ENG2[CIS22]
794	1C68	eTPU Engine 2 Channel 23	eTPU_GCR_CISR_ENG2[CIS23]
795	1C6C	eTPU Engine 2 Channel 24	eTPU_GCR_CISR_ENG2[CIS24]
796	1C70	eTPU Engine 2 Channel 25	eTPU_GCR_CISR_ENG2[CIS25]
797	1C74	eTPU Engine 2 Channel 26	eTPU_GCR_CISR_ENG2[CIS26]
798	1C78	eTPU Engine 2 Channel 27	eTPU_GCR_CISR_ENG2[CIS27]
799	1C7C	eTPU Engine 2 Channel 28	eTPU_GCR_CISR_ENG2[CIS28]

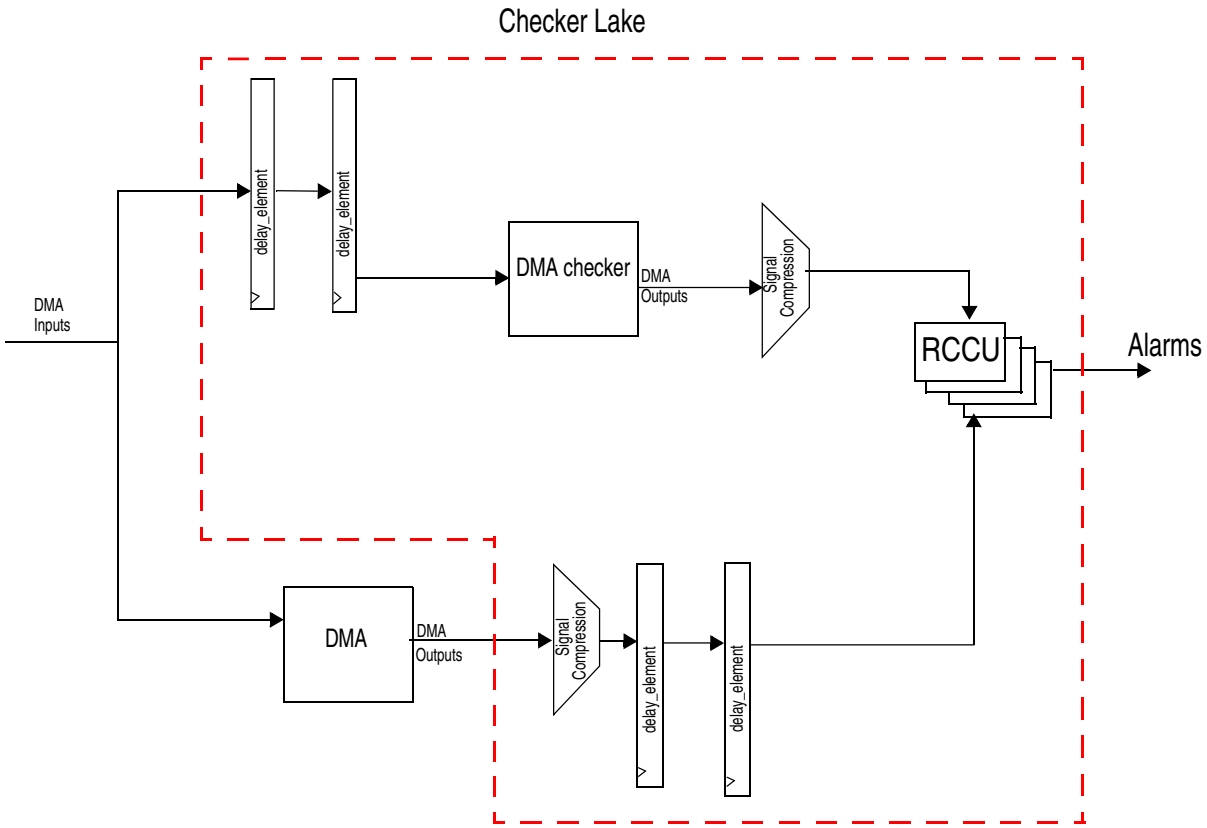
Table continues on the next page...

Table 1-13. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
800	1C80	eTPU Engine 2 Channel 29	eTPU_GCR_CISR_ENG2[CIS29]
801	1C84	eTPU Engine 2 Channel 30	eTPU_GCR_CISR_ENG2[CIS30]
802	1C88	eTPU Engine 2 Channel 31	eTPU_GCR_CISR_ENG2[CIS31]
900	1E10	BCTU_0_TRIG	BCTU_0_TRIG
901	1E14	BCTU_0_ADC0	BCTU_0_ADC0
902	1E18	BCTU_0_ADC1	BCTU_0_ADC1
903	1E1C	BCTU_0_ADC2	BCTU_0_ADC2
904	1E20	BCTU_0_ADC3	BCTU_0_ADC3
905	1E24	BCTU_0_LIST	BCTU_0_LIST
915	1E4C	REACM_G	ipi_int_reacm
916	1E50	REACM_0	ipi_int_reacm_ch0
917	1E54	REACM_1	ipi_int_reacm_ch1
918	1E58	REACM_2	ipi_int_reacm_ch2
919	1E5C	REACM_3	ipi_int_reacm_ch3
920	1E60	REACM_4	ipi_int_reacm_ch4
921	1E64	REACM_5	ipi_int_reacm_ch5
922	1E68	REACM_6	ipi_int_reacm_ch6
923	1E6C	REACM_7	ipi_int_reacm_ch7
924	1E70	REACM_8	ipi_int_reacm_ch8
925	1E74	REACM_9	ipi_int_reacm_ch9
932	1E90	DECFIL_0_In	ipi_int_dec_filter_0_in
933	1E94	DECFIL_0_Out	ipi_int_dec_filter_0_out
934	1E98	DECFIL_0_Err	ipi_int_dec_filter_0_err
935	1E9C	DECFIL_1_In	ipi_int_dec_filter_1_in
936	1EA0	DECFIL_1_Out	ipi_int_dec_filter_1_out
937	1EA4	DECFIL_1_Err	ipi_int_dec_filter_1_err

1.5 Lockstep

This chip contains an eDMA Checker which, when enabled, operates in Lockstep mode with the primary eDMA, executing the exact same transfer profiles. The eDMA Checker checks to ensure the primary eDMA is operating correctly. A block diagram is shown below.



1.5.1 Initialization

To prevent the eDMA Checker from generating a false comparison error after power-on-reset, the eDMA's data output buffer must be initialized prior to enabling the Checker. To initialize the eDMA data output buffer, the eDMA must perform one or more 32-bit data transfers using any data value.

1.5.2 Errors

To handle source-address errors (SAE), perform the following steps.

1. Fix the addressing error.
2. Clear the error bit in "Enable Error Interrupt RegisterLow (DMA_ERRL)" or "Error Interrupt Register High (DMA_ERRH)".
3. Start the channel in a normal fashion.

NOTE

"Error Status Register (DMA_ES)" is read-only; the bits cannot be cleared. It saves the last recorded error. The VLD bit shows

the user whether any error bits in the Error Registers are set, thus indicating an error occurred that has not been cleared.

1.6 LINFlex clocking

The LINFlex protocol clock (LINCLK) domain selection may be configured via soft DCF client programming...

[*Content deleted for brevity.*]

In either mode the LINCLK must be enabled by setting the Auxiliary Clock 0 Divider Configuration Register (MC_CGM_AC0_DC4) Divider Enable (DE) bit to 0b1.

1.7 Clock configuration

The relationship between input and output frequency is determined by programming the PLL0DV, PLL1DV, and PLL1FD registers, and calculated according to the following equations:

$$f_{pll0_phi} = f_{pll0_ref} \times \left(\frac{PLL0DV[MFD]}{PLL0DV[PREDIV] \times PLL0DV[RFDPHI]} \right)$$

Equation 1. PLL0 PHI Output Frequency

$$f_{pll0_phi1} = f_{pll0_ref} \times \left(\frac{PLL0DV[MFD]}{PLL0DV[PREDIV] \times PLL0DV[RFDPHI1]} \right)$$

Equation 2. PLL0 PHI1 Output Frequency

$$f_{pll1_phi} = f_{pll1_ref} \times \left(\frac{PLL1DV[MFD] + \frac{PLL1FD[FRCDIV]}{2^{12}} + \frac{1}{2^{13}}}{2 \times PLL1DV[RFDPHI]} \right)$$

Equation 3. PLL1 PHI Output Frequency (PLLDIG_PLL1FD[FDEN] = 1b)

$$f_{pll1_phi} = f_{pll1_ref} \times \left(\frac{PLL1DV[MFD]}{2 \times PLL1DV[RFDPHI]} \right)$$

Equation 4. PLL1 PHI Output Frequency (PLLDIG_PLL1FD[FDEN] = 0b)

The relationship between the VCO frequency (f_{VCO}) and the output frequency of the PLLs is determined by the configuration of the PLL1DV, PLL1FD, and PLL0DV registers, according to the following equations:

$$f_{pll0_VCO} = f_{pll0_ref} \times \left(\frac{PLL0DV[MFD] \times 2}{PLL0DV[PREDIV]}\right)$$

Equation 5. PLL0 VCO Frequency

$$f_{pll1_VCO} = f_{pll1_ref} \times \left(PLL1DV[MFD] + \frac{PLL1FD[FRCDIV]}{2^{12}} + \frac{1}{2^{13}}\right)$$

Equation 6. PLL1 VCO Frequency (PLLDIG_PLL1FD[FDEN] = 1b)

$$f_{pll1_VCO} = f_{pll1_ref} \times PLL1DV[MFD]$$

Equation 7. PLL1 VCO Frequency (PLLDIG_PLL1FD[FDEN] = 0b)

NOTE

f_{pll0_phi1} is the reference clock generated by PLL0 for PLL1

When programming the PLLs, user software must not violate the maximum system clock frequency or max/min VCO frequency specification of PLL0 and PLL1. Furthermore, the PLL0DV[PREDIV] value must not be set to any value that causes the input frequency to the phase detector of analog PLL blocks to go below the prescribed ranges.

The lock signal and PLL*n*SR[LOCK] flag are immediately negated if the fields of the PLL*n*DV registers are changed without powering down the analog PLLs.

When any of these events occur, an internal timer is initialized to count 64 cycles of the PLL input clock. During this period (64 cycles and a few extra clock cycles for synchronization, for example, 64 to 72 cycles), the PLL*n*SR[LOCK] flag is held negated. After the timer expires, the PLL*n*SR[LOCK] flag reflects the value coming from the PLL lock detection circuitry. To prevent an immediate reset, the PLL*n*CR[LOLRE] bit of the respective PLLs must be cleared before doing any of the above operations.

Note

The PLL must be powered down and powered up by configuring, then executing, a mode change in the MC_ME_<*mode*>_MC before PLL0DV[PREDIV], PLL0DV[MFD], PLL1DV[MFD], or the input clocks are modified.

The recommended procedure to program the PLLs and engage normal mode is shown in Initialization information.

Coming out of reset, PLL1 and PLL0 are off and in bypass mode. The recommended procedure to program the PLLs and engage normal mode is described in the 'Clocking' chapter.

1.8 CMU Measurement Duration Register (CMU_MDR)

Added the following note in the CMU_MDR register field description for the MD field:

Only values that do not cause a CMU_FDR[FD] overflow shall be written to CMU_MDR[MD]. There is no internal status of a CMU_FDR[FD] overflow.

1.9 System Clock Dividers

The MC_CGM generates the following derived clocks from the system clock.

- Fast XBAR clock - controlled by the **CGM_SC_DC0** register
- Slow XBAR clock - controlled by the **CGM_SC_DC1** register
- AIPS clock - controlled by the **CGM_SC_DC2** register

NOTE

All masters, except the cores, must not be active during system clock divider configuration.

1.10 Auxiliary Clock Generation

Added the following content at the end of the section.

The following procedures avoid undefined system behavior.

Auxiliary clock enable procedure:

1. Software needs to ensure the clock source is enabled and available (for example, oscillator stable, PLL locked).
2. Configure IRC clock source in selector registers (MC_CGM_AC*x*_SC).
3. Configure the appropriate clock source selector registers (MC_CGM_AC*x*_SC).
4. Configure and enable any corresponding auxiliary clock dividers (MC_CGM_AC*x*_DC*n*). This should be done in an atomic 32-bit wide register access.

Auxiliary clock disable procedure:

1. Software needs to ensure that the state of the module driven by this clock allows to disable the clock.
2. Disable auxiliary clock dividers (MC_CGM_AC*x*_DC*n*[DE] = 0).
3. Wait for at least 27 system clock cycles to allow divider configuration to complete.

In the event a switch to a different clock source is required, while auxiliary clock has been configured and used, the above auxiliary clock disable procedure followed by the above auxiliary clock enable procedure should be applied.

In the event the clock divider value requires to be altered, while auxiliary clock has been configured and used, this can be done by changing the value in the corresponding clock divider configuration register (MC_CGM_AC*x*_DC*n*).

Before any clock source is switched off, all clock divider need to be disabled.

1.11 eTPU chapter

See Chapter 2, Enhanced Time Processing Unit (eTPU) for an updated eTPU chapter, and Appendix A for a listing of all the changes in that chapter.

1.12 BCTU Introduction

The Body Cross-triggering Unit (BCTU) is a module that is used to trigger the on chip ADCs. The trigger sources are generated by on chip timer modules. Usually in Body domain of applications the timers used are the enhanced Modular IO Sub-system (eMIOS), enhanced Time Processing Unit (ETPU),and the Programmable Interrupt Timer (PIT). In addition to triggering ADC single conversions, the BCTU also implements a LIST—a set of registers programmed by software that stores ADC channel numbers—that triggers several conversions based on a single trigger occurrence. Up to 32 trigger inputs are available. The CPU can also be a source of conversion triggers, thus implementing a software trigger control. Up to 4 ADC modules are supported. The conversion results are stored in a register which is available for CPU read access. Figure below shows the BCTU top level diagram (the figure is not included in this RMAD because it was not modified).

1.13 BCTU memory map

Updated the "BCTU memory map" table as shown below.

Before:

22C	Software Trigger register 1 (BCTU_SFTRGR1)	32	R/W	0000_0000h	46.6.5/1799
234	Software Trigger register 1 (BCTU_SFTRGR3)	32	R/W	0000_0000h	46.6.6/1802
23C	ADC0 Result Data register (BCTU_ADC0DR)	32	R	0000_0000h	46.6.7/1806

After:

22C	Software Trigger register 1 (BCTU_SFTRGR1)	32	R/W	0000_0000h	46.6.5/1799
23C	ADC0 Result Data register (BCTU_ADC0DR)	32	R	0000_0000h	46.6.7/1806

Figure 1-1. BCTU memory map changes

1.14 Status Register (DSPI_SR)

In the Status Register (DSPI_SR), changed TXRXS bitfield access from W1C to RO.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF	TXRXS	SPITCF	EOQF	TFUF	DSITCF	TFFF	BSYF	CMDTCF	DPEF	SPEF	DDIF	RFOF	TFIWF	RFDF	CMDFFF
W	w1c		w1c	w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXCTR				TXNXTPTR				RXCTR				POPNXTPTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1-2. Status Register (DSPI_SR)

1.15 Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 1)

Added the following note:

NOTE

When MTFE=1 with continuous SCK enabled (MCR [CONT_SCKE] =1) in master mode, configure CTAR[LSBFE]=0 for correct operations while receiving unequal length frames. If PUSHR[CONT] is also set for back to back frame transfer, also configure the frame size of the first frame as less than or equal to the frame size of the next frame. In this scenario, make sure that for all received frames, the bits are read equal to their respective frame sizes and any extra bits during POP operation are masked.

1.16 *PHASE2[FUNC]* Phase: flash initial configuration

This phase is entered on exit from ***PHASE1[FUNC]***. The reset state machine exits ***PHASE2[FUNC]*** and enters ***PHASE3[FUNC]*** on verification of the following:

- All processes that need to be done in ***PHASE2[FUNC]*** are completed

During this phase, the reset is released to the flash which can start its initial configuration. Watchdog is monitoring execution ensuring reset in case of flash configuration failure.

This phase is duplication of ***PHASE2[DEST]*** after self-test.

1.17 *PHASE3[FUNC]* Phase: device configuration monitoring

This phase is entered either on exit from ***PHASE2[FUNC]*** or immediately from ***IDLE*** on an enabled, shortened 'functional' reset event. The reset state machine exits ***PHASE3[FUNC]*** and enters ***IDLE*** on verification of the following:

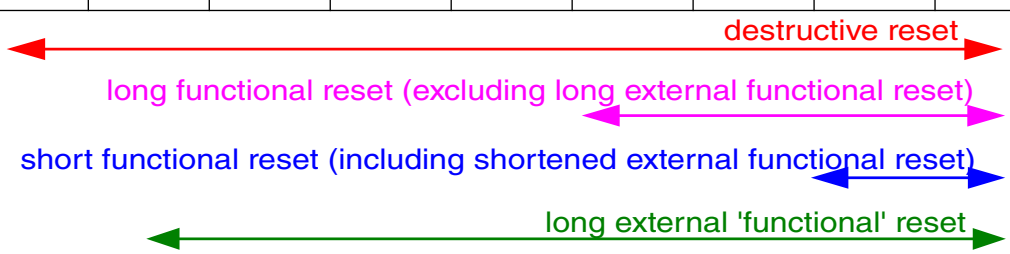
- All processes that need to be done in ***PHASE3[FUNC]*** are completed. In particular PORST and RESET pin are not forced low externally.
- A minimum number of cycles have elapsed since the last enabled, shortened 'functional' reset event

After ***PHASE3[FUNC]*** internal actions have been completed, it is possible to maintain the device within ***PHASE3[FUNC]*** by forcing either PORST or RESET pin low. This can be used for debug purpose to connect through JTAG to the internal debugger.

1.18 Module status during reset sequence

The following table is providing status of main modules during the different reset phases:

Table 1-14. Module status during reset phases

Module	PHASE								
	P0 (RED)	P1 [DEST] (RED)	P2 [DEST] (RED)	P3 [DEST] (RED)	IDLE [DEST] (RED)	P1 [FUNC] (PURPLE)	P2 [FUNC] (PURPLE)	P3 [FUNC] (BLUE)	IDLE [FUNC] (WHITE)
Internal RCOSC	ON	ON	ON	ON	ON	ON	ON	ON	ON
Power management controller	ON	ON	ON	ON	ON	ON	ON	ON	ON
Reset management controller	ON[1]	ON	ON	ON	ON	ON	ON	ON	ON
system watchdog	RESET	RESET	ON	RESET	LBIST[2]	RESET	ON	RESET	ON[3]
flash module	RESET	RESET	ON	ON	LBIST	RESET	ON	ON	ON
OSC	RESET	RESET	RESET	ON[3]	ON[3]	ON[3]	ON[3]	ON[3]	ON[3]
DEVICE CONFIG	RESET	ON	ON	LOAD	HOLD	HOLD	HOLD	LOAD	HOLD
SSCM	RESET	RESET	RESET	ON	ON	RESET	RESET	ON	ON
FCCU	RESET	RESET	CONFIG	CONFIG	LBIST	RESET/ON[4]	CONFIG/ON	CONFIG/ON	ON
STCU	RESET	RESET	CONFIG	CONFIG	ON	IDLE	IDLE	IDLE	IDLE[5]
pit_rti_1 timer	RESET	RESET	RESET	RESET	LBIST	IDLE/ON	IDLE/ON	IDLE/ON	IDLE/ON
other modules including cores[6]	RESET	RESET	RESET	RESET	LBIST	RESET	RESET	RESET	ON[7]
Debug Modules	RESET	RESET	RESET	RESET	RESET	ON	ON	ON	ON
									

1. RESET only during POR.
2. STCU time-out monitoring LBIST/MBIST execution
3. Depending on flash configuration
4. FCCU remains ON during functional reset, it is instead reset in PHASE1[FUNC] if coming from IDLE[DEST]
5. Can be triggered by SW

6. The ADC is reset as described in the table, but the ADC_CALBISTREG and ADC_OFSGNUSR registers and the ADC_MSR[CALIBRTD] field are only reset by PHASE1[FUNC] or PHASE1[DEST].
7. After enabling from BAF

1.19 Overview

The reset generation module (MC_RGM) centralizes the different reset sources and manages the reset sequence of the chip. It provides a register interface and a reset sequencer. Various registers are available to monitor and control the chip reset sequence. The reset sequencer is a state machine which controls the different phases (***PHASE0***, ***PHASE1***, ***PHASE2***, ***PHASE3***, and ***IDLE***) of the reset sequence and the reset signals generated in the system.

The following figure shows the MC_RGM block diagram.

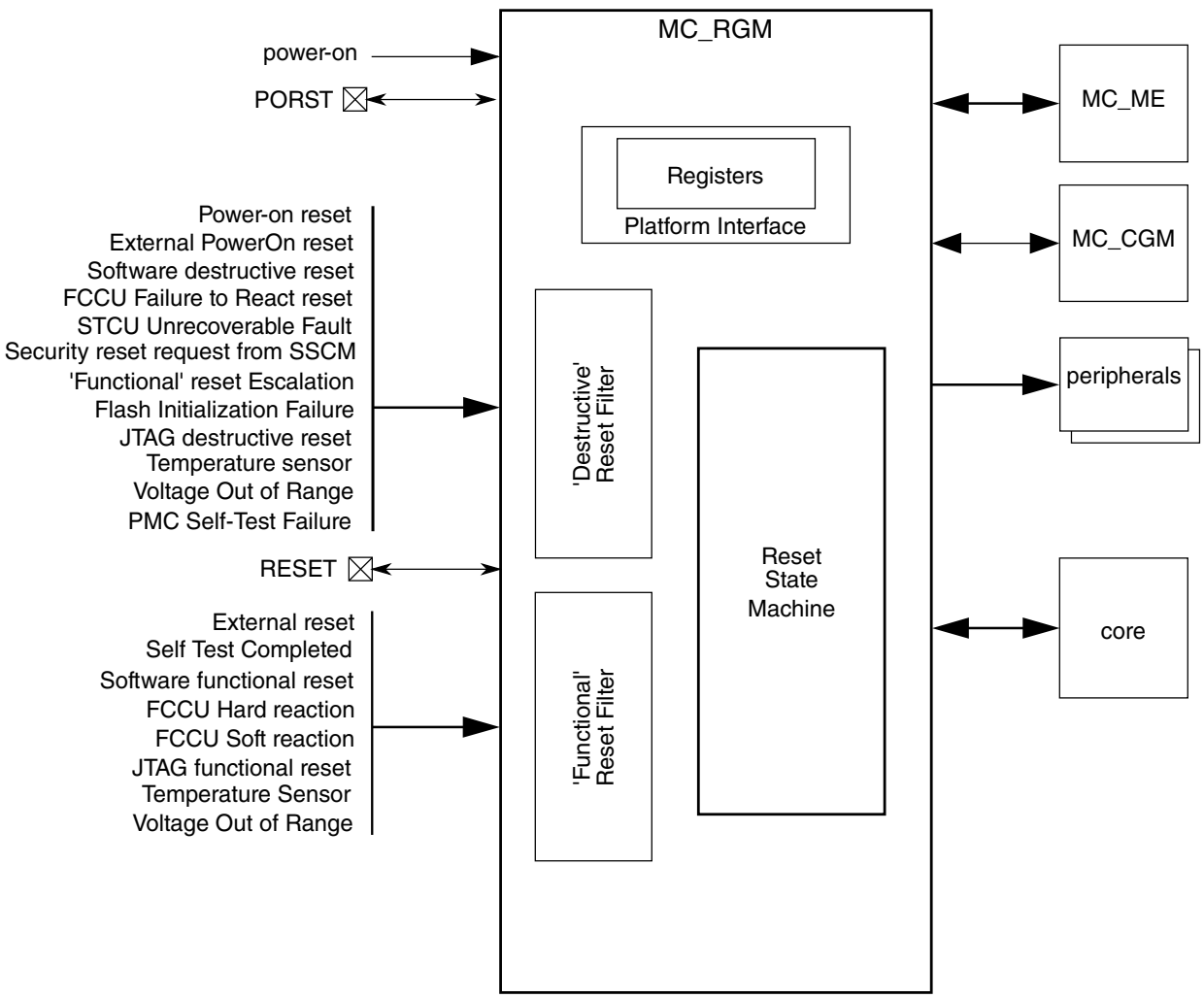


Figure 1-3. MC_RGM Block Diagram

Peripheral resets cannot be controlled individually.

1.20 BAF NMI event handling

The BAF executes under various conditions such as Power-On-Reset (POR), Destructive Reset or any Functional Reset (Long or Short). Due to device configuration, there can be a case where pending FCCU NMI events are latched in the device when the BAF starts execution. These NMI events are masked until a mode transition is completed after reset. The BAF issues a mode transition before it passes control to the application. So, in such case the application may get an NMI exception before it executes its first instruction. Application IVPR can be installed before it gets control by using DCF DCF_IVPR0 for CPU0.

1.21 System Status and Configuration Module (SSCM) chip-specific module information

Added the following paragraph to this section following Table 58-2:

The device will search for valid RCHW locations in the order shown in Table 58-2, "SSCM startup information addresses", and will use the first valid entry to determine the boot location. Care must be taken that a location is not accidentally written with a valid RCHW marker when the intended location is only searched later.

1.22 ECC error monitoring

Added the following:

NOTE

During DCF Scanning, if multibit ECC error on DCF start record and DCF start record pattern is corrupted then, SSCM will not be able to identify that as a valid start record and will ignore the flash region as well. In this case, an ECC error address would be logged in FCCU or MEMU. SoC software can monitor FCCU or MEMU for any address of start record. If any start record address found in FCCU or MEMU address log then software should generate a security reset.

1.23 Power Management Controller (PMC)

The Power Management Controller implements a linear voltage regulator to generate the internal 1.25V digital supply from an external 3.5V to 5.5V supply and monitors various supply voltages of the MPC5746R chip for functional safety purposes. An additional regulator for the flash block is also within the PMC to provide the necessary 3.3V source to the FLASH module. Another internal regulator is present and generates a clean 1.26V supply for the LFAST PLL, as shown in the diagram in the next section. Depending on the supply voltages, power-on-reset (POR) and destructive/functional resets may be generated.

The PMC connects to the Reset Generation Module (MC_RGM), the Fault Collection Control Unit (FCCU) and the flash memory controller. The on-chip ADC receives signals from the PMC for diagnostic purposes.

1.24 Supply Concept

Some supplies may already be connected internally inside the package depending on the package type, mainly in order to save pins in low pin count packages.

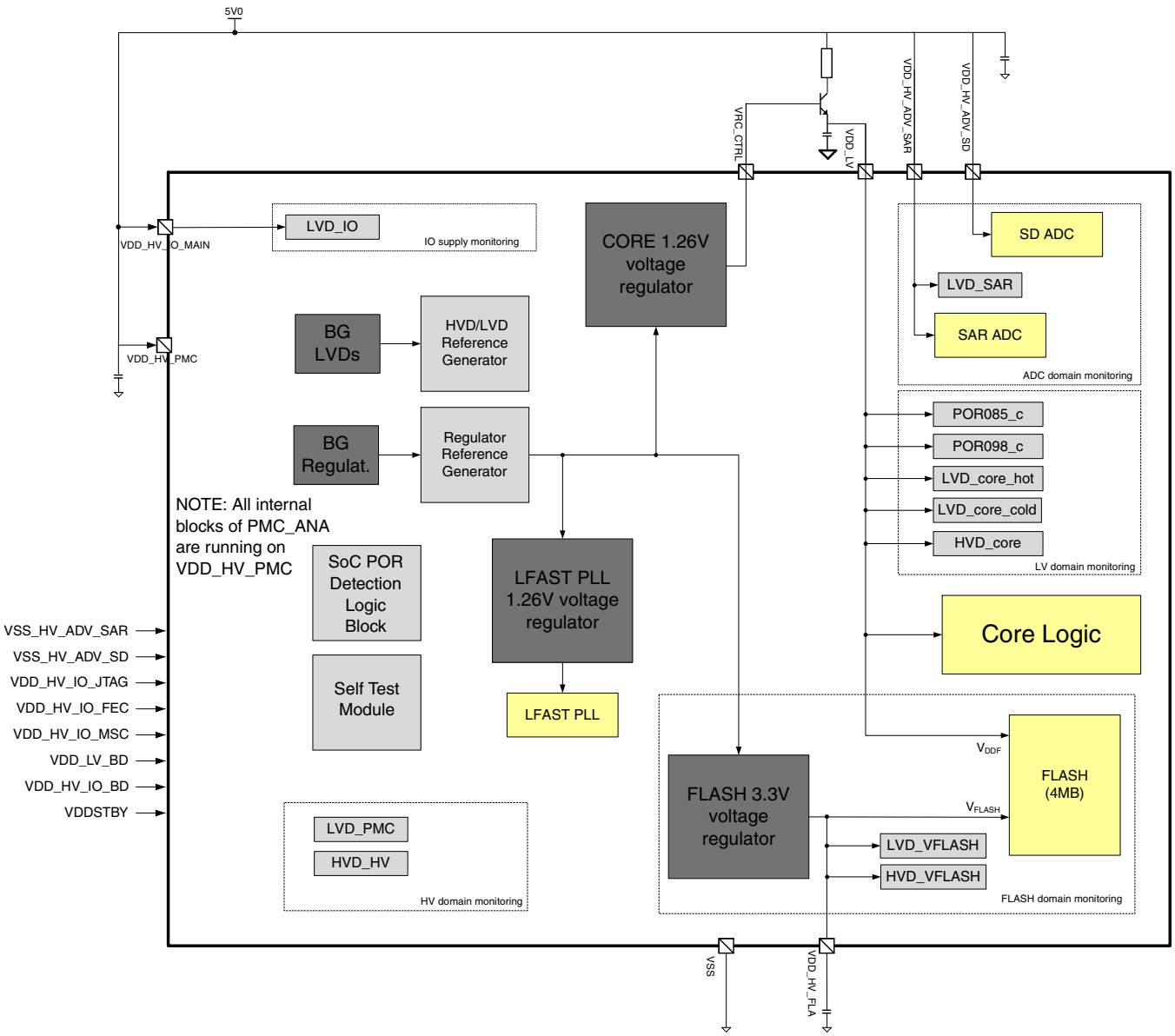


Figure 1-4. Supply concept

1.25 Chip-specific PMC information

The PMC implements a linear voltage regulator to generate the internal 1.25V digital supply from an external 3.5V to 5.5V supply. It also contains an option to bypass this regulator and instead use an externally supplied 1.25V input.

An additonal regulator for the flash block is also within the PMC to provide the 3.3V source necessary to the FLASH module.

Various voltage monitors also reside within the PMC, allowing the visibility of internal regualting points and external supplies. Other functional blocks such as the Power On reset (POR) are also available.

1.26 PMC register values defined by DCF clients

The following PMC registers are defined by the corresponding DCF clients:

- **PMC_REE_TD**: DCF_PMC_REE_TD DCF client
- **PMC_REE**: DCF_PMC_REE_CTRL DCF client
- **PMC_RES**: DCF_PMC_RES_CTRL DCF client

Refer to Table 6.5, DCF clients.

NOTE

It is recommended to verify the PMC_REE register value against the DCF_PMC_REE_CTRL DCF client value after any reset. If the values do not match, then write the DCF_PMC_REE_CTRL DCF client value into PMC_REE register.

1.27 Temperature Event Status register (PMC_ESR_TD)

Added the following Note to the PMC_ESR_TD register field descriptions for TEMP1_2 and TEMP0_2:

This temperature sensor status flag might become set when the ADC supply varies below its LVD level and its LVD destructive reset is masked.

1.28 LVD-HVD self test mechanism

The PMC implements an LVD/HVD self test mechanism for use in satisfying safety goals. The digital block assists and controls the analog block during its performance of the self test.

Currently, three self test modes (described in the following sections) are supported by the analog block (based on the mode specified by the PMC_SELF_TEST_UM_VD_REG[ST_MODE] register bit).

The self test status and result are provided in the PMC_SELF_TEST_UM_VD_REG[ST_DONE, ST_RESULT] register fields, respectively.

1.28.1 Default mode

The self test is always performed during start up (i.e power on). The self test starts during the phase 3 RGM state, when the SSCM has completed reads from the Flash memory. A counter is provided that starts from zero and increments by one after an 8 µs time gap. The counter value is the source for the output “pmc_lvd_self_test” signal, which is provided to the analog block.

Meanwhile, the LVD/HVD indication from the analog block is latched in the PMC_SELF_TEST_UM_VD_REG register. At the end of the self test (i.e. when all of the LVDs/HVDs have been tested), the PMC_SELF_TEST_UM_VD_REG[ST_DONE] status bit is set to '1' and all of the bits in the PMC_SELF_TEST_UM_VD_REG register are checked for ‘1’. One or more bits not set to '1' generates a fault indication to the FCCU block, and the PMC_SELF_TEST_UM_VD_REG[ST_RESULT] register bit is set to ‘0’. In the case where all of the register bits are set to '1', the test is considered to be PASSED, the PMC_SELF_TEST_UM_VD_REG[ST_RESULT] register bit is set to ‘1’, and no fault indications are provided to the FCCU.

This startup self test function also provides gating of the reset phase3 exit in RGM. This means the RGM will not be able to exit from phase3 until the self test is successfully completed.

NOTE

The total run time of the LVD self test is equal to: ‘time required per LVD (20 µs)’ times 'number of LVDs (23)' = 460 µs.

1.28.2 Software-triggered self test

This mode is entered by software by writing '01' to the PMC_SELF_TEST_UM_VD_REG[ST_MODE] register 2-bit field. The complete self test, as is performed in default mode, is repeated. The difference is that the self test in this mode is initiated by the host while the self test in default mode is started automatically during power on.

1.28.3 Single voltage detect test

This mode is entered by software writing '10' to the PMC_SELF_TEST_UM_VD_REG[ST_MODE] register 2-bit field, which results in the assertion of the “pmc_lvd_self_test” input. The test completes after the prescribed time gap has elapsed. The PASS/FAIL result is then updated based on the corresponding LVD output tripping, or not.

NOTE

During the running of the LVD self test, all of the LVD and HVD outputs are masked including some PORs that are also under self test. Therefore, they will have no effect on POR generation, or in any reset and/or fault generation.

1.29 Core Status (MC_ME_CS)

Updated the field names in the "MC_ME_CS field descriptions" table to match the implementation.

MC_ME_CS field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 S_COREC	Core Status - This bit specifies the current status of an additional core which is controlled by the MC_ME. 0 Core is disabled 1 Core is running
30 S_CORE0	Core Status - This bit specifies the current status of an additional core which is controlled by the MC_ME. 0 Core is disabled 1 Core is running
31 S_CORE1	Core Status - This bit specifies the current status of an additional core which is controlled by the MC_ME. 0 Core is disabled 1 Core is running

Figure 1-5. MC_ME_CS field descriptions

1.30 Module Configuration Register (JTAGM_MCR)

The following note was added to this section.

NOTE

An unexpected interrupt will occur if the Idle Interrupt is enabled after the end of frame transfer. The interrupt occurs after a JTAGM frame has completed when the JTAGM Idle Interrupt was not enabled (JTAGM_MCR[IIE] = 0b0), if the JTAGM module is then configured to enable interrupts (JTAGM_MCR[IIE] = 0b1). To clear any previous interrupts, set the Idle bit (JTAGM_SR[Idle]) before enabling a JTAGM idle interrupt (e.g. setting JTAGM_MCR[IIE] = 0b1).

1.31 Module Configuration Register (JTAGM_MCR)

Changed the MCR[TCKSEL] 010 field value to Reserved.

27–29 TCKSEL	TCK clock frequency selection. When the JTAGM is enabled, this bit should not be programmed with 000. 000 Reserved 001 Reserved 010 Reserved 011 TCK is system clock ÷ 4 100 TCK is system clock ÷ 5 101 TCK is system clock ÷ 6 110 TCK is system clock ÷ 7 111 TCK is system clock ÷ 8

Figure 1-6. JTAGM_MCR field descriptions

1.32 Data Out Register 0/1/2/3 (JTAGM_DOR0/1/2/3)

Added the stipulation that writing to these registers when DTM = 0 may cause unintended results. Updated the register field descriptions as shown below.

Data Out Register 0 (JTAGM_DOR0)

- TMS_HIGH: Higher word of data for TMS, bits 59-28. Writeable by software only when DTM = 1. Writing to this register when DTM = 0 may cause unintended results.

Data Out Register 1 (JTAGM_DOR1)

- TMS_LOW: Lower word of data for TMS, bits 27-0. Writeable by software only when DTM = 1. Writing to this register when DTM = 0 may cause unintended results. TMS_LOW[0] is shifted out first.

Data Out Register 2 (JTAGM_DOR2)

- TDI_HIGH: Higher word of data for TDI, bits 59-28. Writeable by software only when DTM = 1. Writing to this register when DTM = 0 may cause unintended results.

Data Out Register 3 (JTAGM_DOR3)

- TDI_LOW: Lower word of data for TDI, bits 27-0. Writeable by software only when DTM = 1. Writing to this register when DTM = 0 may cause unintended results. TDI_LOW[0] is shifted out first.
- Send: Send bit. When this bit is set, data is sent to the DCI. It is a self-clearing bit and is always read as 0. Writeable by software only when DTM = 1. Writing to this register when DTM = 0 may cause unintended results.

1.33 Concurrent overflow register (OFLW) implementation

Table 1-15. Corresponding error sources for system RAM OFLW0 register

Bit number	Error source
3	Core0 Instruction TCM error (Note: This error source could be triggered if Core0 IMEM is not initialized before Core0 is started)
9	Core1 Instruction TCM error (Note: This error source could be triggered after reset)

1.34 FCCU chapter

See Chapter 3, Fault Collection and Control Unit (FCCU) for an updated FCCU chapter, and Appendix A for a listing of all the changes in that chapter.

1.35 AUTOLOCK_VALUE for register write access and DCF_COMPLETION value for DCF write completion

Register write-access watchdog timer refers to two clock-cycle values related to the STCU2's write-access watchdog. The following table provides these values on this chip.

Clock-cycle value	Number of STCU2 clock cycles
AUTOLOCK_VALUE	1024
DCF_COMPLETION	16

1.36 Password and Device Security Module (PASS) chip-specific module information

The following table indicates byte-to-byte correspondence between UTEST DCF address and JTAG_PASSWORD.

Table 1-16. JTAG_PASSWORD register bit correspondence

UTEST Address Offset	Register Bit Position (MSB is 255)
0x00	JTAG_PASSWORD[7:0]
0x01	JTAG_PASSWORD[15:8]
...	...
0x1E	JTAG_PASSWORD[247:240]
0x1F	JTAG_PASSWORD[255:248]

1.37 Flash erase counter and tamper detection

The following content has been added to this section, before the first note:

When attempting to update the TDM diary and erase flash blocks, set MCR[PECIE] in the Embedded Flash Memory registers to enable interrupt notification for program and erase operations. Then clear MCR[PECIE] to service the interrupt after each program or erase operation. This sequence must be executed before attempting to update the TDM diary, and then again before attempting to erase the unlocked flash blocks.

Chapter 2
Enhanced Time Processing Unit (eTPU)

2.1 Enhanced Time Processing Unit (eTPU) chip-specific module information

2.1.1 Configuration

eTPU is an intelligent, semi-autonomous co-processor designed for timing control. Operating in parallel with the Host CPU, the eTPU processes instructions, real-time input events, performs output waveform generation, and accesses shared data without Host intervention. Consequently, for each timer event, the Host CPU setup and service times are minimized or eliminated.

The eTPU2+ is an enhanced version of the eTPU2 targeted at C55 devices.

2.1.2 Device-specific information

- Dual (64 channels, dual core) Engine
- Shared Code Memory (SCM) error detection, error correction
- Shared Data Memory (SDM) error detection, error correction
- Class 3 Nexus support
- GTBE is a logical OR with the eMIOS GTBE
- Channels used to trigger the ADC subsystem
- Channels used to enable the Reaction Module modulation
- Channels with DMA request
- Channel output buffers are controlled through SIUL2, and cannot be controlled with microcode.
- Six latency monitors in each of 2 engines.

2.1.3 Register reset values

Some eTPU registers on this device have a different reset value, which overrides the corresponding reset value that is shown in the body of the eTPU documentation, as shown in the following table.

Table 2-1. Reset value overrides

Register	Reset value override
eTPU_SCR_MCR	000B_0000h
eTPU_SCR_ECR_ENG1	5000_0000h
eTPU_SCR_ECR_ENG2	5000_0000h
eTPU_TBR_TBCR_ENG1	2000_0000h
eTPU_TBR_TBCR_ENG2	2000_0000h
eTPU_TBR_STAC_ENG1	0000_0200h
eTPU_TBR_STAC_ENG2	0100_0300h

2.1.4 Memory

The eTPU is a true Harvard architecture machine that implements separate code and data memories. The eTPU engines share the same code and data memories. To meet the safety goals, the memories have ECC.

Memory characteristics:

- Code memory 24k with ECC. The code memory can only be accessed with 32-bit accesses so only 7 bits of ECC are implemented. The module size is 6144 locations x 39-bits wide.
- Data memory 6k with ECC. The data memory can be accessed as bytes, and cycles for updating the ECC are not allowed, so 20 bits of ECC are implemented. The required module size is 1536 locations x 52 bits wide.

2.1.5 Clocking

The eTPU is clocked by the non-FM clock used for the peripherals.

The eTPU BIU can optionally run at double the clock speed of the other peripheral on the AIPS. Double clock speed can be used in cases where higher eTPU processing throughput is needed.

2.1.6 Debug

The eTPU implements Nexus class 3 debug, accessed through the same Nexus debug port pins used for core debug.

2.1.7 STAC bus

The eTPU timebases and angle counter can be exported to eMIOS through the STAC bus.

Table 2-2. STAC server ID

Engine	Timebase	Server ID
1	TCR1	0
1	TCR2	2
2	TCR1	1
2	TCR2	3

NOTE

If the STAC-bus Export/Import feature is used, the clock ratio of eTPU : eMIOS should be 1:1 or 2:1, such as the following:

- eTPU 100MHz : eMIOS 100MHz
- eTPU 200MHz : eMIOS 100MHz

This applies whether the STAC-bus Export/Import is used from eTPU to eMIOS or used between eTPUs.

2.1.8 Input connections

The channel inputs are connected to:

- External pins,
- SD ADC trigger outputs,
- BCTU trigger outputs, and the
- Input glitch filter.

Channels with more than 1 input source are connected through the IOMUX (see SIUL2 chapter).

The TCRCLK input is connected to the same input as channel 0.

2.1.9 Output connections

The channel outputs are connected to:
- External pins
- ADC triggers
- Decimation filters

2.2 Introduction

eTPU is an intelligent, semi-autonomous co-processor designed for timing control. Operating in parallel with the Host CPU, the eTPU processes instructions and real-time input events, performs output waveform generation, and accesses shared data without Host intervention. Consequently, for each timer event, the Host CPU setup and service times are minimized or eliminated.

High-level assembler, compiler and documentation allows customers to develop their own functions on the eTPU.

eTPU architecture aims at high resolution timing capabilities. From a system perspective, high resolution timing is limited by Host CPU overhead required for servicing timing tasks such as period measurement, pulse measurement, pulse width modulated waveform generation, etc. On the eTPU, high resolution timing is achieved by three main capabilities:

- Reduced latency: pin actions are immediate.
- Reduce or eliminate host interrupt service time.
- Double action channel capability reducing the channel request rate.

eTPU provides higher resolution than the Host CPU can achieve and creates no Host overhead for servicing timing tasks.

Latency is the interval from occurrence of an event to the start of event servicing. eTPU can service its own events without interrupting the Host. There are two types of timing events:

- Input pin transition.
- Selected Time Base match, i.e., a selected Time Base counter reached or exceeded a pre-programmed value.

Service time is the time spent servicing an event. In general, in microcontrollers the service time is constrained because the instruction set is not optimized for time function synthesis. The eTPU instruction set is optimized, so that time functions can be implemented with much fewer instructions than the Host CPU. Instructions execute faster, service time is reduced and program memory compacted.

Instructions executed by the eTPU are connected directly to the eTPU timing hardware and allow parallelism of hardware related actions.

2.2.1 Overview

The figure below shows a top-level eTPU block diagram. It displays a dual eTPU Engine configuration.

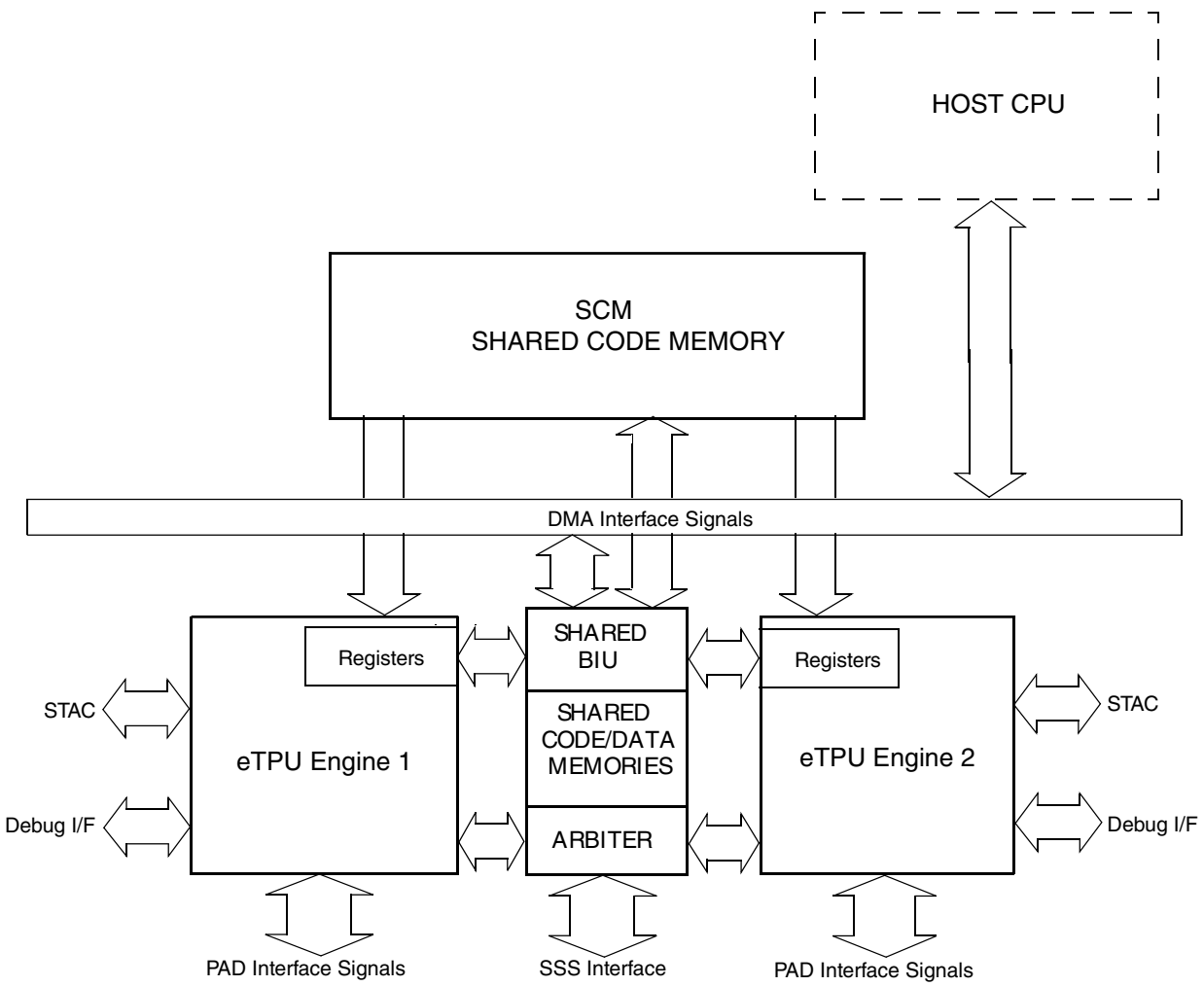


Figure 2-1. eTPU Block Diagram

eTPU Engine is responsible for processing input pin transitions and output pin waveform generation based on the **Time Bases**. Each eTPU Engine has its own microprocessor and dedicated hardware for processing signals on I/O pins and can also interface with external time bases through the STAC bus.

Both eTPU Engine CPUs, hereafter called **microengines**, fetch microinstructions from a **Shared Code Memory - SCM**.

Shared Data Memory - **SDM -** holds eTPU application parameters and work data. It is accessed by Host and both microengines.

Bus Interface Unit - BIU - allows Host to access eTPU registers, SCM and SDM.

Each I/O signal pair is associated with a dedicated **Channel**, which provides hardware for input signal processing and output signal generation, in relationship with selected Time Bases.

The eTPU, as a microprocessed subsystem, works much like a typical real-time system: it runs microengine code from instruction memory (SCM) to handle specific events, accessing data memory (SDM) for parameters, work data and application state info; events may originate from I/O Channels (due to pin transitions and/or time base matches), Host CPU requests or inter-channel requests; events that call for local eTPU processing activate the microengine by issuing a **Service Request**. The Service Request microcode may set an interrupt to the Host CPU. I/O channel events cannot directly interrupt the Host CPU.

Each Channel is associated with a **Function**, which defines its behavior: the Function is a software entity consisting, within the eTPU, of a set of microengine routines that attend to Service Requests. The Function routines are also responsible for Channel configuration. Function routines reside in SCM, which may contain several Functions. A Function may be assigned to several Channels, but a Channel can be associated with just one Function at a given moment. The association between Functions and Channels is defined by Host CPU, and is explained in detail in Functions and threads.

eTPU hardware supplies resource sharing features that support concurrency:

- A hardware **Scheduler** dispatches the Service Request microengine routines based on a set of priorities defined by the Host CPU. Each Channel has its associated priority.
- A Service Request routine cannot be interrupted until it ends. This sequence of uninterrupted instruction execution is called a **Thread**.

- Channel-specific context (registers and flags) is automatically switched between the end of a Thread and the beginning of the next one.
- SDM arbitration, a dual-parameter coherency controller and semaphores can be used to ensure coherent access to eTPU data shared by both eTPU Engines and Host CPU.

2.2.1.1 eTPU Engine

Each eTPU Engine consists of all blocks that are duplicated in a dual eTPU configuration: two 24-bit time bases, 32 independent timer channels, a task scheduler, a microengine, and a Host interface. In addition, a 32-bit Shared Data Memory (SDM) is used for two eTPU Engines data storage and for passing information between the eTPU Engines and the Host CPU.

The following figure shows the block diagram for the eTPU Engine.

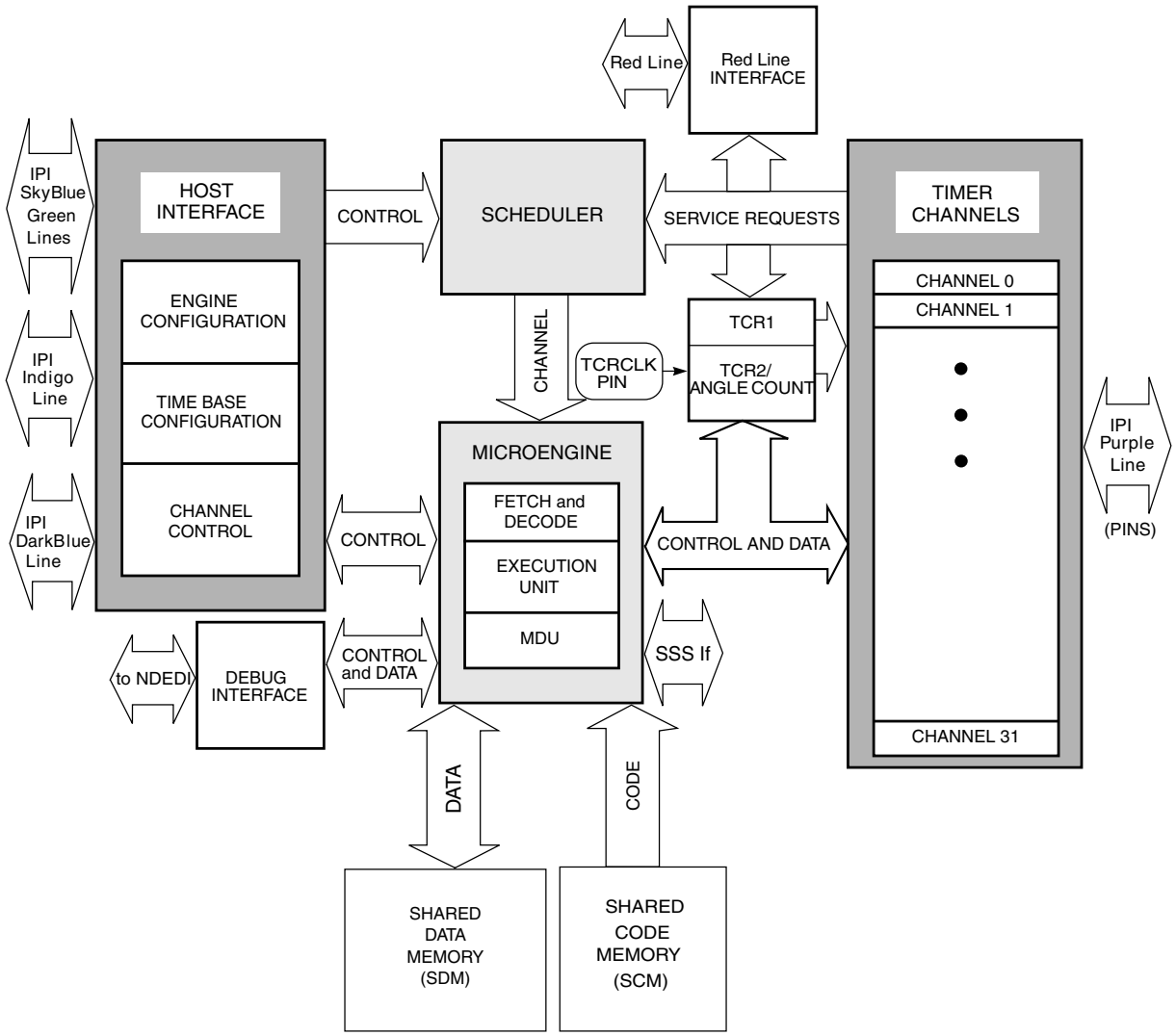


Figure 2-2. eTPU Engine Block Diagram

eTPU Engines 1 and 2 are sometimes called eTPU 1 and eTPU 2 for short throughout this chapter.

2.2.1.1.1 Time Bases

Two 24-bit counters TCR1 and TCR2 provide reference time bases for all match and input capture events. Prescalers for both time bases are controlled by the Host CPU through bit fields in the eTPU Engine configuration registers. The eTPU is able to export/import time to/from TCR1 or TCR2 in accordance to the STAC bus specification.

The clock for each of TCR1 and TCR2 clock can be independently derived from the eTPU clock or from an external input via the TCRCLK clock pin. In addition, the TCR2 timebase can be derived from special angle-clock hardware which enables implementing angle-based functions. This feature is added to support advanced angle based engine control applications.

For further details refer to Time bases.

2.2.1.1.2 eTPU Timer Channels

Each eTPU Engine has 32 independent channels, each corresponding to an Input/Output signal pair. The channel time resolution is 24 bits, and are all identical.

Each channel consists of logic which supports two events and output controls. The event logic contains two 24-bit capture registers, two 24-bit match registers, greater-equal and equal-only comparators. Supporting two events enables many combinations of double action functions (for example the channel can handle two events with a single microcode service).

The channel configuration can be changed immediately by the microengine. Each channel can perform double capture, double match and other capture-match combinations. Channel modes available can do ordered or unordered match. Some modes are also provided that can block one match by the occurrence of the other. Service request can be generated on one or both of the match events.

Input signal can be separated from output signal in each channel. They can, optionally, be combined in a single I/O pin driver. An output buffer enable is controlled by the SIU (or SIUL). Digital filters are provided for the input signals, with distinct filtering modes available.

Each channel can use any time base or angle counter for either match or capture operation. For example, a match on TCR1 can capture the value of TCR2. The channels can request service from the microengine due to recognized pin transitions (input events) or timebase matches.

The eTPU channels also support the basic single-action operations found on TPU3 functionality with the exception that time resolution is 24-bits.

Channel configuration combinations:

- Single input capture, no match (TPU3 functionality)
- Single input capture with single match timeout (TPU3 functionality)
- Single input capture with double match timeout with several double match sub-modes
- Double input capture with single or double match timeout with several double match sub-modes
- Single output match (TPU3 functionality)

- Double output match with several double match sub-modes
- Input-dependent output generation

The double match functionality has various combinations for generation of service request and determining pin actions. For more details refer to Enhanced Channels.

In addition to the predefined channel configurations above, the user can also program its own channel configuration, defining how input captures, matches and service-requests are related.

2.2.1.1.3 Host interface

The Host interface allows the Host CPU to control the operation of the eTPU. The Host CPU must initialize the eTPU by writing to the appropriate Host interface registers to assign a Function and priority to each channel. In addition, the Host writes to the Host Service Request and channel configuration registers to further define Function operation for each initialized channel. Refer to Host interface for a detailed description.

When the SCM is implemented by RAM, the Host must first initialize it with the proper microcode program prior to enabling any eTPU Function, and then enable eTPU access (which also disables Host access).

2.2.1.1.4 Shared Data Memory - SDM

The SDM works as data RAM which can be accessed by the Host CPU and up to two eTPU Engines. This memory is used for information transfer between the Host CPU and the eTPU, as data storage for the eTPU microcode program or for communication between the two eTPU Engines. SDM width is 32 bits, and is accessible by the Host as byte, 16-bit or 32-bit wide. eTPU can access it as full 32-bits, lower 24-bits or upper byte (8-bit).

The host can also access the SDM space mirrored in other area with Parameter Sign Extension (PSE). Parameter Sign Extension accesses differ from the usual host accesses to the original SDM area as follows:

- Writes are effective only to the lower 3 bytes of a word: the word's most significant byte is kept unaltered in SDM.
- Reads return the lower 3 bytes of a word sign-extended to 32 bits, i.e.: the most significant bit of the word's 2nd most significant byte is copied in all 8 bits of the most significant read byte.

Each eTPU channel can be associated with a variable number of parameters located in the SDM, according to its selected Function. In addition, the SDM can be fully shared between two eTPU Engines, enabling direct communication between them.

High flexibility of the SDM utilization is achieved as follows:

- Each channel has a programmable base address pointing to the address of its first parameter with two parameter granularity. This way the SDM can be partitioned according to the actual function needs.
- The microcode can access the first 128 parameters of the selected channel in channel relative access mode.
- Each Engine can access all the SDM address space in indirect addressing mode. Blocks of data are easily transferred using stack operation.
- Absolute addressing mode can access the first 256 parameters (TPU3 functionality), implementing a shared pool of parameters holding global variables.

In the Host address space each parameter occupies four bytes. eTPU usage of the upper byte is achieved by having a 32-bit P register which can access the upper byte, the lower 24 bits or all the 32 bits. The microcode can switch between access sizes at any time.

Each Function may require a different number of parameters. During the eTPU initialization the Host has to program channel base addresses, allocating proper parameters for each channel according to its selected Function.

2.2.1.1.5 Shared Subsystem Area

The Shared Subsystem (SSS) area is a memory map region used by both the host and the eTPU to access registers of the reaction channel, giving the user the option to control them by the host, the eTPU code, or a combination of both.

2.2.1.1.6 Scheduler

Out of reset, all channels are disabled. The Host CPU makes a channel active by assigning it one of three priorities: high, middle, or low. The Scheduler determines the order in which channels are serviced based on channel number and assigned priority. The priority mechanism, implemented in hardware, ensures that all requesting channels are serviced. For additional details refer to Scheduler.

2.2.1.1.7 Microengine

eTPU microengine is a simple RISC implementation which performs each instruction in a microcycle of two eTPU clocks, while prefetching the next instruction through an instruction pipeline. Instruction execution time is constant unless it gets wait states from the SDM arbitration. Two eTPU Engines share code memory without having any performance degradation by interleaving their accesses (the Shared Code Memory has one-clock access time).

Instruction width is 32 bits. The microengine instruction set provides basic arithmetic and logic operations, flow control (jumps and subroutine calls), SDM access, and Channel configuration and control. The instruction formats are defined in such a way that allow particular combinations of two or three of these operations with unconflicting resources to be executed in parallel in the same microcycle.

Microengine has also an independent Multiply/Divide/MAC unit that performs these complex operations in parallel with other microengine instructions.

Channel functionality is tightly integrated to the instruction set through Channel Control operations and conditional Branch operations, which support jumps/calls on Channel-specific conditions. This allows quick and terse Channel configuration and control code, contributing to reduced service time.

Detailed description can be found in Microengine.

2.2.1.1.8 Dual eTPU Engine System

A typical eTPU implementation includes two eTPU Engines sharing SDM and the same code in SCM.

The two eTPU Engines share the Bus Interface Unit (BIU) and the data memory (SDM) which enable Host-eTPU and eTPU Engine-Engine communication. The shared BIU includes coherency logic which supports dual parameter (8 bytes) coherency in transfers between Host and eTPU, using a temporary parameter area within the SDM. More details on this can be found on Parameter sharing and coherency.

2.2.2 Features

The following sections show eTPU features and enhancements.

2.2.2.1 eTPU Feature Summary

The eTPU includes these distinctive features:

- up to 32 channels for each eTPU Engine, each channel is associated with an Input/ Output signal pair.
  - enhanced input digital filters on the input pins for improved noise immunity. The eTPU digital filter can use 2 samples, 3 samples or work in continuous mode.
  - identical, orthogonal channels, except for channel 0: each channel can perform any time function. Each time function can be assigned to more than one channel at a given time, so each signal can have any functionality. Channel 0 has the same capabilities of the others, but can also work with special Angle Counter logic (see below).
  - Link Service Request allows activation of a Channel function by request of another channel, even between eTPU Engines.
  - Host Service Request allows activation of a Channel function by Host CPU request
  - each channel has an event mechanism which supports single and double action functionality in various combinations. It includes two 24-bit capture registers, two 24-bit match registers, 24-bit greater-equal and equal-only comparators.
- Two independent 24-bit time bases for channel synchronization:
  - first time base clocked by eTPU clock with programmable prescaler division from 1 to 512 (in steps of 2), or by output of second time base prescaler.
  - first time base can also be clocked by external signal with programmable prescaler division of 1 to 256.
  - second time base clocked by external signal with programmable prescaler division from 1 to 64.
  - second time base external clock source can be replaced by eTPU clock divided by 8.
  - both time bases can be exported or imported via Shared Time and Counter bus.
  - second time base counter can work as an Angle counter, enabling angle based applications to match angle instead of time.
  - second time base can also be used as a pulse accumulator gated by external signal.
- Event-Triggered RISC processor (microengine):
  - 3 stage pipeline implementation (fetch, decoding and execution), with separate instruction memory - SCM - and data memory - SDM (Harvard architecture)

- fixed-length instruction execution in one microcycle of two eTPU clocks
- interleaved SCM access in dual eTPU Engine avoids contention in time for instruction memory
- SCM address space of up to 16K positions (64 Kbytes) (see chip-specific eTPU information)
- SDM with interleaved access in dual eTPU Engine avoids contention for data memory
- SDM address space (both Engines) of up to 8 Kbytes (see chip-specific eTPU information)
- instruction set with embedded Channel support, including specialized Channel control subinstructions and conditional branching on Channel-specific flags.
- channel-oriented addressing: channel-bound address mode with Host configured Channel Base Address allows channel data isolation, independent of microengine application code.
- channel-bound data address space of up to 128 32-bit parameters (512 bytes)
- global parameter address mode allows access to common Channel data of up to 256 32-bit parameters (1024 bytes)
- support for indirect and stacked data access schemes.
- parallel execution of: data access, ALU, Channel control and flow control subinstructions in selected combinations.
- 32-bit microengine registers and 24-bit resolution ALU, with 1 microcycle addition and subtraction, absolute value, bitwise logical operations on 24-bit, 16-bit, or byte operands; single bit manipulation, shift and rotate operations, sign extension and conditional execution.
- additional 24-bit Multiply/MAC/Divide unit which supports all signed/unsigned Multiply/MAC combinations, and unsigned 24-bit Divide. The MAC/Divide unit works in parallel with the regular microcode commands.

- Resource sharing features support channel sharing of channel registers, memory and microengine time:
  - hardware Scheduler works as a "task management" unit, dispatching event service routines by predefined, Host-configured priority.

- automatic Channel context switch when a "task switch" occurs, i.e., one Function Thread ends and another begins to service a request from other Channel: Channel-specific registers, flags and parameter base address are automatically loaded for the next serviced channel.
- individual channel priority setting in 3 levels: high, middle and low.
- Scheduler priority scheme allows calculation of worst case latency for event servicing and ensures servicing all channels by preventing permanent blockage.
- SDM shared between Host CPU and both eTPU Engines, supporting communication either between Channels and Host or inter-channel.
- hardware implementation of 4 Semaphores supports resource sharing between both eTPU Engines.
- Hardware semaphores directly supported by the microengine instruction set.
- dual parameter coherency hardware support allows atomic (to host) access to 2 parameters by microengine(s) in back-to-back accesses.
- coherent dual-parameter controller allows atomic (to microengines) accesses to 2 parameters by the host.

- Development support features:
  - Nexus class 3 debug support (optional, associated with the eTPU-Nexus Block NDEDI).
  - Software breakpoints.
  - Debug interface supporting single-step execution, forced microinstruction execution, Hardware breakpoints and watchpoints on several conditions.
- Safety support features:
  - SCM (code memory) continuous signature-check built-in self test (MISC - Multiple Input Signature Calculator), runs concurrently with eTPU normal operation.
  - See eTPU Plus Enhancements over eTPU) for additional safety features.

2.2.2.2 eTPU Enhancements over TPU3

- 32 orthogonal channels with enhanced functionality. Full support for double action with double match and double transition sub-mode combinations.

- Input and Output features separated in channel logic and microinstructions, allowing input and output signals to be processed separately or combined.
- Increased time resolution and execution unit to 24 bits.
- Increased linear code memory, shared by two eTPU Engines, configurable up to 16K positions (64 KB).
- Increased SDM address range (8 KB each Engine) and width (32 bits per parameter). The SDM can be dynamically allocated to support variable number of parameters for each channel. Each channel can have access to at least 256 parameters.
- The SDM is fully shared by two eTPU Engines (SDM), supporting direct inter-engine communication with the help of hardware semaphores.
- Enhanced arithmetic operations, including add/subtract with carry, absolute value, multiple shift and rotate, conditional execution with variable operand widths.
- Enhanced logic operations, including bitwise operations (and, or, xor) and bit manipulation, with conditional execution. Support for read-modify-write of any bit in the SDM.
- Hardware for Multiply/MAC/Divide, running in parallel to execution of other operations. The 24-bit divide result is available after 13 other unrelated instructions. Multiplication supports any data width of both operands (8, 16 or 24 bits), signed or unsigned. A 24x24 Multiply/MAC result is available after four other unrelated instructions. A 24x8 Multiply/MAC result is available after one other unrelated instruction.
- Supports export/import of time bases from other sources through the real time bus (STAC - Shared Time and Counter bus). This internal bus is used for sharing real time data between multiple peripherals.
- Contains angle clock hardware, supported by microcode, which can provide a 24-bit angle bus instead of time bus. This feature enables the eTPU to run angle based engine control applications.
- More interrupt types. Each eTPU channel can generate a data transfer request interrupt, in addition to regular interrupts, and one global exception interrupt. Data Transfer requests can be used either as interrupt sources or DMA requests. This feature takes advantage of DMA peripherals which off-load the Host. Interrupt Overflow status is also provided.
- Improved visibility to the Host (pin states, time bases, serviced channel).

- An edge case of priority inversion on TPU3 Scheduler was resolved.
- Supports channel link requests between eTPU Engines.

2.2.2.3 eTPU Plus Enhancements over eTPU

eTPU Plus features include the enhancements below in addition to all eTPU features

- TCR1, channel logic and digital filters (both channel and TCRCLK) now have an option to run at divisions of full eTPU clock speed, besides eTPU clock / 2 (see T2/T4 channel timing).
- Channels support unordered transitions: transition B can now be detected before transition A. Related to this enhancement, TDLA and TDLB can now be independently negated by microcode (see Channel Mode Logic and Event Flags)
- Channel logic can now work on full eTPU clock, allowing faster response with slow clocks (see T2/T4 channel timing).
- Added a new User Programmable Channel Mode: the blocking, enabling, service request and capture characteristics of this channel mode can be programmed via microcode (see General Channel registers).
- Microinstructions now provide an option to issue Interrupt and Data Transfer requests selected by CHAN. They can also be requested simultaneously at the same instruction (see Channel Interrupt and Data Transfer Requests).
- Channel Flags 0 and 1 can now be tested for branching, besides selecting the entry point (see Branch Operations).
- Channel digital filters can be bypassed, and can be clocked by full eTPU clock (see Bypass Mode).

Backward compatibility is maintained so that all legacy object code (both Host and microcode) runs on eTPU Plus without modification.

2.2.2.4 eTPU2 enhancements over eTPU Plus

eTPU2 features include the enhancements below in addition to all eTPU Plus enhancements and eTPU features.

- Scheduler priority-passing mechanism can now be disabled (see Priority passing disabling).
- New Watchdog mechanism kills threads over a programmable timeout (see Watchdog).

- New counter allows microengine load information collection for performance analysis (see Idle Counter).
- Channels 1 and 2 (besides channel 0) can now be selected to control the EAC (see the TBCR_ENG1 and TBCR_ENG2 eTPU Time Base Configuration Registers).
- Timebase prescalers are now reset when the GTBE input is negated, guaranteeing synchronization with eMIOS in all cases (see GTBE - Global Time Base Enable).
- New MISC flag indicates when an SCM signature calculation round is completed. This allows measuring of the average MISC scan period in a real application situation (see the eTPU Module Configuration Register - MCR).
- New channel TCCEA flag allows continuous capture even after TDLA is set, making it fully compatible with TPU behavior (see TCCEA - Transition Continuous Capture Enable).
- New branch condition PRSS tells the pin state at the time when a channel (match or transition) service request occurred (see Pin Control Registers).
- MRLEA/B can now be negated independently by microcode (see Disable Matches).
- New Engine Relative address mode allows a function to access SDM address space common to one engine, but distinct between engines (see Engine Relative Addressing Mode under SDM Addressing Modes).
- Memory error support for Code (SCM) and Data (SDM) memories: error detection, correction, and hardware-assisted soft-error fixing (see Memory error support).
- Sharing of microengine and host accesses to subsystem modules (for instance, the Reaction Module), allowing their control by the eTPU, the host, or both (see Shared Subsystems support).

Backward compatibility is maintained so legacy object code (both Host and microcode) runs on eTPU2 without modification.

2.2.2.5 eTPU2 Plus enhancements over eTPU2

eTPU2 Plus features include the enhancements listed below in addition to all eTPU features, eTPU Plus enhancements and eTPU2 enhancements.

- New Latency Monitor allows latency measurements, either for individual channels or channels combined in a single priority (High, Medium or Low).
- Watchdog can now be configured to disable thread forced ending and channel blocking, allowing it to monitor thread lengths with no interference on the application.

- EAC Angle Tick Generator can now be clocked by the TCR2 prescaler output, as an alternative to the TCR1 clock.
- EAC now supports up to 7 missing teeth on hardware.

Backward compatibility is maintained so legacy object code (both Host and microcode) runs on eTPU2 Plus without modification.

2.2.3 Modes of operation

eTPU can be seen as capable of working in the following modes:

- **User Configuration Mode**

  User has the ability to program the eTPU Cores with User Time Functions, having access to the Shared Code Memory (SCM).

- **User Mode**

  User does not access the eTPU Shared Code Memory:

  - Use of predefined eTPU Functions
  - No need for eTPU Core programming ability

- **Debug Mode**

  User debugs eTPU code, accessing special Trace/Debug features via Nexus interface:

  - hardware breakpoint/watchpoint setting
  - access to internal registers
  - single-step execution
  - forced instruction execution
  - software breakpoint insertion and removal.

- **Module Disable Mode**

  eTPU Engine clocks are stopped through a register write to the MDIS field in eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2), saving power. Input sampling stops. eTPU Engines can be in Module Disable Mode independently. Module Disable Mode stops only the Engine clock, so that the Shared BIU, and Global

Channel registers can be accessed, and interrupts and DMAs can be cleared and enabled/disabled. An Engine only enters Module Disable Mode when any currently running thread is finished (see Functions and threads).

- **Stop Mode**

  Stop Mode is entered when eTPU answers device stop request assertion with stop acknowledge. The definition of which clocks are stopped is made at the MCU level, which defines whether or not registers can be accessed, interrupts and DMA requests cleared.

These modes are loosely selected: there is no unique register field or signals to choose between them. Some features of one mode can be used with features of other mode(s). More on this subject can be found on eTPU mode selection below.

Note

Throughout this chapter, an engine is said to be "stopped" if it is either in Module Disable mode or Stop mode.

2.2.3.1 eTPU mode selection

User and User Configuration are the production operating modes, and differ from each other only in access to SCM. User programmability is only possible with a RAM SCM.

On chips where the SCM is implemented as a RAM, it can either be accessed directly from IP-Bus for code loading, or for software breakpoint setting . On chips with a ROM SCM, an internal SCM Emulation RAM may be used, depending on the specific MCU implementation, to replace ROM SCM for test or debug purposes. SCM Emulation RAM is selected in a MCU-specific way. For more details, see .

Debug Mode is characterized by the use of the debug interface features. Specifically, Nexus blocks provide Nexus class 3 debug features. For more information on debug features, refer to Development support features.

Module Disable Mode is entered by setting the MDIS field in eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2). eTPU Engines can be individually stopped going into Module Disable Mode (there is one eTPU_SCR_ECR register for each Engine). Each engine can leave Module Disable Mode by writing MDIS=0 (which can only be done if VIS=0).

Stop Mode is activated by IP-Bus (device stop request). In this case, the eTPU waits for both eTPU Engines to enter in stop mode, and then asserts the stop acknowledge line. eTPU leaves Stop Mode when device stop request is negated, but only if VIS=0. If device stop request is negated and VIS=1, eTPU will leave Stop Mode as soon as VIS=0.

Note

An Engine can stay in Module Disable mode when it leaves Stop Mode if its bit MDIS=1, even if the other leaves it.

2.3 Detailed signal descriptions

Detailed signal descriptions are presented in the following sections.

2.3.1 eTPU channel output signals

Each channel output signal is associated with a channel. The microcode may affect the logic level of an output signal[1] by implementing one of two actions:

- Specify the logic level output to the signal when there is a match or a transition.
- Immediately force a logic level.

The output signal may also be forced to a logic level, independently of the output value from the channel logic, by one of the four (each Engine) output disable input signals (see eTPU channel output disable signals).

The output buffer enable signals are not used.

2.3.2 eTPU channel input signals

Each channel input signal is associated with a channel. The microcode can directly control the effect of the transition edge. Each channel can be programmed to sense a transition when a rising and/or falling edge is detected. The channel logic can also process two transition events, and relate these events to each other and to other programmed timer events. The edge sensitivities of the two transition events are configured independently by microcode. For further information refer to Enhanced Channels and Transition Detection and Pin Action Control.

1. Note that the minimum pulse width is one microcycle (two eTPU clocks), and slow pads may not be able to transfer it on time. For generation of very short pulses the eTPU pads have to be programmed by the system integration for fast operation mode with the voltage levels defined for fast pad operation in the MCU technology.

Each channel input signal has an associated synchronizer made of two flip-flops sampling the signal on every other eTPU clock[2], followed by a digital filter. This digital filter can work in three sub-modes, whose purpose is to filter out noise pulses that have width less then a programmed value of eTPU clocks, preventing these transitions from being input to the transition detect logic. The synchronizer and digital filter are guaranteed to pass pulses that are greater than a programmed value. All channel input filters in one Engine work on the same mode and sampling clock. For more information on channel input filters, refer to Enhanced Digital Filter - EDF . In one of the Angle Modes, the output of the digital filter of channel 0 is replaced by the output of TCRCLK signal digital filter (see EAC - eTPU Angle Counter).

2.3.3 Time Base Clock Signal — TCRCLK

TCRCLK is an input signal used to control the Time Bases TCR1 and TCR2. There is one independent TCRCLK input for each Engine. For pulse accumulator operations TCRCLK can be used as a gate for a counter based on the eTPU clock divided by eight. For Angle operations TCRCLK can be used to get the tooth transition indications in Angle Mode. Further details can be found on Time bases and EAC - eTPU Angle Counter.

Like the channel input signals, the TCRCLK signal has an associated synchronizer followed by a digital filter. This digital filter can work in two sub-modes, whose purpose is to filter out noise pulses that have width less then a programmed value of eTPU clocks, preventing these transitions from being input to the transition detect logic. The synchronizer and digital filter are guaranteed to pass pulses that are greater than a programmed value. The clock and operation sub-mode of the TCRCLK filter is configured independently of the other channel input filters, through the TCRCF field in eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1) and eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2). For more information on filter sub-modes, see TCRCLK digital filter. In one of the Angle Modes, the output of the digital filter of channel 0 is replaced with the output of TCRCLK signal digital filter (see EAC - eTPU Angle Counter).

2.3.4 eTPU channel output disable signals

Each of these 4 input signals are used to force the outputs of a group of 8 channels to an inactive level. When an ODIS input is active, all the channels in its group of 8 that have their fields ODIS=1 in eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2

2. Sampled on the T4 microcycle phase, see Microcycle and I/O timing.

(eTPU_CCCR_C*n*CR_ENG2) have their outputs forced to the opposite of the value specified in bit OPOL of the same register. Therefore, channels can be individually selected to be affected by the output disable signals, as well as their disabling forced polarity (see 'Pin State Input/Output Logic' figure in Pin Control Registers).

See the chip-specific eTPU information for details on the output channel disable group definition.

In a Dual-engine eTPU there are 8 output disable signals for the 64 channels.

2.4 Memory map/register definition

The guideline for the description of all bits and fields throughout the“Memory Map/ Register Definition” section is to provide only a brief explanation (without examples or method of use) of the features, since it will be used mainly as a reference for the reader that is studying the “Functional Description" section, where those features are explained in detail.

2.5 Memory map

The eTPU System simplified memory map is shown in Table 2-3. Each of the register areas shown may have their own reserved address areas.

Table 2-4 shows a detailed memory map. Offsets are relative to the eTPU Base address, which is MCU-dependent.

See the chip-specific eTPU information about accessing eTPU registers in chip-level Stop mode.

SCM unused area is decoded and returns a fixed opcode, determined by eTPU SCM Off-range Data Register (eTPU_SCR_SCMOFFDATAR).

Table 2-3. High-level memory map

Offset	Use
0x00 – 0x1F	System Configuration Registers
0x20 – 0x2F	eTPU 1 Time Base Registers
0x30 – 0x3F	Reserved[1]
0x40 – 0x4F	eTPU 2 Time Base Registers
0x50 – 0x5F	Reserved[1]
0x60 – 0x6F	eTPU 1 Extra Engine Registers
0x70 – 0x7F	eTPU 2 Extra Engine Registers

Table continues on the next page...

Table 2-3. High-level memory map (continued)

Offset	Use
0x80 – 0xFF	Latency Monitor Registers
0x100 – 0x13F	Memory Error Support Registers
0x140 – 0x1FF	Reserved[1]
0x200 – 0x2FF	eTPU 1/2 Global Channel Registers
0x300 – 0x3FF	Reserved[1]
0x400 – 0x7FF	eTPU 1 Channel Registers
0x800 – 0xBFF	eTPU 2 Channel Registers
0xC00 – 0xFFF	Shared Subsystems Integration Registers
0x1000 – 0x3FFF	Reserved[1]
0x4000 – 0x7FFF	SSS - Shared Subsystems area
0x8000 – 0xBFFF[2]	SDM - Shared Data Memory
0xC000 – 0xFFFF[2]	SDM PSE mirror[3]
0x10000 – 0x1FFFF[2]	SCM - Shared Code Memory[4]

1. Reserved addresses must not be used. Access to these memory positions complete with 0-wait-states, but the result is unspecified.
2. Actual sizes of SCM and SDM are MCU-dependent (see the chip-specific eTPU information).
3. Parameter Sign Extension access area, see Parameter Access.
4. SCM access is available only when bit VIS=1 on register eTPU_SCR_MCR, under certain conditions (see MCR).

Table 2-4. Detailed memory map

Offset	Use
System Configuration Registers	
0x00	eTPU_SCR_MCR - eTPU Module Configuration Register
0x04	eTPU_SCR_CDCR - eTPU Coherent Dual-Parameter Controller Register
0x08	Reserved
0x0C	eTPU_SCR_MISCCMPR - eTPU MISC Compare Register
0x10	eTPU_SCR_SCMOFFDATAR - eTPU SCM Off-range Data Register[1]
0x14	eTPU_SCR_ECR_ENG1 - eTPU 1 Engine Configuration Register
0x18	eTPU_SCR_ECR_ENG2 - eTPU 2 Engine Configuration Register
0x1C	Reserved
Time Base Registers	
0x20	eTPU_TBR_TBCR_ENG1 - eTPU 1 Time Base Configuration Register
0x24	eTPU_TBR_TB1R_ENG1 - eTPU 1 Time Base 1
0x28	eTPU_TBR_TB2R_ENG1 - eTPU 1 Time Base 2
0x2C	eTPU_TBR_STAC_ENG1 - eTPU 1 STAC Configuration Register
0x30	Reserved
0x34	Reserved
0x38	Reserved
0x3C	Reserved
0x40	eTPU_TBR_TBCR_ENG2 - eTPU 2 Time Base Configuration Register

Table continues on the next page...

Table 2-4. Detailed memory map (continued)

Offset	Use
0x44	eTPU_TBR_TB1R_ENG2 - eTPU 2 Time Base 1
0x48	eTPU_TBR_TB2R_ENG2 - eTPU 2 Time Base 2
0x4C	eTPU_TBR_STAC_ENG2 - eTPU 2 STAC Configuration Register
0x50	Reserved
0x54	Reserved
0x58	Reserved
0x5C	Reserved
Engine Related Registers	
0x60	eTPU_ER_WDTR_ENG1 - eTPU 1 Watchdog Timer Register
0x64	Reserved
0x68	eTPU_ER_IDLER_ENG1 - eTPU 1 Idle Counter Register
0x6C	Reserved
0x70	eTPU_ER_WDTR_ENG2 - eTPU 2 Watchdog Timer Register
0x74	Reserved
0x78	eTPU_ER_IDLER_ENG2 - eTPU 2 Idle Counter Register
0x7C	Reserved
Latency Monitor Registers	
0x80	eTPU_LMR_LM0CR_ENG1 - eTPU 1 Latency Monitor 0 Configuration Register
0x84	eTPU_LMR_LM0MR_ENG1 - eTPU 1 Latency Monitor 0 Measure Register
0x88	eTPU_LMR_LM1CR_ENG1 - eTPU 1 Latency Monitor 1 Configuration Register
0x8C	eTPU_LMR_LM1MR_ENG1 - eTPU 1 Latency Monitor 1 Measure Register
0x90	eTPU_LMR_LM1CR_ENG1 - eTPU 1 Latency Monitor 2 Configuration Register
0x94	eTPU_LMR_LM1MR_ENG1 - eTPU 1 Latency Monitor 2 Measure Register
0x98	eTPU_LMR_LM1CR_ENG1 - eTPU 1 Latency Monitor 3 Configuration Register
0x9C	eTPU_LMR_LM1MR_ENG1 - eTPU 1 Latency Monitor 3 Measure Register
0xA0	eTPU_LMR_LM1CR_ENG1 - eTPU 1 Latency Monitor 4 Configuration Register
0xA4	eTPU_LMR_LM1MR_ENG1 - eTPU 1 Latency Monitor 4 Measure Register
0xA8	eTPU_LMR_LM1CR_ENG1 - eTPU 1 Latency Monitor 5 Configuration Register
0xAC	eTPU_LMR_LM1MR_ENG1 - eTPU 1 Latency Monitor 5 Measure Register
0xC0	eTPU_LMR_LM0CR_ENG2 - eTPU 2 Latency Monitor 0 Configuration Register
0xC4	eTPU_LMR_LM0MR_ENG2 - eTPU 2 Latency Monitor 0 Measure Register
0xC8	eTPU_LMR_LM1CR_ENG2 - eTPU 2 Latency Monitor 1 Configuration Register
0xCC	eTPU_LMR_LM1MR_ENG2 - eTPU 2 Latency Monitor 1 Measure Register
0xD0	eTPU_LMR_LM1CR_ENG2 - eTPU 2 Latency Monitor 2 Configuration Register
0xD4	eTPU_LMR_LM1MR_ENG2 - eTPU 2 Latency Monitor 2 Measure Register
0xD8	eTPU_LMR_LM1CR_ENG2 - eTPU 2 Latency Monitor 3 Configuration Register
0xDC	eTPU_LMR_LM1MR_ENG2 - eTPU 2 Latency Monitor 3 Measure Register
0xE0	eTPU_LMR_LM1CR_ENG2 - eTPU 2 Latency Monitor 4 Configuration Register
0xE4	eTPU_LMR_LM1MR_ENG2 - eTPU 2 Latency Monitor 4 Measure Register

Table continues on the next page...

Table 2-4. Detailed memory map (continued)

Offset	Use
0xE8	eTPU_LMR_LM1CR_ENG2 - eTPU 2 Latency Monitor 5 Configuration Register
0xEC	eTPU_LMR_LM1MR_ENG2 - eTPU 2 Latency Monitor 5 Measure Register
Memory Error Support Registers[2]	
0x100	eTPU_MESR_MECR - eTPU Memory Error Control Register
0x104	eTPU_MESR_DEIAR - eTPU Data Error Injection Address Register
0x108	eTPU_MESR_DEIDPR - eTPU Data Error Injection Data Pattern Register
0x10C	eTPU_MESR_DEIPPR - eTPU Data Error Injection Parity Pattern Register
0x110	eTPU_MESR_DERAR - eTPU Data Error Report Address Register
0x114	eTPU_MESR_DERDR - eTPU Data Error Report Data Register
0x118	eTPU_MESR_DERSR - eTPU Data Error Report Syndrome Register
0x11C	Reserved
0x120	eTPU_MESR_MESR - eTPU Memory Error Status Register
0x124	eTPU_MESR_CEIAR - eTPU Code Error Injection Address Register
0x128	eTPU_MESR_CEIDPR - eTPU Code Error Injection Data Pattern Register
0x12C	eTPU_MESR_CEIPPR - eTPU Code Error Injection Parity Pattern Register
0x130	eTPU_MESR_CERAR - eTPU Code Error Report Address Register
0x134	eTPU_MESR_CERDR - eTPU Code Error Report Data Register
0x138	eTPU_MESR_CERSR - eTPU Code Error Report Syndrome Register
0x13C	eTPU_MESR_CEFR - eTPU Code Error Fix Register
0x140 - 0x1FF	Reserved
Global Channel Registers	
0x200	eTPU_GCR_CISR_ENG1 - eTPU 1 Channel Interrupt Status Register
0x204	eTPU_GCR_CISR_ENG2 - eTPU 2 Channel Interrupt Status Register
0x208	Reserved
0x20C	Reserved
0x210	eTPU_GCR_CDTRSR_ENG1 - eTPU 1 Channel Data Transfer Request Status Register
0x214	eTPU_GCR_CDTRSR_ENG2 - eTPU 2 Channel Data Transfer Request Status Register
0x218	Reserved
0x21C	Reserved
0x220	eTPU_GCR_CIOSR_ENG1 - eTPU 1 Channel Interrupt Overflow Status Register
0x224	eTPU_GCR_CIOSR_ENG2 - eTPU 2 Channel Interrupt Overflow Status Register
0x228	Reserved
0x22C	Reserved
0x230	eTPU_GCR_CDTROSR_ENG1 - eTPU 1 Channel Data Transfer Request Overflow Status Register
0x234	eTPU_GCR_CDTROSR_ENG2 - eTPU 2 Channel Data Transfer Request Overflow Status Register
0x238	Reserved
0x23C	Reserved
0x240	eTPU_GCR_CIER_ENG1 - eTPU 1 Channel Interrupt Enable Register
0x244	eTPU_GCR_CIER_ENG2 - eTPU 2 Channel Interrupt Enable Register

Table continues on the next page...

Table 2-4. Detailed memory map (continued)

Offset	Use
0x248	Reserved
0x24C	Reserved
0x250	eTPU_GCR_CDTRER_ENG1 - eTPU 1 Channel Data Transfer Request Enable Register
0x254	eTPU_GCR_CDTRER_ENG2 - eTPU 2 Channel Data Transfer Request Enable Register
0x258	Reserved
0x25C	Reserved
0x260	eTPU_GCR_WDSR_ENG1 - eTPU 1 Watchdog Status Register
0x264	eTPU_GCR_WDSR_ENG2 - eTPU 2 Watchdog Status Register
0x268-0x27F	Reserved
0x280	eTPU_GCR_CPSSR_ENG1 - eTPU 1 Channel Pending Service Status Register
0x284	eTPU_GCR_CPSSR_ENG2 - eTPU 2 Channel Pending Service Status Register
0x288	Reserved
0x28C	Reserved
0x290	eTPU_GCR_CSSR_ENG1 - eTPU 1 Channel Service Status Register
0x294	eTPU_GCR_CSSR_ENG2 - eTPU 2 Channel Service Status Register
0x298	Reserved
0x29C	Reserved
0x300 - 0x3FF	Reserved
Channel Configuration and Control Registers	
0x400	eTPU_CCCR_C0CR_ENG1 - eTPU 1 Channel 0 Configuration Register
0x404	eTPU_CCCR_C0SCR_ENG1 - eTPU 1 Channel 0 Status and Control Register
0x408	eTPU_CCCR_C0HSRR_ENG1 - eTPU 1 Channel 0 Host Service Request Register
0x40C	Reserved
0x410	eTPU_CCCR_C1CR_ENG1 - eTPU 1 Channel 1 Configuration Register
0x414	eTPU_CCCR_C1SCR_ENG1 - eTPU 1 Channel 1 Status and Control Register
0x418	eTPU_CCCR_C1HSRR_ENG1 - eTPU 1 Channel 1 Host Service Request Register
0x41C	Reserved
-----	-----
0x5F0	eTPU_CCCR_C31CR_ENG1 - eTPU 1 Channel 31 Configuration Register
0x5F4	eTPU_CCCR_C31SCR_ENG1 - eTPU 1 Channel 31 Status and Control Register
0x5F8	eTPU_CCCR_C31HSRR_ENG1 - eTPU 1 Channel 31 Host Service Request Register
0x5FC - 0x7FF	Reserved
0x800	eTPU_CCCR_C0CR_ENG2 - eTPU 2 Channel 0 Configuration Register
0x804	eTPU_CCCR_C0SCR_ENG2 - eTPU 2 Channel 0 Status and Control Register
0x808	eTPU_CCCR_C0HSRR_ENG2 - eTPU 2 Channel 0 Host Service Request Register
0x80C	Reserved
0x810	eTPU_CCCR_C1CR_ENG2 - eTPU 2 Channel 1 Configuration Register
0x814	eTPU_CCCR_C1SCR_ENG2 - eTPU 2 Channel 1 Status and Control Register
0x818	eTPU_CCCR_C1HSRR_ENG2 - eTPU 2 Channel 1 Host Service Request Register

Table continues on the next page...

Table 2-4. Detailed memory map (continued)

Offset	Use
0x81C	Reserved
-----	-----
0x9F0	eTPU_CCCR_C31CR_ENG2 - eTPU 2 Channel 31 Configuration Register
0x9F4	eTPU_CCCR_C31SCR_ENG2 - eTPU 2 Channel 31 Status and Control Register
0x9F8	eTPU_CCCR_C31HSRR_ENG2 - eTPU 2 Channel 31 Host Service Request Register
0x9FC - 0xBFF	Reserved
Shared Subsystem Support Registers	
0xC00	eTPU_SSIR_SSS0IR - eTPU SSS 0 Integration Register
0xC04	eTPU_SSIR_SSS1IR - eTPU SSS 1 Integration Register
0xC08	eTPU_SSIR_SSS2IR - eTPU SSS 2 Integration Register
0xC0A	eTPU_SSIR_SSS3IR - eTPU SSS 3 Integration Register
0xC0C	eTPU_SSIR_SSS4IR - eTPU SSS 4 Integration Register
0xC10	eTPU_SSIR_SSS5IR - eTPU SSS 5 Integration Register
0xC14	eTPU_SSIR_SSS6IR - eTPU SSS 6 Integration Register
0xC18	eTPU_SSIR_SSS7IR - eTPU SSS 7 Integration Register
0xC1A	eTPU_SSIR_SSS8IR - eTPU SSS 8 Integration Register
0xC1C	eTPU_SSIR_SSS9IR - eTPU SSS 9 Integration Register
0x1000-0x3FFF	Reserved
0x4000 - 0x7FFF	Shared Subsystems area - SSS
0x8000 - 0xBFFF[3]	Shared Data Memory - SDM
0xC000 - 0xFFFF[3]	Shared Data Memory - SDM - PSE mirror [4]
0x10000-1FFFF[-1]	Shared Code Memory - SCM[7]

1. This register is not implemented in some MCUs; see SCMOFFDATAR.
2. These registers are not implemented in some MCUs; see Memory error support.
3. The actual SDM size is MCU-dependent (see the chip-specific eTPU information). SDM is allocated from the starting value of the given range of offsets.
4. Parameter Sign Extension access area, see Parameter Access.
5. The actual SCM size is MCU-dependent (see the chip-specific eTPU information). SCM is allocated from the starting value of the given range of offsets. When the size is not the maximum, the unused SCM address range returns the value of SCMOFFDATAR.
6. The actual SCM size is MCU-dependent (see the chip-specific eTPU information). SCM is allocated from the starting value of the given range of offsets. When the size is not the maximum, the unused SCM address range returns the value of SCMOFFDATAR.
7. SCM access is available only when eTPU_SCR_MCR[VIS]=1, under certain conditions (see MCR).

2.6 System Configuration memory map and registers

eTPU_SCR memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	eTPU Module Configuration Register (eTPU_SCR_MCR)	32	R/W	See section	2.6.1/92
4	eTPU Coherent Dual-Parameter Controller Register (eTPU_SCR_CDCR)	32	R/W	0000_0000h	2.6.2/96
C	eTPU MISC Compare Register (eTPU_SCR_MISCCMPR)	32	R/W	0000_0000h	2.6.3/98
10	eTPU SCM Off-range Data Register (eTPU_SCR_SCMOFFDATAR)	32	R/W	F377_5FFBh	2.6.4/98
14	eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1)	32	R/W	See section	2.6.5/99
18	eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2)	32	R/W	See section	2.6.6/103

2.6.1 eTPU Module Configuration Register (eTPU_SCR_MCR)

This register is global to both eTPU engines, and resides in the shared BIU. It gathers global configuration and status in the eTPU system, including global exception. It is also used for configuring the SCM (shared code memory) operation and test.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	SDMERR	WDTO1	WDTO2	MGE1	MGE2	ILF1	ILF2	Reserved	0	0	SCMSIZE				
W	GEC															
Reset	0	0	0	0	0	0	0	0	0	0	0	*	*	*	*	*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LTO1	LTO2	0		SCMMISC	SCMMISF	SCMMISEN	SCMERR	0	VIS	0					GTBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* Notes:
- SCMSIZE field: See the chip-specific eTPU information for this field's reset value.

eTPU_SCR_MCR field descriptions

Field	Description
0 GEC	Global Exception Clear. This write-only bit negates global exception request and clears global exception status bits SDMERR, SSAERR, SCMERR, WDTO1/2, MGE1/2, ILF1/2, LTO1/2, and SCMMISF. GEC works the same way with either one or both Engines in Module Disable Mode. 0 Keep Global Exception request and status bits SDMERR, SCMERR, WDTO1/2, ILF1/2, MGE1/2, LTO1/2, and SCMMISF as is. 1 Negate Global Exception, clear status bits SDMERR, SCMERR, WDTO1/2, ILF1/2, MGE1/2, LTO1/2, and SCMMISF.
1 SDMERR	SDM Read Error This flag indicates that an SDM non-correctable read error occurred on a microengine or host (including CDC) read, generating a Global Exception. This bit is cleared by writing 1 to GEC. 0 No Global Exception pending because of SDM read error. 1 Global Exception requested by SDM read error is pending.
2 WDTO1	Watchdog 1 Timeout. Flag WDTO1 indicates that a Watchdog Timeout occurred in engine 1, generating a Global Exception. This bit is cleared by writing 1 to GEC. 0 No Global Exception pending because of Watchdog timeout. 1 Global Exception requested by Watchdog timeout.
3 WDTO2	Watchdog 2 Timeout. Flag WDTO2 indicates that a Watchdog Timeout occurred in engine 2, generating a Global Exception. This bit is cleared by writing 1 to GEC. 0 No Global Exception pending because of Watchdog timeout. 1 Global Exception requested by Watchdog timeout.
4 MGE1	Microcode Global Exception Engine 1. This bit indicates that a global exception was asserted by microcode executed on engine 1. The determination of the reason why the global exception was asserted is application dependent: it can be coded in an SDM status parameter, for instance. This bit is cleared by writing 1 to GEC. 0 No microcode-requested global exception pending. 1 Global exception requested by microcode is pending.
5 MGE2	Microcode Global Exception Engine 2. This bit indicates that a global exception was asserted by microcode executed on engine 2. The determination of the reason why the global exception was asserted is application dependent: it can be coded in an SDM status parameter, for instance. This bit is cleared by writing 1 to GEC. 0 No microcode-requested global exception pending. 1 Global exception requested by microcode is pending.
6 ILF1	Illegal Instruction Flag for Engine 1. The ILFA1 flag is set by the microengine to indicate that an illegal instruction was decoded in Engine 1. This bit is cleared by host writing 1 to GEC. See Illegal instructions for more details.

Table continues on the next page...

eTPU_SCR_MCR field descriptions (continued)

Field	Description
	0 Illegal Instruction not detected. 1 Illegal Instruction detected by Engine 1.
7 ILF2	Illegal Instruction Flag for Engine 2. The ILFA2 flag is set by the microengine to indicate that an illegal instruction was decoded in Engine 2. This bit is cleared by host writing 1 to GEC. See Illegal instructions for more details. 0 Illegal Instruction not detected. 1 Illegal Instruction detected by Engine 2.
8 Reserved	Reserved. This field is reserved.
9 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
10 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
11–15 SCMSIZE	SCM Size = 64 Kbytes. This read-only field holds the number of 2 KB SCM Blocks minus 1. See the chip-specific eTPU information for this field's reset value.
16 LTO1	Latency Timeout for Engine 1. The LTO1 flag indicates the latency measured by the Latency Monitor of Engine 1 surpassed the specified maximum. See Latency Monitor for details. This bit is cleared by writing 1 to GEC. 0 No latency timeout detected by Engine 1’s Latency Monitor. 1 Latency timeout detected by Engine 1's Latency Monitors.
17 LTO2	Latency Timeout for Engine 2. The LTO2 flag indicates the latency measured by the Latency Monitor of Engine 2 surpassed the specified maximum. See Latency Monitor for details. This bit is cleared by writing 1 to GEC. 0 No latency timeout detected by Engine 2’s Latency Monitor. 1 Latency timeout detected by Engine 2's Latency Monitors.
18–19 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
20 SCMMISC	SCM MISC Complete flag. Flag SCMMISC indicates that MISC has completed the evaluation of the SCM signature since reset or since the last time it was cleared. SCMMISC is cleared by writing 1 to it, and is not cleared when MISC is disabled (SCMMISEN=0). SCMMISC asserts at the end of the SCM memory scan, either if the signature matches or not.

Table continues on the next page...

eTPU_SCR_MCR field descriptions (continued)

Field	Description
	0 MISC has not yet completed an SCM signature calculation and compare since the last time SCMMISC was cleared. 1 MISC completed at least one SCM signature calculation and compare since the last time SCMMISC was cleared.
21 SCMMISF	SCM MISC Flag. The SCMMISF bit is set by the SCM MISC (Multiple Input Signature Calculator) logic to indicate that the calculated signature does not match the expected value, at the end of a MISC iteration. See SCM Test - Multiple Input Signature Calculator for more details. This bit is cleared when Global Exception is cleared by writing 1 to GEC. 0 Signature mismatch not detected. 1 MISC has read entire SCM array and the expected signature in eTPU MISC Compare Register (eTPU_SCR_MISCCMPR) does not match the value calculated.
22 SCMMISEN	SCM MISC Enable. The SCMMISEN bit is used for enabling/disabling the operation of the MISC logic. SCMMISEN is readable and writable at any time. The MISC logic will only operate when this bit is set to 1. When the bit is reset, the MISC address counter is set to the initial SCM address. When enabled, the MISC will continuously cycle through the SCM addresses, reading each and calculating a CRC. In order to save power, the MISC can be disabled by clearing the SCMMISEN bit. See SCM Test - Multiple Input Signature Calculator for more details. SCMMISEN resets automatically when MISC logic detects an error, i.e., when SCMMISF transitions from 0 to 1, disabling the MISC operation. 0 MISC operation disabled. The MISC logic is reset to its initial state. 1 MISC operation enabled.
23 SCMERR	SCM Read Error. This flag indicates that an SCM non-correctable read error occurred on a host or microengine (including MISC) read, generating a Global Exception. This bit is cleared by writing 1 to GEC. 0 No Global Exception pending because of SCM read error. 1 Global Exception requested by SCM read error is pending.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 VIS	SCM Visibility Bit. VIS bit turns SCM visible to the IP-Bus and resets MISC state (but SCMMISEN keeps its value). This bit is write protected when any of the Engines is not halted or stopped. When VIS=1, the MDIS bits in the eTPU Memory Error Support registers are write protected. 0 SCM is not visible to the slave bus. Accessing SCM address space issues a bus error, writes are protected and reads are meaningless. 1 SCM is visible to the slave bus. MISC state is reset.
26–30 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

eTPU_SCR_MCR field descriptions (continued)

Field	Description
31 GTBE	Global Time Base Enable. GTBE enables time bases in both Engines, allowing them to be started synchronously. **NOTE:** Global Time Base Enable action may also depend on other blocks, as explained in GTBE - Global Time Base Enable. **NOTE:** When GTBE is turned off with Angle Mode enabled, the EAC must be reinitialized before GTBE is turned on again. The EAC reinitialization procedure is described in Restarting angle logic. 0 Time bases in both Engines are disabled to run. 1 Time bases in both Engines are enabled to run.

2.6.2 eTPU Coherent Dual-Parameter Controller Register (eTPU_SCR_CDCR)

This register configures and controls dual-parameter coherent transfers. For more information, see Coherent Dual-parameter Controller (CDC).

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	STS	CTBASE							PBBASE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PWIDTH	PARM0							WR	PARM1						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_SCR_CDCR field descriptions

Field	Description
0 STS	Start Bit. This bit is set by the host in order to start the data transfer between the parameter buffer pointed by PBBASE and the target addresses selected by the concatenation of fields CTBASE and PARM0/1. The host receives wait-states until the data transfer is complete, when this bit is reset by the coherency logic (see Coherent Dual-parameter Controller (CDC)). Therefore, host always reads STS as 0.

Table continues on the next page...

eTPU_SCR_CDCR field descriptions (continued)

Field	Description
	0 Write does not start a coherent transfer. 1 Write starts a coherent transfer.
1–5 CTBASE	Channel Transfer Base. This field concatenates with fields PARM0 and PARM1 to determine the absolute word offset (from the SDM base) of the parameters to be transferred: Parameter 0 word address = {CTBASE, PARM0} + SDM base word address. Parameter 1 word address = {CTBASE, PARM1} + SDM base word address.
6–15 PBBASE	Parameter Buffer Base Address. This field points to the base address of the parameter buffer location, with granularity of 2 parameters (8 bytes). The host (byte) address of the first parameter in the buffer is PBBASE * 8 + SDM Base Byte Address. The microengine absolute (word) address of the first parameter in the buffer is PBBASE * 2.
16 PWIDTH	Parameter Width Selection. This bit selects the width of the parameters to be transferred between the PB and the target address. 0 Transfer 24-bit parameters. The upper byte remains unchanged in the destination address. 1 Transfer 32-bit parameters. All 32 bits of the parameters are written in the destination address.
17–23 PARM0	Channel Parameter number 0. The PARM0 and PARM1 fields in concatenation with CTBASE determine the word address offset (from the SDM base) of the parameters that are destination or source (defined by WR) of the coherent transfer. The word SDM address offset of the parameters are {CTBASE, PARM0/1}. **NOTE:** PARM0 and PARM1 allow non-contiguous parameters to be transferred coherently. The parameter pointed by {CTBASE, PARM0} is the first transferred.
24 WR	Read/Write selection. This bit selects the direction of the coherent data transfer. 0 Read operation. Data transfer is from the selected SDM address to the PB. 1 Write operation. Data transfer is from the PB to the selected SDM address.
25–31 PARM1	Channel Parameter number 1. The PARM0 and PARM1 fields in concatenation with CTBASE determine the word address offset (from the SDM base) of the parameters that are destination or source (defined by WR) of the coherent transfer. The word SDM address offset of the parameters are {CTBASE, PARM0/1}. **NOTE:** PARM0 and PARM1 allow non-contiguous parameters to be transferred coherently. The parameter pointed by {CTBASE, PARM0} is the first transferred.

2.6.3 eTPU MISC Compare Register (eTPU_SCR_MISCCMPR)

This register holds the 32-bit signature expected from the whole SCM array. This register must be written by the host with the 32-bit word to be compared against the calculated signature at the end of the MISC cycle. This register is global to both eTPU Engines. For more detail, see SCM Test - Multiple Input Signature Calculator.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ETPUMISCCMP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_SCR_MISCCMPR field descriptions

Field	Description
0–31 ETPUMISCCMP	Expected Multiple Input Signature Register value. For details, see SCM Test - Multiple Input Signature Calculator.

2.6.4 eTPU SCM Off-range Data Register (eTPU_SCR_SCMOFFDATAR)

This register holds the 32-bit value returned when SCM array is accessed at non-implemented addresses, either by the Host or by the microengine. This register can be written by the host with the 32-bit instruction to be executed by the microengine to recover from runaway code. This register is global to both eTPU Engines. For more detail see SCM Off-range data.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ETPUSCMOFFDATAR																															
Reset	1	1	1	1	0	0	1	1	0	1	1	1	0	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1

eTPU_SCR_SCMOFFDATAR field descriptions

Field	Description
0–31 ETPUSCMOFFDATAR	SCM Off-range read data value. See SCM Off-range data.

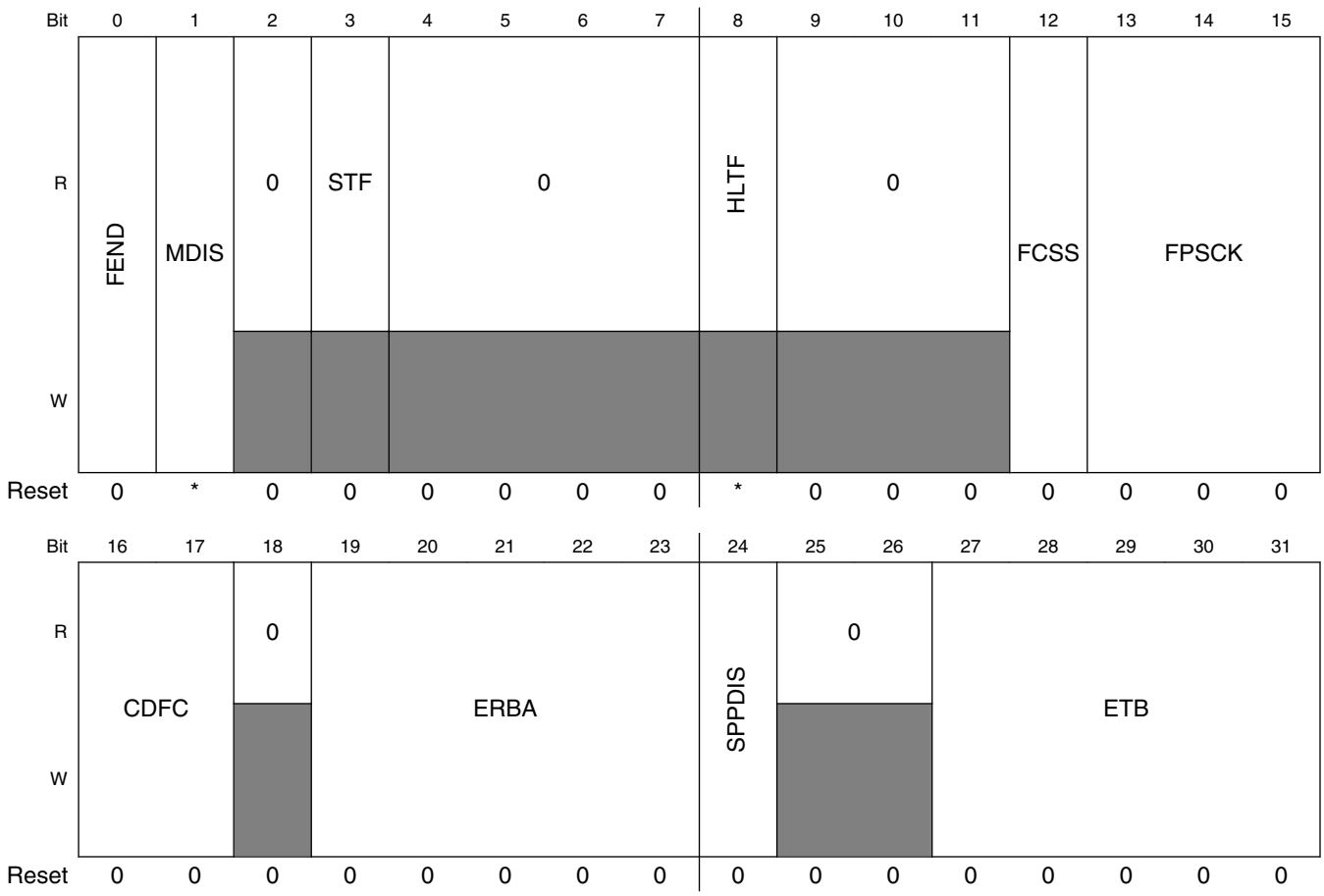
2.6.5 eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1)

This register holds configuration and status fields for engine 1.

NOTE

See Register Reset Values in the chip-specific eTPU information.

Address: 0h base + 14h offset = 14h



* Notes:

- HLTF field: HLTF field reset value is 1 if, after a reset, the eTPU Nexus debug support is on and a breakpoint request is asserted. Otherwise the reset value is 0.
- MDIS field: Reset value is 1.

eTPU_SCR_ECR_ENG1 field descriptions

Field	Description
0 FEND	Force End. FEND assertion terminates any current running thread as if an END instruction have been executed (see Ending Current Thread - END). This bit is self-negating when the thread ends.

Table continues on the next page...

eTPU_SCR_ECR_ENG1 field descriptions (continued)

Field	Description
	NOTE: Only on rare occasions (e.g., during a long stall, see Microengine Stall) FEND can be read as 1, because it negates as soon as the end begins execution. Writing FEND = 1 is ignored and FEND stays 0 when the microengine is in TST, halted, stopped, or idle (no thread executing). 0 Normal operation. 1 Ends any ongoing thread.
1 MDIS	Module Disable Bit. When MDIS is set, the Engine shuts down its internal clocks, going into Module Disable Mode. TCR1 and TCR2 cease to increment, and input sampling stops. The Engine asserts the stop flag (STF) bit to indicate that it has stopped. However, the BIU continues to run, and the Host can access all registers except for the channel registers (see channel registers starting with eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1)) **NOTE:** The Timebase registers can still be read with MDIS=1, but writes are ineffective and a Bus Error is issued. Global Channel Registers and SDM can be accessed normally. After MDIS is set, even before STF asserts, data read from the channel registers is not meaningful and writes are ineffective, issuing a Bus Error. When the MDIS bit is asserted while the microcode is executing, the eTPU will stop when the thread is complete. Stop completes on the next eTPU clock after the stop condition is valid. **NOTE:** Once MDIS is switched from 1 to 0 or vice versa, it must not write a different value until STF changes accordingly. The MDIS bit is write-protected when VIS=1. See Register Reset Values in the chip-specific eTPU information. 0 eTPU Engine runs. 1 Commands Engine to stop its clocks.
2 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
3 STF	Stop Flag Bit. Each Engine asserts its stop flag (STF) to indicate that it has stopped. Only then the host can assume that the Engine has actually stopped. The eTPU system is fully stopped when the STF bits of both eTPU Engines are asserted. In case of a stop request, the eTPU responds by acknowledging stop only after both eTPU 1 and eTPU 2 have been stopped. Engine only stops when any ongoing thread is complete also in this case. Summarizing Engine stop conditions, which STF reflects: STF_1 := (after stop completed) MDIS_1 | device stop request. STF_2 := (after stop completed) MDIS_2 | device stop request. STF_1 and STF_2 mean STF bit from engine 1 and STF bit from engine 2, respectively. See Register Reset Values in the chip-specific eTPU information. 0 Engine is operating. 1 Engine has stopped (after the local MDIS bit has been asserted, or after the IP-Bus stop line has been asserted).

Table continues on the next page...

eTPU_SCR_ECR_ENG1 field descriptions (continued)

Field	Description
4–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8 HLTF	Halt Mode Flag. If eTPU Engine entered halt state, this flag is asserted. The flag remains asserted while the microengine is in halt state, even during a single-step or forced instruction execution. See Development support features for further details about entering Halt Mode. 0 eTPU Engine is not halted. 1 eTPU Engine is halted.
9–11 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
12 FCSS	Filter Clock Source Selection Speeds up the filter clock source before the prescaler, allowing more input capture resolution at minimum prescaling. **NOTE:** FCSS=1 also makes the channel work on T2/T4 timing mode (see T2/T4 channel timing). 0 Use eTPU clock / 2 as EDF clock source before prescaler. 1 Use eTPU clock as EDF clock source before prescaler
13–15 FPSCK	Filter Prescaler Clock Control. FPSCK controls the prescaling of the clocks used in digital filters for the channel input signals and TCRCLK input, as shown in the table below. Filtering can be controlled independently by each engine, but all input digital filters in the same engine have same clock prescaling. For more details, see Filter clock prescaler. A new value written to FPSCK only becomes effective when the filter prescaler finishes the current count.
16–17 CDFC	Channel Digital Filter Control.

Table 2-5. Filter prescaler clock control

Filter control	Sample on system clock divided by:	
	FCSS = 0	FCSS = 1
000	2	1
001	4	2
010	8	4
011	16	8
100	32	16
101	64	32
110	128	64
111	256	128

Table continues on the next page...

eTPU_SCR_ECR_ENG1 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>These bits select a digital filtering mode for the channels when configured as inputs for improved noise immunity (refer to the table below). The eTPU has three digital filtering modes for the channels which provide programmable trade-off between signal latency and noise immunity (see Enhanced Digital Filter - EDF). Changing CDFC during eTPU normal input channel operation is not recommended since it changes the behavior of the transition detection logic while executing its operation.
Table 2-6. Channel digital filter control
<table>
<tr><th>CDFC</th><th>Selected digital filter</th></tr>
<tr><td>00</td><td>TPU2/3 Two Sample Mode: Using the filter clock which is the system clock divided by (2, 4, 8,..., 256) as a sampling clock (selected by FPSCK), comparing two consecutive samples which agree with each other sets the input signal state. This is the default reset state.</td></tr>
<tr><td>01</td><td>eTPU bypass mode: the input signal is taken unfiltered, also making the channels work on T2/T4 timing mode.[3]</td></tr>
<tr><td>10</td><td>eTPU Three Sample Mode: Similar to the TPU2/3 two sample mode, but comparing three consecutive samples which agree with each other sets the input signal state.</td></tr>
<tr><td>11</td><td>eTPU Continuous Mode: Signal need to be stable for the whole filter clock period. This mode compares all the values at the rate of eTPU clock (FCSS=1) or eTPU clock divided by two(FCSS=0), between two consecutive filter clock pulses. Signal needs to be continuously stable for the entire period. If all the values agree with each other, input signal state is updated.</td></tr>
</table>
1. See T2/T4 channel timing.</td></tr>
<tr><td>18
Reserved</td><td>Reserved.

This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>19–23
ERBA</td><td>Engine Relative Base Address.

This field value is concatenated with the AID instruction field in engine relative address mode to form the SDM address (see Engine Relative Addressing Mode).</td></tr>
<tr><td>24
SPPDIS</td><td>Schedule Priority Passing Disable.

SPPDIS is used to disable the priority passing mechanism of the microengine scheduler (see Primary Scheme - Priority Among Channels on Different Levels).

NOTE: SPPDIS bit must not be changed while any channel is enabled.

0 Scheduler priority passing mechanism enabled.
1 Scheduler priority passing mechanism disabled.</td></tr>
<tr><td>25–26
Reserved</td><td>Reserved.

This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>27–31
ETB</td><td>Entry Table Base.</td></tr>
</table>

Table continues on the next page...

eTPU_SCR_ECR_ENG1 field descriptions (continued)

Field	Description
	The field determines the location of the microcode entry table for the eTPU functions in SCM (see Entry Points). The following table shows the entry table base address options.

Table 2-7. Entry table base address options

ETB	Entry Table Base Address for CPU Host Address (byte format)	Entry Table Base Address for Microcode Address (word format)
00000	0x000	0x000
00001	0x800	0x200
00010	0x1000	0x400
--------	----------	---------
11110	0xF000	0x3C00
11111	0xF800	0x3E00

3. See T2/T4 channel timing.

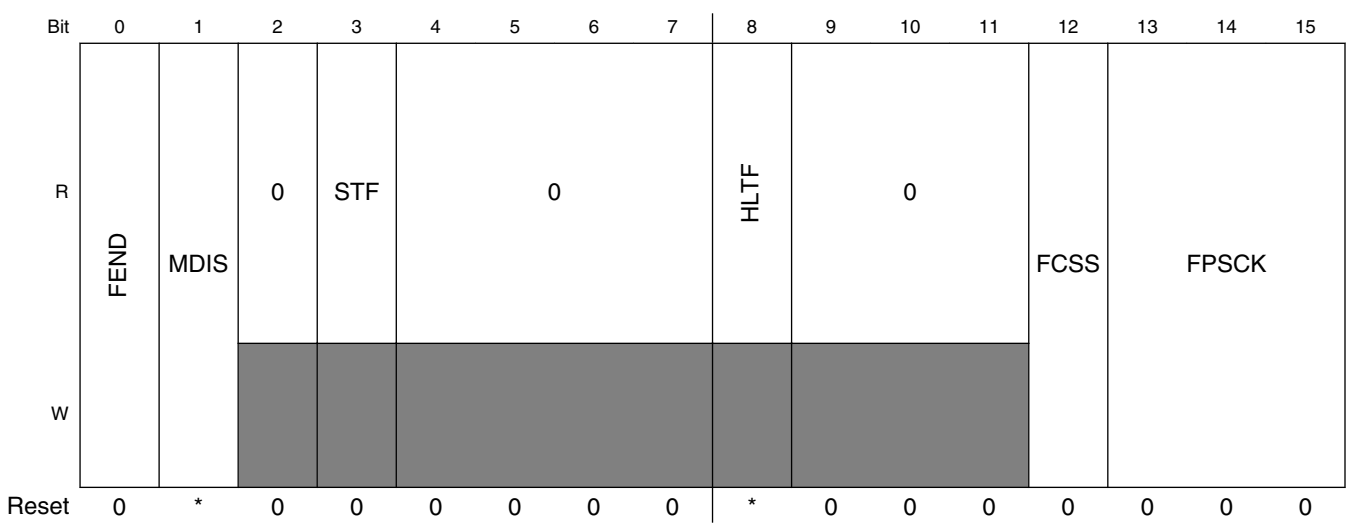
2.6.6 eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2)

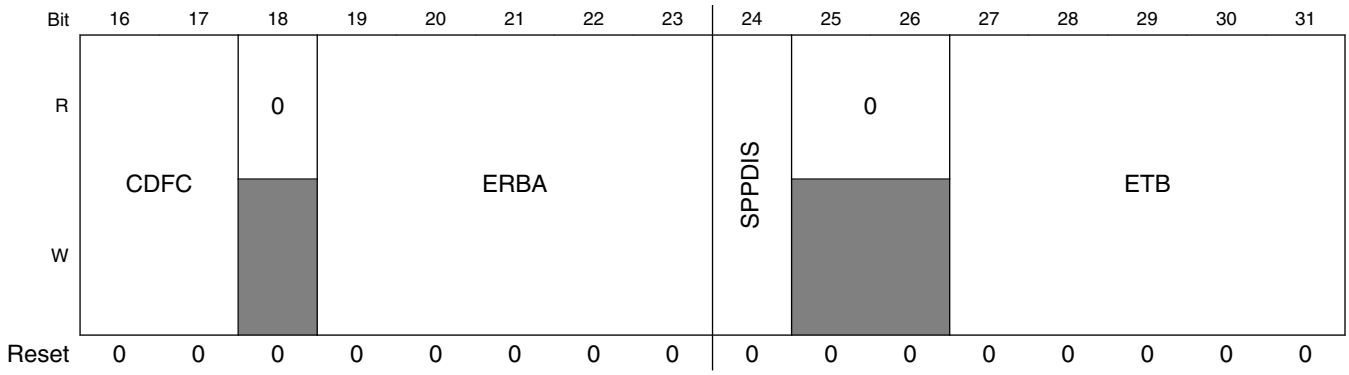
This register holds configuration and status fields for engine 2.

NOTE

See Register Reset Values in the chip-specific eTPU information.

Address: 0h base + 18h offset = 18h





* Notes:
- HLTF field: HLTF field reset value is 1 if, after a reset, the eTPU Nexus debug support is on and a breakpoint request is asserted. Otherwise the reset value is 0.
- MDIS field: Reset value is 1.

eTPU_SCR_ECR_ENG2 field descriptions

Field	Description
0 FEND	Force End. FEND assertion terminates any current running thread as if an END instruction have been executed (see Ending Current Thread - END). This bit is self-negating when the thread ends. **NOTE:** Only on rare occasions (e.g., during a long stall, see Microengine Stall) FEND can be read as 1, because it negates as soon as the end begins execution. Writing FEND = 1 is ignored and FEND stays 0 when the microengine is in TST, halted, stopped, or idle (no thread executing). 0 Normal operation. 1 Ends any ongoing thread.
1 MDIS	Module Disable Bit. When MDIS is set, the Engine shuts down its internal clocks, going into Module Disable Mode. TCR1 and TCR2 cease to increment, and input sampling stops. The Engine asserts the stop flag (STF) bit to indicate that it has stopped. However, the BIU continues to run, and the Host can access all registers except for the channel registers (see channel registers starting with eTPU Channel Configuration-Control registers) **NOTE:** The Timebase registers can still be read with MDIS=1, but writes are ineffective and a Bus Error is issued. Global Channel Registers and SDM can be accessed normally. After MDIS is set, even before STF asserts, data read from the channel registers is not meaningful and writes are ineffective, issuing a Bus Error. When the MDIS bit is asserted while the microcode is executing, the eTPU will stop when the thread is complete. Stop completes on the next eTPU clock after the stop condition is valid. **NOTE:** Once MDIS is switched from 1 to 0 or vice versa, it must not write a different value until STF changes accordingly. The MDIS bit is write-protected when VIS=1. See Register Reset Values in the chip-specific eTPU information.

Table continues on the next page...

eTPU_SCR_ECR_ENG2 field descriptions (continued)

Field	Description
	0 eTPU Engine runs. 1 Commands Engine to stop its clocks.
2 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
3 STF	Stop Flag Bit. Each Engine asserts its stop flag (STF) to indicate that it has stopped. Only then the host can assume that the Engine has actually stopped. The eTPU system is fully stopped when the STF bits of both eTPU Engines are asserted. In case of IP-Bus stop, the eTPU system responds by acknowledging stop only after both eTPU 1 and eTPU 2 have been stopped. Engine only stops when any ongoing thread is complete also in this case. Summarizing Engine stop conditions, which STF reflects: STF_1 := (after stop completed) MDIS_1 | device stop request. STF_2 := (after stop completed) MDIS_2 | device stop request. STF_1 and STF_2 mean STF bit from engine 1 and STF bit from engine 2, respectively. See Register Reset Values in the chip-specific eTPU information. 0 Engine is operating. 1 Engine has stopped (after the local MDIS bit has been asserted, or after the IP-Bus stop line has been asserted).
4–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8 HLTF	Halt Mode Flag. If eTPU Engine entered halt state, this flag is asserted. The flag remains asserted while the microengine is in halt state, even during a single-step or forced instruction execution. See Development support features for further details about entering Halt Mode. 0 eTPU Engine is not halted. 1 eTPU Engine is halted.
9–11 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
12 FCSS	Filter Clock Source Selection Speeds up the filter clock source before the prescaler, allowing more input capture resolution at minimum prescaling. **NOTE:** FCSS=1 also makes the channel work on T2/T4 timing mode (see T2/T4 channel timing). 0 Use eTPU clock / 2 as EDF clock source before prescaler. 1 Use eTPU clock as EDF clock source before prescaler
13–15 FPSCK	Filter Prescaler Clock Control.

Table continues on the next page...

eTPU_SCR_ECR_ENG2 field descriptions (continued)

Field	Description
	FPSCK controls the prescaling of the clocks used in digital filters for the channel input signals and TCRCLK input, as shown in the table below. Filtering can be controlled independently by each engine, but all input digital filters in the same engine have same clock prescaling. For more details, see Filter clock prescaler. A new value written to FPSCK only becomes effective when the filter prescaler finishes the current count. **Table 2-8. Filter prescaler clock control** Filter control: 000 → Sample on system clock divided by: 2 001 → 4 010 → 8 011 → 16 100 → 32 101 → 64 110 → 128 111 → 256
16–17 CDFC	Channel Digital Filter Control. These bits select a digital filtering mode for the channels when configured as inputs for improved noise immunity (refer to the table below). The eTPU has three digital filtering modes for the channels which provide programmable trade-off between signal latency and noise immunity (see Enhanced Digital Filter - EDF). Changing CDFC during eTPU normal input channel operation is not recommended since it changes the behavior of the transition detection logic while executing its operation. **Table 2-9. Channel digital filter control** CDFC: 00 → Selected digital filter: TPU2/3 Two Sample Mode: Using the filter clock which is the system clock divided by (2, 4, 8,..., 256) as a sampling clock (selected by FPSCK), comparing two consecutive samples which agree with each other sets the input signal state. This is the default reset state. 01 → eTPU bypass mode: the input signal is taken unfiltered, also making the channels work on T2/T4 timing mode.[3] 10 → eTPU Three Sample Mode: Similar to the TPU2/3 two sample mode, but comparing three consecutive samples which agree with each other sets the input signal state. 11 → eTPU Continuous Mode: Signal need to be stable for the whole filter clock period. This mode compares all the values at the rate of eTPU clock (FCSS=1) or eTPU clock divided by two(FCSS=0), between two consecutive filter clock pulses. Signal needs to be continuously stable for the entire period. If all the values agree with each other, input signal state is updated. 1. See T2/T4 channel timing.
18 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

eTPU_SCR_ECR_ENG2 field descriptions (continued)

Field	Description
19–23 ERBA	Engine Relative Base Address. This field value is concatenated with the AID instruction field in engine relative address mode to form the SDM address (see Engine Relative Addressing Mode).
24 SPPDIS	Schedule Priority Passing Disable. SPPDIS is used to disable the priority passing mechanism of the microengine scheduler (see Primary Scheme - Priority Among Channels on Different Levels). **NOTE:** SPPDIS bit must not be changed while any channel is enabled. 0 Scheduler priority passing mechanism enabled. 1 Scheduler priority passing mechanism disabled.
25–26 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–31 ETB	Entry Table Base. The field determines the location of the microcode entry table for the eTPU functions in SCM (see Entry Points). The following table shows the entry table base address options.

Table 2-10. Entry table base address options

ETB	Entry Table Base Address for CPU Host Address (byte format)	Entry Table Base Address for Microcode Address (word format)
00000	0x000	0x000
00001	0x800	0x200
00010	0x1000	0x400
--------	----------	---------
11110	0xF000	0x3C00
11111	0xF800	0x3E00

3. See T2/T4 channel timing.

2.7 Timebase memory map and registers

Time Base registers allows configuration and visibility of internally-generated time bases TCR1 and TCR2. There is one of each of these registers for each eTPU Engine.

NOTE

Writes to these registers issue bus error and are ineffective when MDIS=1. Reads are always allowed.

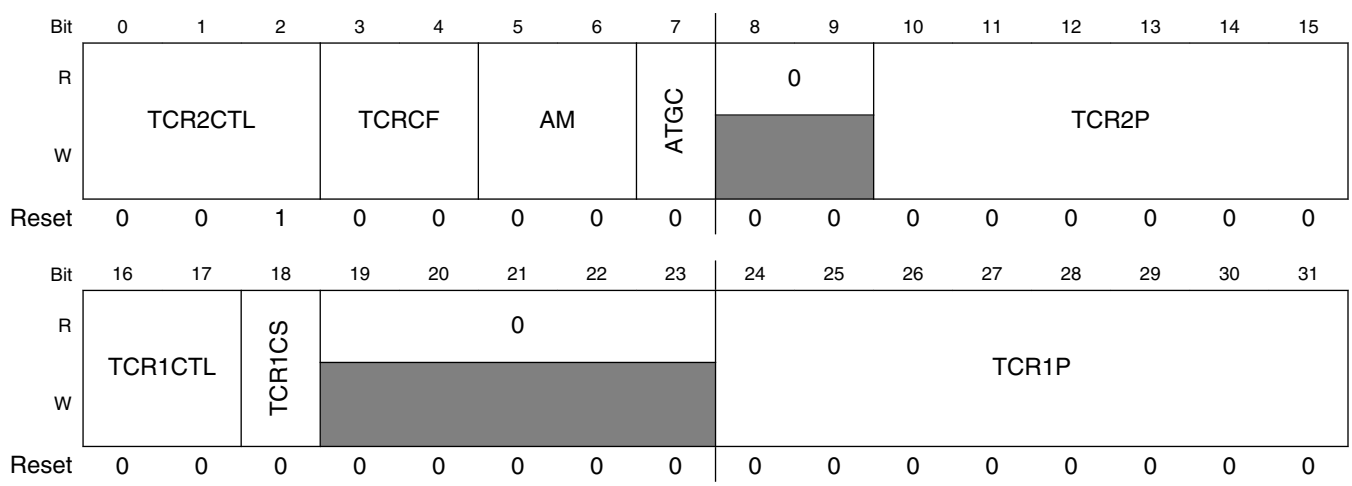
eTPU_TBR memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20	eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1)	32	R/W	2000_0000h	2.7.1/108
24	eTPU Engine 1 Time Base 1 (TCR1) Visibility Register (eTPU_TBR_TB1R_ENG1)	32	R/W	0000_0000h	2.7.2/112
28	eTPU Engine 1 Time Base 2 (TCR2) Visibility Register (eTPU_TBR_TB2R_ENG1)	32	R/W	0000_0000h	2.7.3/112
2C	eTPU Engine 1 STAC Configuration Register (eTPU_TBR_STAC_ENG1)	32	R/W	See section	2.7.4/113
40	eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2)	32	R/W	2000_0000h	2.7.5/115
44	eTPU Engine 2 Time Base 1 (TCR1) Visibility Register (eTPU_TBR_TB1R_ENG2)	32	R/W	0000_0000h	2.7.6/119
48	eTPU Engine 2 Time Base 2 (TCR2) Visibility Register (eTPU_TBR_TB2R_ENG2)	32	R/W	0000_0000h	2.7.7/119
4C	eTPU Engine 2 STAC Configuration Register (eTPU_TBR_STAC_ENG2)	32	R/W	See section	2.7.8/120

2.7.1 eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1)

This register configures several timebase options.

Address: 0h base + 20h offset = 20h



eTPU_TBR_TBCR_ENG1 field descriptions

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>0–2
TCR2CTL</td><td>TCR2 Clock/Gate Control.

These bits are part of the TCR2 clocking system (see Time bases). They determine the clock source for TCR2 before the prescaler. TCR2 can count on any detected edge of the TCRCLK signal or use it for gating eTPU clock divided by 8. After reset - TCRCLK signal rising edge is selected. TCR2 can also be clocked by the eTPU clock divided by 8. TCR2CTL also determines the TCRCLK edge selected for angle tooth detection in angle mode. See the following table.
Table 2-11. TCR2 Clock Source
<table><tr><th>TCR2CTL</th><th>AM = 0

TCR2 Clock before prescaler</th><th>AM > 0

Angle Tooth detection</th></tr><tr><td>000</td><td>Gated DIV8 clock (eTPU clock / 8). In this case, when the external TCRCLK signal is low, the DIV8 clock is blocked, preventing it from incrementing the TCR2 prescaler. When the external TCRCLK signal is high, TCR2 prescaler is incremented at the frequency of the eTPU clock divided by 8.</td><td>Do not use with AM > 0</td></tr><tr><td>001</td><td>Rise transition on TCRCLK signal increments the TCR2 prescaler.</td><td>Rising edge</td></tr><tr><td>010</td><td>Fall transition on TCRCLK signal increments the TCR2 prescaler.</td><td>Falling edge</td></tr><tr><td>011</td><td>Rise or Fall transition on TCRCLK signal increments the TCR2 prescaler.</td><td>Both edges</td></tr><tr><td>100</td><td>DIV8 clock (eTPU clock / 8)</td><td>Do not use with AM > 0</td></tr><tr><td>101</td><td>Do not use with AM=0</td><td>Do not use with AM > 0</td></tr><tr><td>110</td><td>Do not use with AM=0</td><td>No edge[1]</td></tr><tr><td>111</td><td>TCR2 frozen, except as STAC client</td><td>Do not use with AM > 0</td></tr></table>1. TCRCLK edges are not detected by the EAC logic, but they can still be detected by the channel 0 logic if AM=01.</td></tr>
<tr><td>3–4
TCRCF</td><td>TCRCLK Signal Filter Control.

This field controls the TCRCLK digital filter (see TCRCLK digital filter), determining whether the TCRCLK signal input (after a synchronizer) is filtered with the same filter clock as the channel input signals (see Enhanced Digital Filter - EDF) or uses the eTPU clock divided by 2, and also whether the TCRCLK digital filter works in integrator mode or two sample mode (see the following table).
Table 2-12. TCRCLK Filter Clock/Mode
<table><tr><th>TCRCF</th><th>Filter Clock</th><th>Filter Mode</th></tr><tr><td>00</td><td>eTPU clock divided by 2</td><td>Two sample</td></tr><tr><td>01</td><td>Filter clock of the channels</td><td>Two sample</td></tr><tr><td>10</td><td>eTPU clock divided by 2</td><td>Integrator</td></tr><tr><td>11</td><td>Filter clock of the channels</td><td>Integrator</td></tr></table></td></tr>
<tr><td>5–6
AM</td><td>Angle Mode Selection.</td></tr>
</table>

Table continues on the next page...

eTPU_TBR_TBCR_ENG1 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>This field enables the Enhanced Angle Counter logic to generate angle information (see EAC - eTPU Angle Counter), and also select the tooth signal input and the channel used to process it, as shown in the following table. When EAC is not disabled by AM and neither TCR1 nor TCR2 are STAC Clients, the EAC (eTPU Angle Clock) hardware provides angle information to the channels using the TCR2 bus. When AM is reset (non-angle mode), the EAC operation is disabled, and its internal registers can be used as general purpose. For more information, see EAC - eTPU Angle Counter.
Table 2-13. AM - Angle Mode Selection
<table>
<tr><th>Value</th><th>TCR2 Value</th><th>Tooth signal</th><th>Tooth processing channel</th></tr>
<tr><td>00</td><td>Timebase (EAC operation disabled)</td><td colspan="2">Not applicable</td></tr>
<tr><td>01</td><td rowspan="3">Angle Ticks</td><td>TCRCLK input</td><td>0</td></tr>
<tr><td>10</td><td>Channel 1 input</td><td>1</td></tr>
<tr><td>11</td><td>Channel 2 input</td><td>2</td></tr>
</table>
If TCR1 or TCR2 is a STAC Bus Client (see STAC interface), the EAC operation is not allowed, and AM must be set to 0.

NOTE: Changing AM may cause unexpected transition detections on the channel selected by AM, depending on the channel mode and state (see Transition detection and Time Base Capture). If AM must be changed with GTBE=1, the recommended procedure is described in Restarting angle logic.</td></tr>
<tr><td>7
ATGC</td><td>Angle Tick Generator Clock.

This bit selects the clock that provides the time interval unit for Angle Tick Generation. For details, see Generating the angle ticks.

0 TCR1 prescaler output provides time intervals for tick generation.
1 TCR2 prescaler output provides time intervals for tick generation.</td></tr>
<tr><td>8–9
Reserved</td><td>Reserved.

This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>10–15
TCR2P</td><td>Timer Count Register 2 Prescaler Control.

These bits are part of the TCR2 clocking system (see Time bases). TCR2 is clocked from the output of a prescaler. The prescaler divides its input by (TCR2P + 1) allowing frequency divisions from 1 to 64. The prescaler input is the eTPU clock divided by 8 (in gated or non-gated clock mode), or Internal Timebase input, or TCRCLK filtered input. This field has no effect on TCR2 in Angle Mode.</td></tr>
<tr><td>16–17
TCR1CTL</td><td>TCR1 Clock/Gate Control.

TCR1CTL is part of the TCR1 clocking system (see Time bases). It determines, together with TCR1CS, the clock source for TCR1. TCR1 can count on detected rising edge of the TCRCLK signal, the eTPU clock, or the eTPU clock divided by 2 (see the table in the TCR1CS bit description). After reset TCRCLK signal is selected.</td></tr>
<tr><td>18
TCR1CS</td><td>TCR1 Clock Source

TCR1CS provides the option to double the TCR1 incrementing speed, using eTPU clock as its clock source instead of eTPU clock / 2.</td></tr>
</table>

Table continues on the next page...

eTPU_TBR_TBCR_ENG1 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
NOTE: If ATGC=0, the clock source of the EAC angle tick generator will still be an even division of eTPU clock if TCR1CS=1, obeying to the field TCR1P as if TCR1CS=0 (see Angle Tick Generator).
Table 2-14. TCR1 Clock Source
<table>
<tr><th>TCR1CTL</th><th>TCR1CS</th><th>TCR1 Clock before prescaler</th></tr>
<tr><td>00</td><td>0</td><td>Selects TCRCLK as clock source for the TCR1 prescaler.[2]</td></tr>
<tr><td>01</td><td>0</td><td>Reserved</td></tr>
<tr><td>01</td><td>1</td><td>Reserved</td></tr>
<tr><td>10</td><td>0</td><td>Selects eTPU clock divided by 2 as clock source for the TCR1 prescaler.</td></tr>
<tr><td>10</td><td>1</td><td>Selects eTPU clock as clock source for the TCR1 prescaler.</td></tr>
<tr><td>11</td><td>0</td><td>TCR1 frozen, except as a STAC client</td></tr>
</table>
1. This selection must not be used in Angle Mode.

0 Use eTPU clock / 2 as TCR1 clock source before the prescaler, if that clock source is selected by TCR1CTL.

1 Use eTPU clock as TCR1 clock source before the prescaler; can only be set in specific combinations with TCR1CTL, as shown in the table.

NOTE: TCR1CS=1 also makes the channel work on T2/T4 timing mode (see T2/T4 channel timing).
</td></tr>
<tr><td>19–23
Reserved</td><td>Reserved.

This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>24–31
TCR1P</td><td>Timer Count Register 1 Prescaler Control.

TCR1 is clocked from the output of a prescaler. The input to the prescaler is the internal eTPU clock divided by 2, eTPU clock, or the output of TCRCLK filter. The prescaler divides this input by (TCR1P + 1) allowing frequency divisions from 1 up to 256.</td></tr>
</table>

1. TCRCLK edges are not detected by the EAC logic, but they can still be detected by the channel 0 logic if AM=01.
2. This selection must not be used in Angle Mode.

2.7.2 eTPU Engine 1 Time Base 1 (TCR1) Visibility Register (eTPU_TBR_TB1R_ENG1)

This register provides visibility of the TCR1 time base for host read access (see Time bases). This register is read-only. The value of the TCR1 time base shown can be driven by the TCR1 counter or imported from STAC bus, depending on the configuration set in eTPU Engine 1 STAC Configuration Register (eTPU_TBR_STAC_ENG1).

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								TCR1																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_TBR_TB1R_ENG1 field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 TCR1	TCR1 value. TCR1 value used on matches and captures. See Time bases.

2.7.3 eTPU Engine 1 Time Base 2 (TCR2) Visibility Register (eTPU_TBR_TB2R_ENG1)

This register provides visibility of the TCR2 time base for host read access (see Time bases). This register is read-only. The value of the TCR2 time base shown can be driven by the TCR2 counter or imported from the STAC bus, depending on Angle Mode and STAC configurations set in eTPU Engine 1 STAC Configuration Register (eTPU_TBR_STAC_ENG1).

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								TCR2																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_TBR_TB2R_ENG1 field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 TCR2	TCR2 value. TCR2 value used on matches and captures. See Time bases.

2.7.4 eTPU Engine 1 STAC Configuration Register (eTPU_TBR_STAC_ENG1)

This register configures the eTPU STAC bus operation as a STAC Server/Client module (see STAC interface).

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	REN1	RSC1	0		SERVER_ID1				0				SRV1			
W																
Reset	0	0	0	0	*	*	*	*	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	REN2	RSC2	0		SERVER_ID2				0				SRV2			
W																
Reset	0	0	0	0	*	*	*	*	0	0	0	0	0	0	0	0

* Notes:
- SERVER_ID2 field: See the chip-specific eTPU information for the reset value.
- SERVER_ID1 field: See the chip-specific eTPU information for the reset value.

eTPU_TBR_STAC_ENG1 field descriptions

Field	Description
0 REN1	TCR1 Resource Client/Server Operation Enable Bit. This bit enables Client/Server operation to eTPU STAC resources. REN1 enables TCR1 STAC bus operations. 0 Server/Client Operation for TCR1 is disabled. 1 Server/Client Operation for TCR1 is enabled.
1 RSC1	TCR1 Resource Server/Client Assignment Bits. This bit selects the eTPU data resource assignment to be used as Servers or Clients. For Server mode, MCU-dependent value determines the unique server address assigned to each TCR. See the chip-specific

Table continues on the next page...

eTPU_TBR_STAC_ENG1 field descriptions (continued)

Field	Description
	eTPU information for the STAC server IDs. For a Client mode, the SRV1 field determines the Server address to which the Client listens. **NOTE:** When TCR1 or TCR2 is configured as a STAC Bus Client (REN1/2 = 1, RSC1/2 = 0) the eTPU Angle Clock hardware cannot be used. **NOTE:** RSC1 must not be changed when the REN1 bit is asserted. 0 Resource Client operation. 1 Resource Server operation.
2–3 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SERVER_ID1	STAC ID 1. STAC Server ID= 0 (read-only MCU-dependent values; see the chip-specific eTPU information for the STAC server IDs) used for TCR1 when STAC servers are used.
8–11 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
12–15 SRV1	TCR1 Resource Server. This bit selects the address of the specific STAC Server to which the local TCR1 listens when configured as a STAC Client. SRV1 selects the STAC Server of TCR1.
16 REN2	TCR2 Resource Client/Server Operation Enable Bit. This bit enables Client/Server operation to eTPU STAC resources. REN2 enables TCR2 STAC bus operations. 0 Server/Client Operation for TCR2 is disabled. 1 Server/Client Operation for TCR2 is enabled.
17 RSC2	TCR2 Resource Server/Client Assignment Bits. **NOTE:** Either time or angle TCR2 values. 0 Resource Client operation. 1 Resource Server operation.
18–19 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
20–23 SERVER_ID2	STAC ID 2. STAC Server ID= 2 (read-only MCU-dependent values; see the chip-specific eTPU information for the STAC server IDs) used for TCR2 when when STAC servers are used.
24–27 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

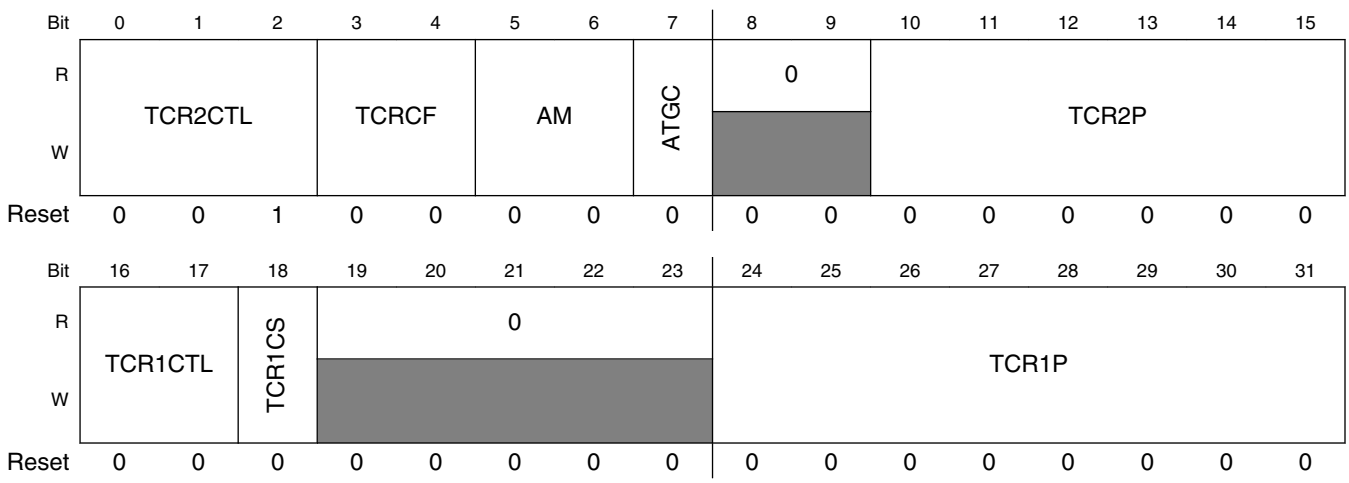
eTPU_TBR_STAC_ENG1 field descriptions (continued)

Field	Description
28–31 SRV2	TCR2 Resource Server. This bit selects the address of the specific STAC Server to which the local TCR2 listens when configured as a STAC Client. SRV2 selects the Red Line STAC Server of TCR2.

2.7.5 eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2)

This register configures several timebase options.

Address: 0h base + 40h offset = 40h



eTPU_TBR_TBCR_ENG2 field descriptions

Field	Description
0–2 TCR2CTL	TCR2 Clock/Gate Control. These bits are part of the TCR2 clocking system (see Time bases). They determine the clock source for TCR2 before the prescaler. TCR2 can count on any detected edge of the TCRCLK signal or use it for gating eTPU clock divided by 8. After reset - TCRCLK signal rising edge is selected. TCR2 can also be

Table continues on the next page...

eTPU_TBR_TBCR_ENG2 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>clocked by the eTPU clock divided by 8. TCR2CTL also determines the TCRCLK edge selected for angle tooth detection in angle mode. See the following table.
Table 2-15. TCR2 Clock Source
<table>
<tr><th>TCR2CTL</th><th>AM = 0
TCR2 Clock before prescaler</th><th>AM > 0
Angle Tooth detection</th></tr>
<tr><td>000</td><td>Gated DIV8 clock (eTPU clock / 8). In this case, when the external TCRCLK signal is low, the DIV8 clock is blocked, preventing it from incrementing the TCR2 prescaler. When the external TCRCLK signal is high, TCR2 prescaler is incremented at the frequency of the eTPU clock divided by 8.</td><td>Do not use with AM > 0</td></tr>
<tr><td>001</td><td>Rise transition on TCRCLK signal increments the TCR2 prescaler.</td><td>Rising edge</td></tr>
<tr><td>010</td><td>Fall transition on TCRCLK signal increments the TCR2 prescaler.</td><td>Falling edge</td></tr>
<tr><td>011</td><td>Rise or Fall transition on TCRCLK signal increments the TCR2 prescaler.</td><td>Both edges</td></tr>
<tr><td>100</td><td>DIV8 clock (eTPU clock / 8)</td><td>Do not use with AM > 0</td></tr>
<tr><td>101</td><td>Do not use with AM=0</td><td>Do not use with AM > 0</td></tr>
<tr><td>110</td><td>Do not use with AM=0</td><td>No edge[1]</td></tr>
<tr><td>111</td><td>TCR2 frozen, except as STAC client</td><td>Do not use with AM > 0</td></tr>
</table>
1. TCRCLK edges are not detected by the EAC logic, but they can still be detected by the channel 0 logic if AM=01.</td></tr>
<tr><td>3–4
TCRCF</td><td>TCRCLK Signal Filter Control.

This field controls the TCRCLK digital filter (see TCRCLK digital filter, determining whether the TCRCLK signal input (after a synchronizer) is filtered with the same filter clock as the channel input signals (see Enhanced Digital Filter - EDF) or uses the eTPU clock divided by 2, and also whether the TCRCLK digital filter works in integrator mode or two sample mode (see the following table).
Table 2-16. TCRCLK Filter Clock/Mode
<table>
<tr><th>TCRCF</th><th>Filter Clock</th><th>Filter Mode</th></tr>
<tr><td>00</td><td>eTPU clock divided by 2</td><td>Two sample</td></tr>
<tr><td>01</td><td>Filter clock of the channels</td><td>Two sample</td></tr>
<tr><td>10</td><td>eTPU clock divided by 2</td><td>Integrator</td></tr>
<tr><td>11</td><td>Filter clock of the channels</td><td>Integrator</td></tr>
</table></td></tr>
<tr><td>5–6
AM</td><td>Angle Mode Selection.

This field enables the Enhanced Angle Counter logic to generate angle information (see EAC - eTPU Angle Counter), and also select the tooth signal input and the channel used to process it, as shown in the following table. When EAC is not disabled by AM and neither TCR1 nor TCR2 are STAC Clients, the EAC (eTPU Angle Clock) hardware provides angle information to the channels using the TCR2 bus. When AM</td></tr>
</table>

Table continues on the next page...

eTPU_TBR_TBCR_ENG2 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>is reset (non-angle mode), the EAC operation is disabled, and its internal registers can be used as general purpose. For more information, see EAC - eTPU Angle Counter.
Table 2-17. AM - Angle Mode Selection
<table>
<tr><th>Value</th><th>TCR2 Value</th><th>Tooth signal</th><th>Tooth processing channel</th></tr>
<tr><td>00</td><td>Timebase (EAC operation disabled)</td><td colspan="2">Not applicable</td></tr>
<tr><td>01</td><td rowspan="3">Angle Ticks</td><td>TCRCLK input</td><td>0</td></tr>
<tr><td>10</td><td>Channel 1 input</td><td>1</td></tr>
<tr><td>11</td><td>Channel 2 input</td><td>2</td></tr>
</table>
If TCR1 or TCR2 is a STAC Bus Client (see STAC interface), the EAC operation is not allowed, and AM must be set to 0.

NOTE: Changing AM may cause unexpected transition detections on the channel selected by AM, depending on the channel mode and state (see Transition detection and Time Base Capture). If AM must be changed with GTBE=1, the recommended procedure is described in Restarting angle logic.</td></tr>
<tr><td>7
ATGC</td><td>Angle Tick Generator Clock.

This bit selects the clock that provides the time interval unit for Angle Tick Generation. For details, see Generating the angle ticks.

0 TCR1 prescaler output provides time intervals for tick generation.
1 TCR2 prescaler output provides time intervals for tick generation.</td></tr>
<tr><td>8–9
Reserved</td><td>Reserved.

This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>10–15
TCR2P</td><td>Timer Count Register 2 Prescaler Control.

These bits are part of the TCR2 clocking system (see Time bases). TCR2 is clocked from the output of a prescaler. The prescaler divides its input by (TCR2P + 1) allowing frequency divisions from 1 to 64. The prescaler input is the eTPU clock divided by 8 (in gated or non-gated clock mode), or Internal Timebase input, or TCRCLK filtered input. This field has no effect on TCR2 in Angle Mode.</td></tr>
<tr><td>16–17
TCR1CTL</td><td>TCR1 Clock/Gate Control.

TCR1CTL is part of the TCR1 clocking system (see Time bases). It determines, together with TCR1CS, the clock source for TCR1. TCR1 can count on detected rising edge of the TCRCLK signal, the eTPU clock, or the eTPU clock divided by 2 (see the table in the TCR1CS bit description). After reset TCRCLK signal is selected.</td></tr>
<tr><td>18
TCR1CS</td><td>TCR1 Clock Source

TCR1CS provides the option to double the TCR1 incrementing speed, using eTPU clock as its clock source instead of eTPU clock / 2.</td></tr>
</table>

Table continues on the next page...

eTPU_TBR_TBCR_ENG2 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
NOTE: If ATGC=0, the clock source of the EAC angle tick generator will still be an even division of eTPU clock if TCR1CS=1, obeying to the field TCR1P as if TCR1CS=0 (see Angle Tick Generator).
Table 2-18. TCR1 Clock Source
<table>
<tr><th>TCR1CTL</th><th>TCR1CS</th><th>TCR1 Clock before prescaler</th></tr>
<tr><td>00</td><td>0</td><td>Selects TCRCLK as clock source for the TCR1 prescaler.[2]</td></tr>
<tr><td>01</td><td>0</td><td>Reserved</td></tr>
<tr><td>01</td><td>1</td><td>Reserved</td></tr>
<tr><td>10</td><td>0</td><td>Selects eTPU clock divided by 2 as clock source for the TCR1 prescaler.</td></tr>
<tr><td>10</td><td>1</td><td>Selects eTPU clock as clock source for the TCR1 prescaler.</td></tr>
<tr><td>11</td><td>0</td><td>TCR1 frozen, except as a STAC client</td></tr>
</table>
1. This selection must not be used in Angle Mode.
0 Use eTPU clock / 2 as TCR1 clock source before the prescaler, if that clock source is selected by TCR1CTL.
1 Use eTPU clock as TCR1 clock source before the prescaler; can only be set in specific combinations with TCR1CTL, as shown in the table.
NOTE: TCR1CS=1 also makes the channel work on T2/T4 timing mode (see T2/T4 channel timing).
</td></tr>
<tr><td>19–23
Reserved</td><td>Reserved.

This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>24–31
TCR1P</td><td>Timer Count Register 1 Prescaler Control.

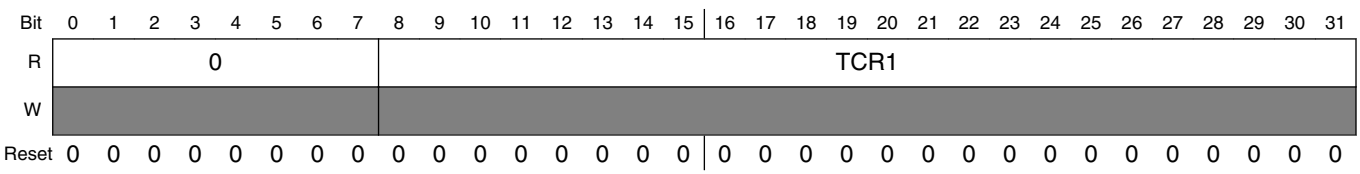
TCR1 is clocked from the output of a prescaler. The input to the prescaler is the internal eTPU clock divided by 2, eTPU clock, or the output of TCRCLK filter. The prescaler divides this input by (TCR1P + 1) allowing frequency divisions from 1 up to 256.</td></tr>
</table>

1. TCRCLK edges are not detected by the EAC logic, but they can still be detected by the channel 0 logic if AM=01.
2. This selection must not be used in Angle Mode.

2.7.6 eTPU Engine 2 Time Base 1 (TCR1) Visibility Register (eTPU_TBR_TB1R_ENG2)

This register provides visibility of the TCR1 time base for host read access (see Time bases). This register is read-only. The value of the TCR1 time base shown can be driven by the TCR1 counter or imported from STAC bus, depending on the configuration set in eTPU Engine 2 STAC Configuration Register (eTPU_TBR_STAC_ENG2).

Address: 0h base + 44h offset = 44h



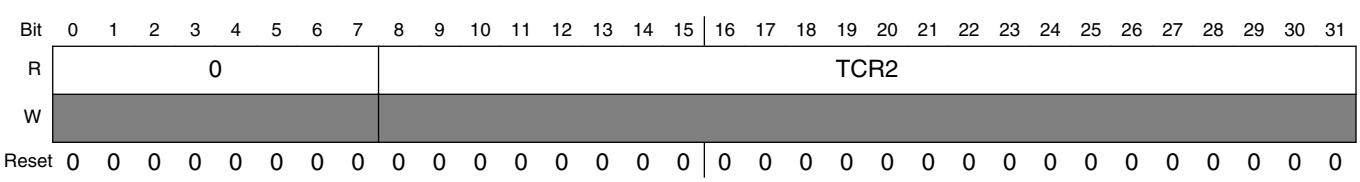
eTPU_TBR_TB1R_ENG2 field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 TCR1	TCR1 value. TCR1 value used on matches and captures. See Time bases.

2.7.7 eTPU Engine 2 Time Base 2 (TCR2) Visibility Register (eTPU_TBR_TB2R_ENG2)

This register provides visibility of the TCR2 time base for host read access (see Time bases). This register is read-only. The value of the TCR2 time base shown can be driven by the TCR2 counter or imported from the STAC bus, depending on Angle Mode and STAC configurations set in eTPU Engine 2 STAC Configuration Register (eTPU_TBR_STAC_ENG2).

Address: 0h base + 48h offset = 48h



eTPU_TBR_TB2R_ENG2 field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 TCR2	TCR2 value. TCR2 value used on matches and captures. See Time bases.

2.7.8 eTPU Engine 2 STAC Configuration Register (eTPU_TBR_STAC_ENG2)

This register configures the eTPU STAC bus operation as a STAC Server/Client module (see STAC interface).

Address: 0h base + 4Ch offset = 4Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	REN1	RSC1	0		SERVER_ID1				0				SRV1			
W																
Reset	0	0	0	0	*	*	*	*	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	REN2	RSC2	0		SERVER_ID2				0				SRV2			
W																
Reset	0	0	0	0	*	*	*	*	0	0	0	0	0	0	0	0

* Notes:
- SERVER_ID2 field: See the chip-specific eTPU information for the reset value.
- SERVER_ID1 field: See the chip-specific eTPU information for the reset value.

eTPU_TBR_STAC_ENG2 field descriptions

Field	Description
0 REN1	TCR1 Resource Client/Server Operation Enable Bit. This bit enables Client/Server operation to eTPU STAC resources. REN1 enables TCR1 STAC bus operations. 0 Server/Client Operation for TCR1 is disabled. 1 Server/Client Operation for TCR1 is enabled.
1 RSC1	TCR1 Resource Server/Client Assignment Bits. This bit selects the eTPU data resource assignment to be used as Servers or Clients. For Server mode, MCU-dependent value determines the unique server address assigned to each TCR. See the chip-specific

Table continues on the next page...

eTPU_TBR_STAC_ENG2 field descriptions (continued)

Field	Description
	eTPU information for the STAC server IDs. For a Client mode, the SRV1 field determines the Server address to which the Client listens. **NOTE:** Resource identifies any parameter that changes along the time and can be exported / imported from other device. In eTPU context, a resource can be TCR1 or TCR2 (either Time or Angle values). **NOTE:** When TCR1 or TCR2 is configured as a STAC Bus Client (REN2 = 1, RSC2 = 0) the eTPU Angle Clock hardware cannot be used. **NOTE:** RSC1 must not be changed when the REN1 bit is asserted. 0 TCR1 Client operation. 1 TCR1 Server operation.
2–3 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SERVER_ID1	STAC ID 1. STAC Server ID= 1 (read-only MCU-dependent values; see the chip-specific eTPU information for the STAC server IDs) used for TCR1 when STAC servers are used.
8–11 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
12–15 SRV1	TCR1 Resource Server. This bit selects the address of the specific STAC Server to which the local TCR1 listens when configured as a STAC Client. SRV1 selects the STAC Server of TCR1.
16 REN2	TCR2 Resource Client/Server Operation Enable Bit. This bit enables Client/Server operation to eTPU STAC resources. REN2 enables TCR2 STAC bus operations. 0 Server/Client Operation for resource 2 is disabled. 1 Server/Client Operation for resource 2 is enabled.
17 RSC2	TCR2 Resource Server/Client Assignment Bits. **NOTE:** Resource identifies any parameter that changes along the time and can be exported / imported from other device. In eTPU context, a resource can be TCR1 or TCR2 (either Time or Angle values). 0 Resource Client operation. 1 Resource Server operation.
18–19 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
20–23 SERVER_ID2	STAC ID 2.

Table continues on the next page...

eTPU_TBR_STAC_ENG2 field descriptions (continued)

Field	Description
	STAC Server ID= 3 (read-only MCU-dependent values; see the chip-specific eTPU information for the STAC server IDs) used for TCR2 when when STAC servers are used.
24–27 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
28–31 SRV2	TCR2 Resource Server. This bit selects the address of the specific STAC Server to which the local TCR2 listens when configured as a STAC Client. SRV2 selects the Red Line STAC Server of TCR2.

2.8 Engine memory map and registers

This section gathers registers that are engine-related, other than eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2).

eTPU_ER memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
60	eTPU Watchdog Timer Register for Engine 1 (eTPU_ER_WDTR_ENG1)	32	R/W	0000_0000h	2.8.1/123
68	eTPU Idle Register for Engine 1 (eTPU_ER_IDLER_ENG1)	32	R/W	Undefined	2.8.2/124
70	eTPU Watchdog Timer Register for Engine 2 (eTPU_ER_WDTR_ENG2)	32	R/W	0000_0000h	2.8.3/125
78	eTPU Idle Register for Engine 2 (eTPU_ER_IDLER_ENG2)	32	R/W	Undefined	2.8.4/126

2.8.1 eTPU Watchdog Timer Register for Engine 1 (eTPU_ER_WDTR_ENG1)

This register configures the watchdog timer for the engine.

Address: 0h base + 60h offset = 60h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WDM		0	WMO	0											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_ER_WDTR_ENG1 field descriptions

Field	Description
0–1 WDM	Watchdog Mode. WDM selects the Watchdog operation mode. For more information on the Watchdog operation, see Watchdog). **NOTE:** Before a new mode is configured, all conditions below must apply: 1. All channels must be disabled (CPR =00). 2. No thread must be executing (eTPU Channel Service Status Register for Engine 1 (eTPU_GCR_CSSR_ENG1) reads 0). 3. The watchdog must be disabled (WDM=00). 00 Disabled 01 Reserved 10 Thread length 11 Busy length
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 WMO	Watchdog Monitor Operation. This bit enables the monitor operation of the watchdog, inhibiting the thread ending and channel disabling in case of a timeout. The Global Exception occurs on a watchdog timeout, independently of WMO. **NOTE:** WDSn, WDTO1, WDTO2, and the Global Exception always asserts in case of a watchdog timeout, independently of WMO. 0 Force the end of the running thread and disable the service channel in case of a timeout. 1 Do not force the end of the running thread, nor disable the service channel in case of a timeout.
4–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 WDCNT	Watchdog Count.

Table continues on the next page...

eTPU_ER_WDTR_ENG1 field descriptions (continued)

Field	Description
	This field indicates the maximum number of microcycles allowed for a thread (in thread length mode) or a sequence of threads (in busy length mode) before the current running thread is forced to end. For more information on Watchdog operation, see Watchdog. **NOTE:** The TST microcycles are also counted by the watchdog. **NOTE:** Fields WDCNT and WMO must not be changed when the watchdog is enabled.

2.8.2 eTPU Idle Register for Engine 1 (eTPU_ER_IDLER_ENG1)

This register counts the microcycles in which the microengine is idle (see Idle Counter

Address: 0h base + 68h offset = 68h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IDLE_CNT															
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IDLE_CNT															ICLR
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

eTPU_ER_IDLER_ENG1 field descriptions

Field	Description
0–30 IDLE_CNT	Idle Count. This is a freerunning count of the number of idle microcycles in the microengine. For more information on idle counter operation, see Idle Counter . **NOTE:** When the register is read, the ICLR bit is concatenated with the IDLE_CNT field (as the LSB) to form a 32-bit value.
31 ICLR	Idle Clear. Writing a 1 to this bit clears the IDLE_CNT field. **NOTE:** When the register is read, the ICLR bit is concatenated with the IDLE_CNT field (as the LSB) to form a 32-bit value. 0 Do not clear the IDLE_CNT field. 1 Clear the IDLE_CNT field.

2.8.3 eTPU Watchdog Timer Register for Engine 2 (eTPU_ER_WDTR_ENG2)

This register configures the watchdog timer for the engine.

Address: 0h base + 70h offset = 70h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WDM		0	WMO	0											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_ER_WDTR_ENG2 field descriptions

Field	Description
0–1 WDM	Watchdog Mode. WDM selects the Watchdog operation mode. For more information on the Watchdog operation, see Watchdog. **NOTE:** Before a new mode is configured, all conditions below must apply: 1. All channels must be disabled (CPR =00). 2. No thread must be executing (eTPU Channel Service Status Register for Engine 2 (eTPU_GCR_CSSR_ENG2) reads 0). 3. The watchdog must be disabled (WDM=00). 00 Disabled 01 Reserved 10 Thread length 11 Busy length
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 WMO	Watchdog Monitor Operation. This bit enables the monitor operation of the watchdog, inhibiting the thread ending and channel disabling in case of a timeout. The Global Exception occurs on a watchdog timeout, independently of WMO. **NOTE:** WDSn, WDTO1, WDTO2, and the Global Exception always assert in case of a watchdog timeout, independently of WMO. 0 Force the end of the running thread and disable the service channel in case of a timeout. 1 Do not force the end of the running thread, nor disable the service channel in case of a timeout.
4–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 WDCNT	Watchdog Count.

Table continues on the next page...

eTPU_ER_WDTR_ENG2 field descriptions (continued)

Field	Description
	This field indicates the maximum number of microcycles allowed for a thread (in thread length mode) or a sequence of threads (in busy length mode) before the current running thread is forced to end.For more information on Watchdog operation, see Watchdog. **NOTE:** The TST microcycles are also counted by the watchdog. **NOTE:** Fields WDCNT and WMO must not be changed when the watchdog is enabled.

2.8.4 eTPU Idle Register for Engine 2 (eTPU_ER_IDLER_ENG2)

This register counts the microcycles in which the microengine is idle (see Idle Counter).

Address: 0h base + 78h offset = 78h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IDLE_CNT															
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IDLE_CNT															ICLR
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

eTPU_ER_IDLER_ENG2 field descriptions

Field	Description
0–30 IDLE_CNT	Idle Count. This is a freerunning count of the number of idle microcycles in the microengine. For more information on idle counter operation, see Idle Counter . **NOTE:** When the register is read, the ICLR bit is concatenated with the IDLE_CNT field (as the LSB) to form a 32-bit value.
31 ICLR	Idle Clear. Writing a 1 to this bit clears the IDLE_CNT field. **NOTE:** When the register is read, the ICLR bit is concatenated with the IDLE_CNT field (as the LSB) to form a 32-bit value. 0 Do not clear the IDLE_CNT field. 1 Clear the IDLE_CNT field.

2.9 Latency Monitor memory map and registers

eTPU_LMR memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
80	eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM0CR_ENG1)	32	R/W	0000_0000h	2.9.1/128
84	eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM0MR_ENG1)	32	R/W	0000_0000h	2.9.2/130
88	eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM1CR_ENG1)	32	R/W	0000_0000h	2.9.1/128
8C	eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM1MR_ENG1)	32	R/W	0000_0000h	2.9.2/130
90	eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM2CR_ENG1)	32	R/W	0000_0000h	2.9.1/128
94	eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM2MR_ENG1)	32	R/W	0000_0000h	2.9.2/130
98	eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM3CR_ENG1)	32	R/W	0000_0000h	2.9.1/128
9C	eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM3MR_ENG1)	32	R/W	0000_0000h	2.9.2/130
A0	eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM4CR_ENG1)	32	R/W	0000_0000h	2.9.1/128
A4	eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM4MR_ENG1)	32	R/W	0000_0000h	2.9.2/130
A8	eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM5CR_ENG1)	32	R/W	0000_0000h	2.9.1/128
AC	eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM5MR_ENG1)	32	R/W	0000_0000h	2.9.2/130
C0	eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM0CR_ENG2)	32	R/W	0000_0000h	2.9.3/131
C4	eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM0MR_ENG2)	32	R/W	0000_0000h	2.9.4/132
C8	eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM1CR_ENG2)	32	R/W	0000_0000h	2.9.3/131
CC	eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM1MR_ENG2)	32	R/W	0000_0000h	2.9.4/132
D0	eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM2CR_ENG2)	32	R/W	0000_0000h	2.9.3/131
D4	eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM2MR_ENG2)	32	R/W	0000_0000h	2.9.4/132
D8	eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM3CR_ENG2)	32	R/W	0000_0000h	2.9.3/131
DC	eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM3MR_ENG2)	32	R/W	0000_0000h	2.9.4/132

Table continues on the next page...

eTPU_LMR memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E0	eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM4CR_ENG2)	32	R/W	0000_0000h	2.9.3/131
E4	eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM4MR_ENG2)	32	R/W	0000_0000h	2.9.4/132
E8	eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM5CR_ENG2)	32	R/W	0000_0000h	2.9.3/131
EC	eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM5MR_ENG2)	32	R/W	0000_0000h	2.9.4/132

2.9.1 eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM*n*CR_ENG1)

This register configures the operation of individual latency monitors.

Address: 0h base + 80h offset + (8d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LMM		0			SRSEL			0			MONCHAN					MAXLAT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_LMR_LM*n*CR_ENG1 field descriptions

Field	Description
0–1 LMM	Latency Monitor Mode. This field selects the operation mode of the latency monitor. See the Latency Monitor section for detailed mode descriptions. **NOTE:** LMM must be in disabled mode to be changed to channel or priority mode. CPR must not be changed while LMM of any Latency Monitor is in priority mode. 00 Disabled mode 01 Channel mode 10 Priority mode 11 Reserved
2–4 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SRSEL	Service Request Selection

Table continues on the next page...

eTPU_LMR_LM*n*CR_ENG1 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>This field selects which service requests are monitored. The SRSEL value has different meanings in channel and priority modes, according to the following table.
Table 2-19. SRSEL - Service Request Selection
<table>
<tr><th>SRSEL</th><th>Channel Mode</th><th>Priority Mode</th></tr>
<tr><td>001</td><td>Link SRs monitored only</td><td>Low priority monitored</td></tr>
<tr><td>010</td><td>Channel SRs monitored only</td><td>Medium priority monitored</td></tr>
<tr><td>100</td><td>HSRs monitored only</td><td>High priority monitored</td></tr>
<tr><td>000</td><td>No SRs monitored</td><td rowspan="5">Reserved</td></tr>
<tr><td>011</td><td>Channel and Link SRs monitored</td></tr>
<tr><td>101</td><td>HSRs and Link SRs monitored</td></tr>
<tr><td>110</td><td>HSRs and Channel SRs monitored</td></tr>
<tr><td>111</td><td>All SRs monitored</td></tr>
</table></td></tr>
<tr><td>8–10
Reserved</td><td>Reserved.

This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>11–15
MONCHAN</td><td>Monitored Channel.

When the latency monitor is configured for channel mode, this field selects which channel has the latency monitored. MONCHAN has no meaning in priority mode.

NOTE: MONCHAN must only be changed when the Latency Monitor is in disabled mode.</td></tr>
<tr><td>16–31
MAXLAT</td><td>Maximum Latency.

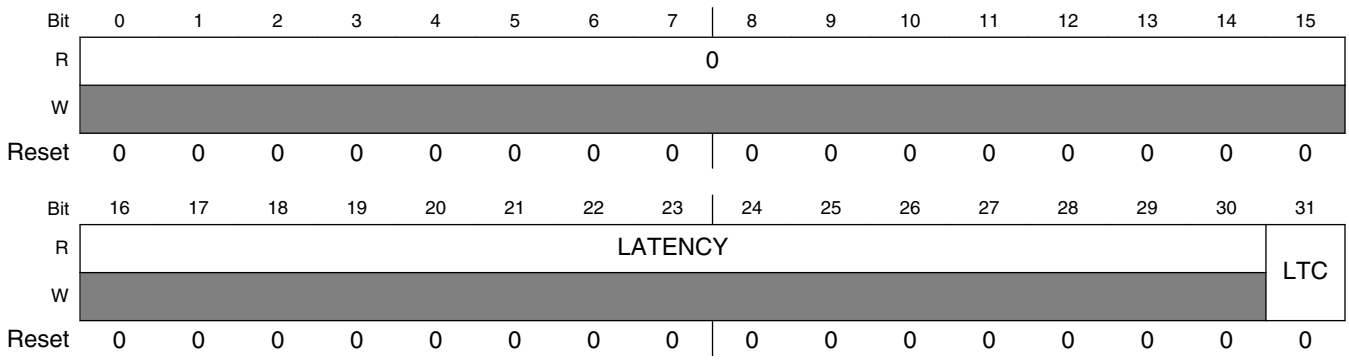
This field specifies the maximum latency monitored, in microcycles, before a latency timeout. If MAXLAT=0, the latency keeps on being counted and measured, but the comparison with MAXLAT asserts neither the LTO1/2 fields in eTPU Module Configuration Register (eTPU_SCR_MCR) nor issues a Global Exception. See Latency Monitor for the detailed latency timeout description.

NOTE: MAXLAT must only be changed when the Latency Monitor is in disabled mode.</td></tr>
</table>

2.9.2 eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM*n*MR_ENG1)

This register holds the latency measured value.

Address: 0h base + 84h offset + (8d × i), where i=0d to 5d



eTPU_LMR_LM*n*MR_ENG1 field descriptions

Field	Description
0–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16–30 LATENCY	Latency measurement. This read-only field holds the value of the greatest latency measured, in microcycles, since reset or the latest clear. When the value is greater than the maximum specified in MAXLAT and MAXLAT value is not zero, a Global Exception is issued and LTO1 (engine 1) or LTO2 (engine 2) is set. See Latency Monitor for the detailed latency measurement description. **NOTE:** When read, the LATENCY field is concatenated with the LTC bit (LSB) to form a 16-bit value. **NOTE:** Any latency measured above 0xFFFF is captured as 0xFFFF into the LATENCY+LTC field.
31 LTC	Latency measurement clear Writing a 1 to the LTC bit clears the LATENCY field. **NOTE:** When read, the LATENCY field is concatenated with the LTC bit (LSB) to form a 16-bit value. **NOTE:** The LATENCY value is not automatically cleared when the Latency Monitor is disabled. 0 Keep the LATENCY field value. 1 Clear the LATENCY field value.

2.9.3 eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM*n*CR_ENG2)

This register configures the operation of individual latency monitors.

Address: 0h base + C0h offset + (8d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LMM		0			SRSEL			0			MONCHAN					MAXLAT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_LMR_LM*n*CR_ENG2 field descriptions

Field	Description
0–1 LMM	Latency Monitor Mode. This field selects the operation mode of the latency monitor. See the Latency Monitor section for detailed mode descriptions. **NOTE:** LMM must be in disabled mode to be changed to channel or priority mode. CPR must not be changed while LMM of any Latency Monitor is in priority mode. 00 Disabled mode 01 Channel mode 10 Priority mode 11 Reserved
2–4 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SRSEL	Service Request Selection This field selects which service requests are monitored. The SRSEL value has different meanings in channel and priority modes, according to the following table.

Table 2-20. SRSEL - Service Request Selection

SRSEL	Channel Mode	Channel Mode
001	Link SRs monitored only	Low priority monitored
010	Channel SRs monitored only	Medium priority monitored
100	HSRs monitored only	High priority monitored
000	No SRs monitored	Reserved
011	Channel and Link SRs monitored	
101	HSRs and Link SRs monitored	
110	HSRs and Channel SRs monitored	
111	All SRs monitored	

Table continues on the next page...

eTPU_LMR_LM*n*CR_ENG2 field descriptions (continued)

Field	Description
8–10 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
11–15 MONCHAN	Monitored Channel. When the latency monitor is configured for channel mode, this field selects which channel has the latency monitored. MONCHAN has no meaning in priority mode. **NOTE:** MONCHAN must only be changed when the Latency Monitor is in disabled mode.
16–31 MAXLAT	Maximum Latency. This field specifies the maximum latency monitored, in microcycles, before a latency timeout. If MAXLAT=0, the latency keeps on being counted and measured, but the comparison with MAXLAT asserts neither the LTO1/2 fields in eTPU Module Configuration Register (eTPU_SCR_MCR) nor issues a Global Exception. See Latency Monitor for the detailed latency timeout description. **NOTE:** MAXLAT must only be changed when the Latency Monitor is in disabled mode.

2.9.4 eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM*n*MR_ENG2)

This register holds the latency measured value.

Address: 0h base + C4h offset + (8d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LATENCY															LTC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_LMR_LM*n*MR_ENG2 field descriptions

Field	Description
0–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16–30 LATENCY	Latency measurement.

Table continues on the next page...

eTPU_LMR_LM*n*MR_ENG2 field descriptions (continued)

Field	Description
	This read-only field holds the value of the greatest latency measured, in microcycles, since reset or the latest clear. When the value is greater than the maximum specified in MAXLAT and MAXLAT value is not zero, a Global Exception is issued and LTO1 (engine 1) or LTO2 (engine 2) is set. See Latency Monitor for the detailed latency measurement description. **NOTE:** When read, the LATENCY field is concatenated with the LTC bit (LSB) to form a 16-bit value. **NOTE:** Any latency measured above 0xFFFF is captured as 0xFFFF into the LATENCY+LTC field.
31 LTC	Latency measurement clear. Writing a 1 to the LTC bit clears the LATENCY field. **NOTE:** When read, the LATENCY field is concatenated with the LTC bit (LSB) to form a 16-bit value. **NOTE:** The LATENCY value is not automatically cleared when the Latency Monitor is disabled. 0 Keep the LATENCY field value. 1 Clear the LATENCY field value.

2.10 Memory Error Support memory map and registers

eTPU_MESR memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
100	eTPU Memory Error Control Register (eTPU_MESR_MECR)	32	R/W	0000_0000h	2.10.1/134
104	eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR)	32	R/W	0000_0000h	2.10.2/137
108	eTPU Data Error Injection Data Pattern Register (eTPU_MESR_DEIDPR)	32	R/W	0000_0000h	2.10.3/138
10C	eTPU Data Error Injection Parity Pattern Register (eTPU_MESR_DEIPPR)	32	R/W	0000_0000h	2.10.4/138
110	eTPU Data Error Report Address Register (eTPU_MESR_DERAR)	32	R	0000_0000h	2.10.5/140
114	eTPU Data Error Report Data Register (eTPU_MESR_DERDR)	32	R	0000_0000h	2.10.6/141
118	eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR)	32	R	0000_0000h	2.10.7/142
120	eTPU Memory Error Status Register (eTPU_MESR_MESR)	32	R/W	0000_0000h	2.10.8/144
124	eTPU Code Error Injection Address Register (eTPU_MESR_CEIAR)	32	R/W	0000_0000h	2.10.9/147
128	eTPU Code Error Injection Data Pattern Register (eTPU_MESR_CEIDPR)	32	R/W	0000_0000h	2.10.10/148
12C	eTPU Code Error Injection Parity Pattern Register (eTPU_MESR_CEIPPR)	32	R/W	0000_0000h	2.10.11/148

Table continues on the next page...

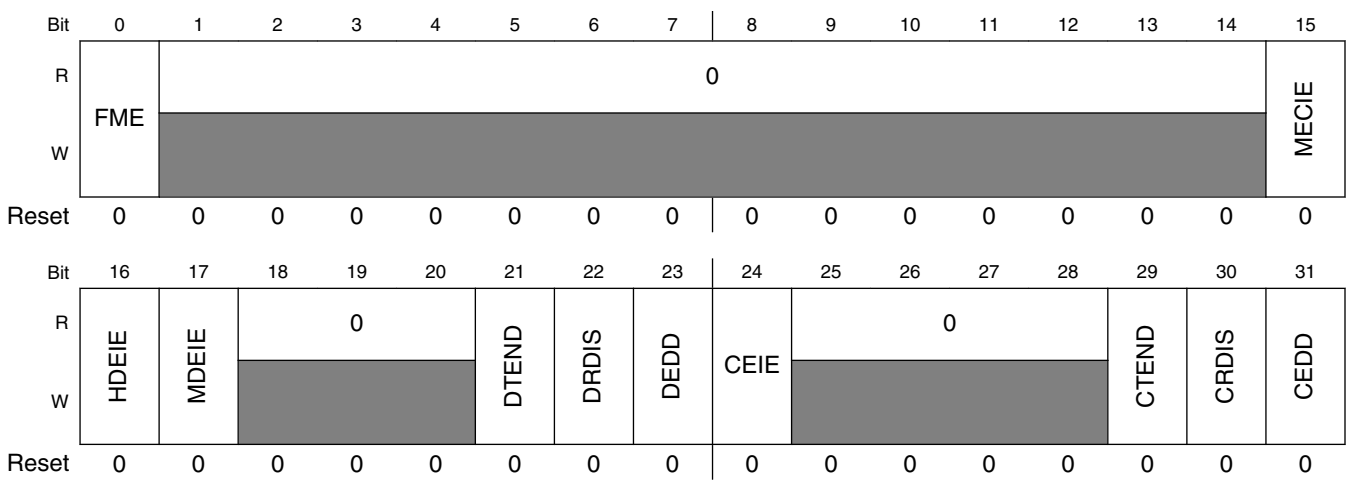
eTPU_MESR memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
130	eTPU Code Error Report Address Register (eTPU_MESR_CERAR)	32	R	0000_0000h	2.10.12/149
134	eTPU Code Error Report Data Register (eTPU_MESR_CERDR)	32	R	0000_0000h	2.10.13/150
138	eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR)	32	R	0000_0000h	2.10.14/151
13C	eTPU Code Error Fix Register (eTPU_MESR_CEFR)	32	R/W	0000_0000h	2.10.15/152

2.10.1 eTPU Memory Error Control Register (eTPU_MESR_MECR)

This register gathers control bits of both SCM and SDM error detection and correction support.

Address: 0h base + 100h offset = 100h



eTPU_MESR_MECR field descriptions

Field	Description
0 FME	Field Modification Enable. Negation of this bit disables the modification of all fields in the register, except MECIE, DRDIS, CRDIS and FME itself: Write accesses do not cause a bus error, regardless of FME state. 0 Fields cannot be modified during a write access, except MECIE, DRDIS, CRDIS and FME. 1 All fields can be modified during a write access.
1–14 Reserved	Reserved.

Table continues on the next page...

eTPU_MESR_MECR field descriptions (continued)

Field	Description
	This field is reserved. This read-only field is reserved and always has the value 0.
15 MECIE	Memory Error Correction Interrupt Enable. This bit enables the Memory Error Correction Interrupt (see Interrupt types and sources). 0 Memory Error Correction Interrupt disabled. 1 Memory Error Correction Interrupt enabled.
16 HDEIE	Host Data Error Injection Enable This bit enables error injection into the SDM host read path, including CDC transfers. The eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR) specifies the address where the error is injected, while eTPU Data Error Injection Data Pattern Register (eTPU_MESR_DEIDPR) and eTPU Data Error Injection Parity Pattern Register (eTPU_MESR_DEIPPR) define the bit flips on data and parity, respectively. **NOTE:** Data error injection is also performed when data error detection is disabled (DEDD=1). 0 SDM error injection disabled for host reads. 1 SDM error injection enabled for host reads.
17 MDEIE	Microengine(s) Data Error Injection Enable. This bit enables error injection into the SDM microengine read path. The eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR) specifies the address where the error is injected, while eTPU Data Error Injection Data Pattern Register (eTPU_MESR_DEIDPR) and eTPU Data Error Injection Parity Pattern Register (eTPU_MESR_DEIPPR) define the bit flips on data and parity, respectively. **NOTE:** Data error injection is also performed when data error detection is disabled (DEDD=1). 0 SDM error injection disabled for microengine reads. 1 SDM error injection enabled for microengine reads.
18–20 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
21 DTEND	Thread End on Data Non-correctable Error. This bit controls the behavior of the microengine when a non-correctable data error occurs. For more details, see Memory error support. **NOTE:** DTEND must only be changed when the engine is stopped or halted, or when all channels are disabled. 0 Thread continues on non-correctable data read access by the microengine. 1 Thread ends on non-correctable data read access by the microengine.
22 DRDIS	Data Error Report Disable. This bit disables the sampling of SDM error address, data and syndrome into the eTPU Memory Error Support Registers, and eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR). It does not, however, disable the setting of eTPU Memory Error Status Register (eTPU_MESR_MESR) bits DCERR and DNCERR.

Table continues on the next page...

eTPU_MESR_MECR field descriptions (continued)

Field	Description
	NOTE: To guarantee coherent reading of eTPU Data Error Report Address Register (eTPU_MESR_DERAR), eTPU Data Error Report Data Register (eTPU_MESR_DERDR), and eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR), DRDIS must be asserted beforehand if SDM concurrent accesses can happen, by host or microengine. After the registers are read, DRDIS may be negated again to allow new samplings. 0 SDM error report sampling enabled. 1 SDM error report sampling disabled.
23 DEDD	Data Error Detection Disable. This bit disables SDM error detection. The disabled state turns ineffective any action due to error detection, including consequently error correction. 0 SDM error detection enabled. 1 SDM error detection disabled.
24 CEIE	Code Error Injection Enable. This bit enables error injection into the SCM read path, either on microengine accesses (see VIS =0) or host accesses (VIS = 1) eTPU Code Error Injection Address Register (eTPU_MESR_CEIAR) specifies the address where the error is injected, while eTPU Code Error Injection Data Pattern Register (eTPU_MESR_CEIDPR) and eTPU Code Error Injection Parity Pattern Register (eTPU_MESR_CEIPPR) define the bit flips on data and parity, respectively. **NOTE:** Code error injection is also performed when code error detection is disabled (CEDD=1). 0 SCM error injection disabled. 1 SCM error injection enabled.
25–28 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
29 CTEND	Thread End on Code Non-correctable Error. This bit controls the behavior of the microengine when a non-correctable code error occurs. For more details, see Memory error support. **NOTE:** CTEND must only be changed when the engine is stopped or halted, or when all channels are disabled. 0 Thread continues on instruction fetch non-correctable error. 1 Thread ends on instruction fetch non-correctable error.
30 CRDIS	Code Error Report Disable. This bit disables the sampling of SCM error address, data and syndrome into the eTPU Code Error Report Address Register (eTPU_MESR_CERAR), eTPU Code Error Report Data Register (eTPU_MESR_CERDR), and eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR). It does not, however, disable the setting of CCERR and CNCERR in eTPU Memory Error Status Register (eTPU_MESR_MESR).

Table continues on the next page...

eTPU_MESR_MECR field descriptions (continued)

Field	Description
	NOTE: To guarantee coherent reading of eTPU Code Error Report Address Register (eTPU_MESR_CERAR), eTPU Code Error Report Data Register (eTPU_MESR_CERDR), and eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR), CRDIS must be asserted beforehand if SCM concurrent accesses can happen, by host or microengine. After the registers are read, CRDIS may be negated again to allow new samplings. 0 SCM error report sampling enabled. 1 SCM error report sampling disabled.
31 CEDD	Code Error Detection Disable. This bit disables SCM error detection. The disabled state turns ineffective any action due to error detection, including consequently error correction. 0 SCM error detection enabled. 1 SCM error detection disabled.

2.10.2 eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR)

This register holds the address where error is to be injected into the SDM read data path. The two least significant bits are 0 (read-only), so that the address is always word-aligned.

NOTE

This register must only be modified when HDEIE and MDEIE are both negated.

Address: 0h base + 104h offset = 104h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																INJ_ADDR														0	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_DEIAR field descriptions

Field	Description
0–17 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
18–29 INJ_ADDR	Injection Address. INJ_ADDR holds the (word) address where an error is injected into the SDM read data path, as specified by eTPU Data Error Injection Data Pattern Register (eTPU_MESR_DEIDPR) and eTPU Data Error

Table continues on the next page...

eTPU_MESR_DEIAR field descriptions (continued)

Field	Description
	Injection Parity Pattern Register (eTPU_MESR_DEIPPR). For more information on error injection, see Error injection.
30–31 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

2.10.3 eTPU Data Error Injection Data Pattern Register (eTPU_MESR_DEIDPR)

This register is used to specify the bits in which error is injected into the SDM read data path.

NOTE

This register must only be modified when HDEIE and MDEIE are both negated.

Address: 0h base + 108h offset = 108h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DFLIP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_DEIDPR field descriptions

Field	Description
0–31 DFLIP	Data Flip bits. Each bit corresponds to a bit in the actual data read from the SDM address defined in eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR), specifying whether it should be inverted or not. The inversion only occurs if the SDM error injection is enabled by HDEIE and/or MDEIE in eTPU Memory Error Control Register (eTPU_MESR_MECR). For more information on error injection, see Error injection. 0 Read data bit is not inverted. 1 Read data bit is inverted.

2.10.4 eTPU Data Error Injection Parity Pattern Register (eTPU_MESR_DEIPPR)

This register is used to specify the bits in which error is injected into the SDM read parity path.

NOTE

This register must only be modified when HDEIE and MDEIE are both negated.

Address: 0h base + 10Ch offset = 10Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			PFLIP_19_15					0			PFLIP_14_10					0			PFLIP_9_5					0			PFLIP_4_0				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_DEIPPR field descriptions

Field	Description
0–2 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
3–7 PFLIP_19_15	Parity Flip bits. Each bit corresponds to a bit in the actual parity read from the SDM address defined in eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR), specifying whether it should be inverted or not. The inversion only occurs if the SDM error injection is enabled by the HDEIE and/or MDEIE fields in eTPU Memory Error Control Register (eTPU_MESR_MECR). Bits PFLIP[19:15] corresponds to the parity bits for the most significant byte, PFLIP[14:10] to the next most significant byte and so on until PFLIP[4:0] for the least significant byte. For more information on error injection, see Error injection. 0 Parity data bit is not inverted. 1 Parity data bit is inverted.
8–10 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
11–15 PFLIP_14_10	Parity Flip bits. Each bit corresponds to a bit in the actual parity read from the SDM address defined in eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR), specifying whether it should be inverted or not. The inversion only occurs if the SDM error injection is enabled by the HDEIE and/or MDEIE fields in eTPU Memory Error Control Register (eTPU_MESR_MECR). Bits PFLIP[19:15] corresponds to the parity bits for the most significant byte, PFLIP[14:10] to the next most significant byte and so on until PFLIP[4:0] for the least significant byte. For more information on error injection, see Error injection. 0 Parity data bit is not inverted. 1 Parity data bit is inverted.
16–18 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
19–23 PFLIP_9_5	Parity Flip bits. Each bit corresponds to a bit in the actual parity read from the SDM address defined in eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR), specifying whether it should be inverted or not. The inversion only occurs if the SDM error injection is enabled by the HDEIE and/or MDEIE fields in eTPU Memory Error Control Register (eTPU_MESR_MECR). Bits PFLIP[19:15] corresponds to the parity bits for

Table continues on the next page...

eTPU_MESR_DEIPPR field descriptions (continued)

Field	Description
	the most significant byte, PFLIP[14:10] to the next most significant byte and so on until PFLIP[4:0] for the least significant byte. For more information on error injection, see Error injection. 0 Parity data bit is not inverted. 1 Parity data bit is inverted.
24–26 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–31 PFLIP_4_0	Parity Flip bits. Each bit corresponds to a bit in the actual parity read from the SDM address defined in eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR), specifying whether it should be inverted or not. The inversion only occurs if the SDM error injection is enabled by the HDEIE and/or MDEIE fields in eTPU Memory Error Control Register (eTPU_MESR_MECR). Bits PFLIP[19:15] corresponds to the parity bits for the most significant byte, PFLIP[14:10] to the next most significant byte and so on until PFLIP[4:0] for the least significant byte. For more information on error injection, see Error injection. 0 Parity data bit is not inverted. 1 Parity data bit is inverted.

2.10.5 eTPU Data Error Report Address Register (eTPU_MESR_DERAR)

This register holds the address where the last non-correctable error was detected from the SDM It also captures the address of the last correctable error detected, as long as DNCERR is 0. The two least significant bits are 0 (read-only), so that the address is always word-aligned.

Address: 0h base + 110h offset = 110h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR_ACC		0	ERR_CHANNEL					0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		ERR_ADDR												0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_DERAR field descriptions

Field	Description
0–1 ERR_ACC	Error Access Source Indication. This two-bit field indicates the source of the access when the error occurred. 00 Host - direct access 01 Engine 1 10 Engine 2 11 CDC
2 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
3–7 ERR_CHANNEL	Error Channel Access Indication. This field holds the number of the channel being serviced (not necessarily CHAN) when the error occurred. Its content is meaningful only when the field ERR_ACC indicates Engine 1 or Engine 2.
8–17 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
18–29 ERR_ADDR	Error Address. ERR_ADDR holds the (word) address where the last captured SDM read error was detected. The last captured error is a non-correctable one if DNCERR =1, or a correctable one if DNCERR =0 and DCERR =1. The field is meaningless if both DNCERR and DCERR are 0. For more information on error detection, see Memory error support.
30–31 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

2.10.6 eTPU Data Error Report Data Register (eTPU_MESR_DERDR)

This register holds the data read with the last non-correctable error detected from the SDM. It also captures the data of the last correctable error read detection, as long as DNCERR is 0. In both cases, the data is captured after the error correction logic. The data is meaningless if both DNCERR and DCERR are 0. For more information on error detection, see Memory error support.

NOTE

Data is sampled before the error correction logic.

Address: 0h base + 114h offset = 114h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_DERDR field descriptions

Field	Description
0–31 DATA	DATA. **NOTE:** Individual data bytes are not meaningful when their respective BE bits in eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR) are not asserted.

2.10.7 eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR)

This register holds the syndrome read with the last non-correctable error detected from the SDM. It also captures the syndrome of the last correctable error read detection, as long as DNCERR is 0.

Address: 0h base + 118h offset = 118h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BE3	0		SYND3					BE2	0		SYND2				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BE1	0		SYND1					BE0	0		SYND0				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_DERSR field descriptions

Field	Description
0 BE3	Byte Enable 3. Each BE bit corresponds to a report byte in a 32-bit aligned data word. BE3 corresponds to the most significant one. BE qualifies the error report for its respective byte. 0 Byte was not selected for read, therefore its respective error report fields are not meaningful. 1 Byte was selected for read, and the error report for the respective byte is meaningful.
1–2 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
3–7 SYND3	Error Syndrome 3. SYND3 holds the syndrome of an error detected in the SDM read data path, as defined in Error Correction Code (ECC) and syndrome definition. The field is meaningless if both DNCERR and DCERR are 0. For more information on error detection, see Memory error support.

Table continues on the next page...

eTPU_MESR_DERSR field descriptions (continued)

Field	Description
	NOTE: Syndrome fields which have their respective BE bits negated are not meaningful.
8 BE2	Byte Enable 2. Each BE bit corresponds to a report byte in a 32-bit aligned data word. BE3 corresponds to the most significant one. BE qualifies the error report for its respective byte. 0 Byte was not selected for read, therefore its respective error report fields are not meaningful. 1 Byte was selected for read, and the error report for the respective byte is meaningful.
9–10 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
11–15 SYND2	Error Syndrome 2. SYND2 holds the syndrome of an error detected in the SDM read data path, as defined in the “Error Correction Code (ECC) and Syndrome Definition”section. The field is meaningless if both DNCERR and DCERR are 0. For more information on error detection, see Memory error support. **NOTE:** Syndrome fields which have their respective BE bits negated are not meaningful.
16 BE1	Byte Enable 1. Each BE bit corresponds to a report byte in a 32-bit aligned data word. BE3 corresponds to the most significant one. BE qualifies the error report for its respective byte. 0 Byte was not selected for read, therefore its respective error report fields are not meaningful. 1 Byte was selected for read, and the error report for the respective byte is meaningful.
17–18 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
19–23 SYND1	Error Syndrome 1. SYND1 holds the syndrome of an error detected in the SDM read data path, as defined in the “Error Correction Code (ECC) and Syndrome Definition”section. The field is meaningless if both DNCERR and DCERR are 0. For more information on error detection, see Memory error support. **NOTE:** Syndrome fields which have their respective BE bits negated are not meaningful.
24 BE0	Byte Enable 0. Each BE bit corresponds to a report byte in a 32-bit aligned data word. BE3 corresponds to the most significant one. BE qualifies the error report for its respective byte. 0 Byte was not selected for read, therefore its respective error report fields are not meaningful. 1 Byte was selected for read, and the error report for the respective byte is meaningful.
25–26 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–31 SYND0	Error Syndrome 0.

Table continues on the next page...

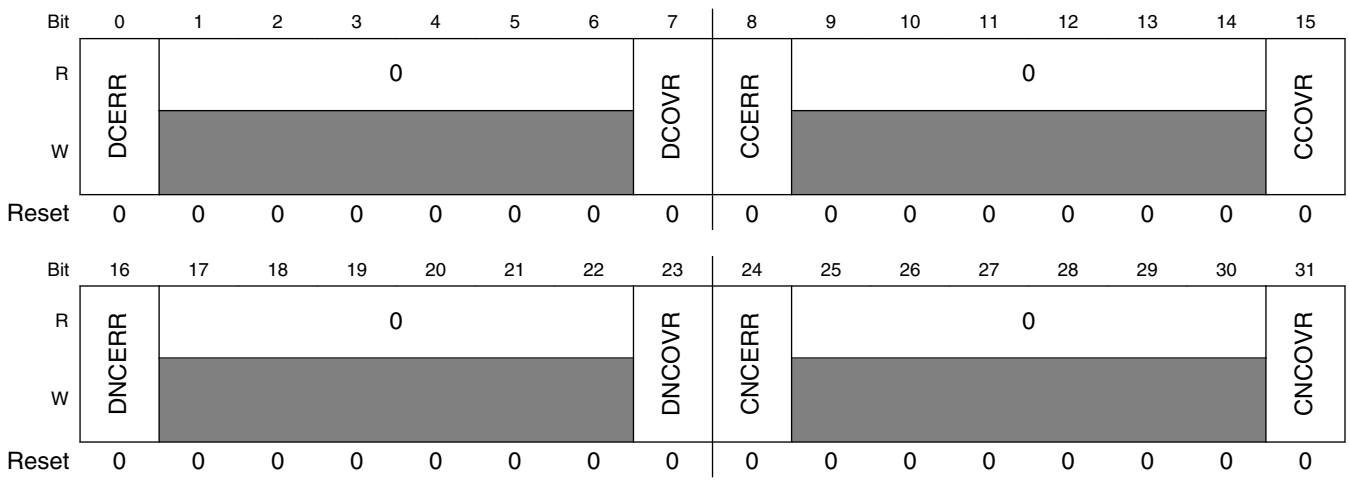
eTPU_MESR_DERSR field descriptions (continued)

Field	Description
	SYND0 holds the syndrome of an error detected in the SDM read data path, as defined in the “Error Correction Code (ECC) and Syndrome Definition”section. The field is meaningless if both DNCERR and DCERR are 0. For more information on error detection, see Memory error support. **NOTE:** Syndrome fields which have their respective BE bits negated are not meaningful.

2.10.8 eTPU Memory Error Status Register (eTPU_MESR_MESR)

This register gathers status bits of both SCM and SDM error detection and correction support.

Address: 0h base + 120h offset = 120h



eTPU_MESR_MESR field descriptions

Field	Description
0 DCERR	Data Correctable Error flag. When read: 0.....No SDM correctable error occurred. 1.....An SDM correctable error occurred. Writing 1 to this bit clears the DCERR status flag. DCERR asserts even if DNCERR is asserted beforehand, but in this case the address and data of the erroneous access is not captured (see Memory Error Support memory map and registers and following sections). DCERR is cleared by writing a 1 to it. Assertion of this bit can also assert the Memory Error Correction Interrupt (see Interrupt types and sources). **NOTE:** A correctable error is not considered for flagging if accompanied of a non-correctable one. Therefore, if both correctable and non-correctable data errors occur on the same access (in distinct bytes), the DNCERR flag asserts, but not the DCERR flag.

Table continues on the next page...

eTPU_MESR_MESR field descriptions (continued)

Field	Description
	0 Do not clear DCERR flag. 1 Clears DCERR flag
1–6 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
7 DCOVR	Data Correctable Overrun flag. Writing 1 to this bit clears the DCOVR status flag. 0 Data correctable error occurred while DCOVR = 1. 1 No data correctable error occurred while DCOVR = 1. DCOVR asserts when an SDM correctable error occurs and the DCERR flag is already asserted. Assertion of this bit causes no interrupt. **NOTE:** A correctable error is not considered for flagging if accompanied by a non-correctable one. Therefore, if both correctable and non-correctable data errors occur on the same access (in distinct bytes), the DCOVR does not assert, even if DCERR is asserted. The DNCOVR flag, on the other hand, asserts if DNCERR is asserted.
8 CCERR	Code Error flag. When read: 0.....No SCM correctable error occurred. 1.....An SCM correctable error occurred. When written: 0.....Do not clear CCERR flag. 1.....Clears CCERR flag. The CCERR flag asserts even if CNCERR is asserted beforehand, but in this case the address and data of the erroneous access is not captured (see eTPU Code Error Report Address Register (eTPU_MESR_CERAR) and the following sections). CCERR is cleared by writing a 1 at the same position. Assertion of this bit can also assert the Memory Error Correction Interrupt (see Interrupt types and sources).
9–14 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
15 CCOVR	Code Correctable Overrun flag This bit asserts when an SCM correctable error occurs and the CCERR flag is already asserted. Assertion of this bit causes no interrupt. When read: 0.....No SCM correctable error overrun occurred. 1.....An SCM correctable error overrun occurred. Writing 1 to this bit clears the CCOVR status flag: 0 Do not clear CCOVR flag. 1 Clear CCOVR flag.

Table continues on the next page...

eTPU_MESR_MESR field descriptions (continued)

Field	Description
16 DNCERR	Data Non-Correctable Error flag DNCERR flags a non-correctable error. When read: 0.....No SDM non-correctable error occurred. 1.....An SDM non-correctable error occurred; registers eTPU Global Channel Registers are not updated by correctable error detections. DNCERR must be cleared (by writing a 1 to it) to allow correctable error updates to eTPU Data Error Report Address Register (eTPU_MESR_DERAR) and eTPU Data Error Report Data Register (eTPU_MESR_DERDR). 0 Do not clear DNCERR flag. 1 Clear DNCERR flag.
17–22 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
23 DNCOVR	Data Non-Correctable Overrun flag. DNCOVR asserts when an SDM non-correctable error occurs and the DNCERR flag is already asserted. Assertion of this bit causes no interrupt. When read: 0.....No SDM non-correctable error overrun occurred. 1.....An SDM non-correctable error overrun occurred. DNCOVR is cleared by writing a 1 to it. 0 Do not clear DNCOVR flag. 1 Clear DNCOVR flag.
24 CNCERR	Code Non-Correctable Error flag CNCERR flags a non-correctable error: When read: 0.....An SCM non-correctable error occurred; the eTPU Global Channel Registers are not updated by correctable error detections. 1....an SCM non-correctable error occurred; eTPU Code Error Report Address Register (eTPU_MESR_CERAR) and eTPU Code Error Report Data Register (eTPU_MESR_CERDR) are not updated by correctable error detections. CNCERR must be cleared to allow correctable error updates to eTPU Code Error Report Address Register (eTPU_MESR_CERAR) and eTPU Code Error Report Data Register (eTPU_MESR_CERDR). CNCERR is cleared by writing a 1 to it. 0 Do not clear CNCERR flag. 1 Clear CNCERR flag.
25–30 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

eTPU_MESR_MESR field descriptions (continued)

Field	Description
31 CNCOVR	Code Non-Correctable Overrun flag. CNCOVR asserts when an SCM non-correctable error occurs and the CNCERR flag is already asserted. Assertion of this bit causes no interrupt. When read: 0.....an SCM non-correctable error overrun occurred. 1.....An SCM non-correctable error overrun occurred. When written: 0 Do not clear CNCOVR flag. 1 Clear CNCOVR flag.

2.10.9 eTPU Code Error Injection Address Register (eTPU_MESR_CEIAR)

This register holds the address where error is to be injected into the SCM read data path. The two least significant bits are 0 (read-only), so that the address is always word-aligned.

NOTE

This register must only be modified when CEIE is negated.

Address: 0h base + 124h offset = 124h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																INJ_ADDR														0	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_CEIAR field descriptions

Field	Description
0–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16–29 INJ_ADDR	Injection Address. INJ_ADDR holds the (word) address where an error is injected in the SCM read data path, as specified by eTPU Code Error Injection Data Pattern Register (eTPU_MESR_CEIDPR) and eTPU Code Error Injection Parity Pattern Register (eTPU_MESR_CEIPPR). For more information on error injection, see Error injection.
30–31 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

2.10.10 eTPU Code Error Injection Data Pattern Register (eTPU_MESR_CEIDPR)

This register is used to specify the bits in which error is injected into the SCM read data path.

NOTE

This register must only be modified when CEIE is negated.

Address: 0h base + 128h offset = 128h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DFLIP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_CEIDPR field descriptions

Field	Description
0–31 DFLIP	Data Flip bits. Each bit corresponds to a bit in the actual data read from the SCM address defined in eTPU Code Error Injection Address Register (eTPU_MESR_CEIAR), specifying whether it should be inverted or not. The inversion only occurs if the SCM error injection is enabled by CEIE. For more information on error injection, see Error injection. 0 Read data bit is not inverted. 1 Read data bit is inverted.

2.10.11 eTPU Code Error Injection Parity Pattern Register (eTPU_MESR_CEIPPR)

This register is used to specify the bits in which error is injected into the SCM read parity path.

NOTE

This register must only be modified when CEIE is negated.

Address: 0h base + 12Ch offset = 12Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																									PFLIP						
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_CEIPPR field descriptions

Field	Description
0–24 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
25–31 PFLIP	Parity Flip bits. Each bit corresponds to a bit in the actual parity/code read from the SCM address defined in eTPU Code Error Injection Address Register (eTPU_MESR_CEIAR), specifying whether it should be inverted or not. The inversion only occurs if the SCM error injection is enabled by CEIE. For more information on error injection, see Error injection. 0 Parity data bit is not inverted. 1 Parity data bit is inverted.

2.10.12 eTPU Code Error Report Address Register (eTPU_MESR_CERAR)

This register holds the address where the last non-correctable error was detected from the SCM. It also captures the address of the last correctable error detected, as long as CNCERR is 0. The two least significant bits are 0 (read-only), so that the address is always word-aligned.

Address: 0h base + 130h offset = 130h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR_ACC		0	ERR_CHANNEL					0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR_ADDR														0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_CERAR field descriptions

Field	Description
0–1 ERR_ACC	Error Access Source Indication. This two-bit field indicates the source of the access when the error occurred. 00 Host 01 Engine 1

Table continues on the next page...

eTPU_MESR_CERAR field descriptions (continued)

Field	Description
	10 Engine 2 11 MISC
2 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
3–7 ERR_CHANNEL	Error Channel Access Indication. This field holds the number of the channel being serviced (not necessarily CHAN) when the error occurred. Its content is meaningful only when the field ERR_ACC indicates Engine 1 or Engine 2.
8–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16–29 ERR_ADDR	Error Address. ERR_ADDR holds the (word) address where the last captured SCM read error was detected. The last captured error is a non-correctable one if CNCERR = 1, or a correctable one if CNCERR = 0 and CCERR = 1. The field is meaningless if both CNCERR and CCERR are 0. For more information on error detection, see Memory error support.
30–31 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

2.10.13 eTPU Code Error Report Data Register (eTPU_MESR_CERDR)

This register holds the data read with the last non-correctable error detected from the SCM. It also captures the data of the last correctable error read detection, as long as CNCERR is 0. In both cases, the data is captured after the error correction logic. The data may be meaningless if both CNCERR and CCERR are 0. For more information on error detection, see Memory error support.

NOTE

CODE is sampled before the error correction logic.

Address: 0h base + 134h offset = 134h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CODE																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_CERDR field descriptions

Field	Description
0–31 CODE	CODE[31-0].

2.10.14 eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR)

This register holds the syndrome and global parity read with the last non-correctable error detected from the SCM. It also captures the syndrome of the last correctable error read detection, as long as CNCERR is 0.

Address: 0h base + 138h offset = 138h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																									SYND						
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

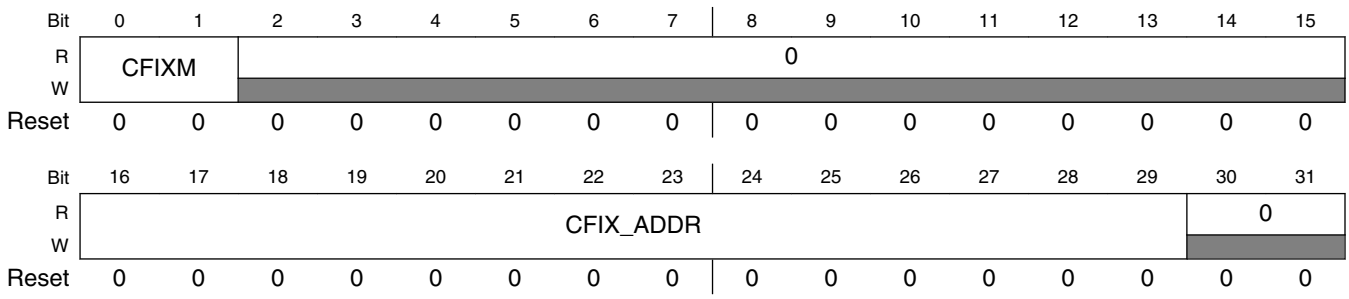
eTPU_MESR_CERSR field descriptions

Field	Description
0–24 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
25–31 SYND	Error Syndrome. SYND holds the syndrome of an error detected in the SCM read data path, as defined in Error Correction Code (ECC) and syndrome definition. The field is meaningless if both CNCERR and CCERR are 0. For more information on error detection, see Memory error support.

2.10.15 eTPU Code Error Fix Register (eTPU_MESR_CEFR)

This register is used to fix single-bit error in a specified SCM position. The fix is done automatically, concurrently with code execution, by the MISC logic (see Error fixing).

Address: 0h base + 13Ch offset = 13Ch



eTPU_MESR_CEFR field descriptions

Field	Description
0–1 CFIXM	Code Fix Mode. This field specifies the SCM MISC fix logic operation mode as shown in the table below. **NOTE:** The CFIXM field can only be changed to/from disabled mode. **Table 2-21. CFIXM - Code Fix Mode** Value / Mode / Description: 00 / Disabled / SCM error fix disabled. 01 / Reserved / Reserved value, must not be used. 10 / Reserved / Reserved value, must not be used. 11 / Automatic / The MISC logic rewrites the code word value read (already fixed by the error correction mechanism) into the same address whenever it finds a correctable error.
2–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16–29 CFIX_ADDR	Code Fix Address. This field is used for hardware debug purposes. CFIX_ADDR may change its contents and must not be written in automatic mode. For more information on error detection and correction, see Memory error support.
30–31 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

2.11 Global Channel memory map and registers

The registers in this section group, by type, the interrupt status and enable bits from all the channels. This organization eases management of all channels or groups of channels by a single interrupt handler routine. These bits, except the service and watchdog status, are mirrored in the individual channel registers, grouped by channel.

eTPU_GCR memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
200	eTPU Channel Interrupt Status Register for Engine 1 (eTPU_GCR_CISR_ENG1)	32	R/W	0000_0000h	2.11.1/154
204	eTPU Channel Interrupt Status Register for Engine 2 (eTPU_GCR_CISR_ENG2)	32	R/W	0000_0000h	2.11.2/154
210	eTPU Channel Data Transfer Request Status Register for Engine 1 (eTPU_GCR_CDTRSR_ENG1)	32	R/W	0000_0000h	2.11.3/155
214	eTPU Channel Data Transfer Request Status Register for Engine 2 (eTPU_GCR_CDTRSR_ENG2)	32	R/W	0000_0000h	2.11.4/156
220	eTPU Channel Interrupt Overflow Status Register for Engine 1 (eTPU_GCR_CIOSR_ENG1)	32	R/W	0000_0000h	2.11.5/156
224	eTPU Channel Interrupt Overflow Status Register for Engine 2 (eTPU_GCR_CIOSR_ENG2)	32	R/W	0000_0000h	2.11.6/157
230	eTPU Channel Data Transfer Request Overflow Status Register for Engine 1 (eTPU_GCR_CDTROSR_ENG1)	32	R/W	0000_0000h	2.11.7/158
234	eTPU Channel Data Transfer Request Overflow Status Register for Engine 2 (eTPU_GCR_CDTROSR_ENG2)	32	R/W	0000_0000h	2.11.8/158
240	eTPU Channel Interrupt Enable Register for Engine 1 (eTPU_GCR_CIER_ENG1)	32	R/W	0000_0000h	2.11.9/159
244	eTPU Channel Interrupt Enable Register for Engine 2 (eTPU_GCR_CIER_ENG2)	32	R/W	0000_0000h	2.11.10/159
250	eTPU Channel Data Transfer Request Enable Register for Engine 1 (eTPU_GCR_CDTRER_ENG1)	32	R/W	0000_0000h	2.11.11/160
254	eTPU Channel Data Transfer Request Enable Register for Engine 2 (eTPU_GCR_CDTRER_ENG2)	32	R/W	0000_0000h	2.11.12/160
260	eTPU Watchdog Status Register for Engine 1 (eTPU_GCR_WDSR_ENG1)	32	R/W	0000_0000h	2.11.13/161
264	eTPU Watchdog Status Register for Engine 2 (eTPU_GCR_WDSR_ENG2)	32	R/W	0000_0000h	2.11.14/162
280	eTPU Channel Pending Service Status Register for Engine 1 (eTPU_GCR_CPSSR_ENG1)	32	R	0000_0000h	2.11.15/162
284	eTPU Channel Pending Service Status Register for Engine 2 (eTPU_GCR_CPSSR_ENG2)	32	R	0000_0000h	2.11.16/163
290	eTPU Channel Service Status Register for Engine 1 (eTPU_GCR_CSSR_ENG1)	32	R	0000_0000h	2.11.17/164
294	eTPU Channel Service Status Register for Engine 2 (eTPU_GCR_CSSR_ENG2)	32	R	0000_0000h	2.11.18/164

2.11.1 eTPU Channel Interrupt Status Register for Engine 1 (eTPU_GCR_CISR_ENG1)

This register groups the host interrupt status from all channels.These bits are mirrored from the Channel Status Control register (see eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C*n*SCR_ENG1)) and the Host must write 1 to clear a status bit.

For details about interrupts, see Interrupts and data transfer requests.

Address: 0h base + 200h offset = 200h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CISn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CISR_ENG1 field descriptions

Field	Description
0–31 CISn	Channel n Interrupt Status. When read, each bit: 0.....Indicates that channel n has no pending interrupt to the CPU. 1.....Indicates that channel n has a pending interrupt to the CPU. When written: 0.....Keep interrupt status bit unaltered. 1.....Clear interrupt status bit.

2.11.2 eTPU Channel Interrupt Status Register for Engine 2 (eTPU_GCR_CISR_ENG2)

This register groups the host interrupt status from all channels.These bits are mirrored from the Channel Status Control register (see eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C*n*SCR_ENG2)) and the Host must write 1 to clear a status bit.

For details about interrupts, see Interrupts and data transfer requests.

Address: 0h base + 204h offset = 204h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CISn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CISR_ENG2 field descriptions

Field	Description
0–31 CISn	Channel n Interrupt Status. When read, each bit: 0.....Indicates that channel n has no pending interrupt to the CPU. 1.....Indicates that channel n has a pending interrupt to the CPU. When written: 0.....Keep interrupt status bit unaltered. 1.....Clear interrupt status bit.

2.11.3 eTPU Channel Data Transfer Request Status Register for Engine 1 (eTPU_GCR_CDTRSR_ENG1)

This register groups data transfer request status from all channels. Their bits are mirrored from the Channel Status Control registers (see eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C*n*SCR_ENG1)) and Host must write 1 to clear a status bit.

For details about interrupts, see Interrupts and data transfer requests.

Address: 0h base + 210h offset = 210h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DTRSn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CDTRSR_ENG1 field descriptions

Field	Description
0–31 DTRSn	Channel n Data Transfer Request Status. These bits mimic the corresponding eTPU DMA requests. DTRSn can be cleared by software writing a 1 to it or by the assertion of corresponding DMA completion acknowledge line. When read: 0.....Indicates that channel n has no pending data transfer request. 1.....Indicates that channel x has a pending data transfer request. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.

2.11.4 eTPU Channel Data Transfer Request Status Register for Engine 2 (eTPU_GCR_CDTRSR_ENG2)

This register groups data transfer request status from all channels.These bits are mirrored from the Channel Status Control registers (see eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C*n*SCR_ENG2)) and Host must write 1 to clear a status bit.

For details about interrupts, see Interrupts and data transfer requests.

Address: 0h base + 214h offset = 214h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DTRSn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CDTRSR_ENG2 field descriptions

Field	Description
0–31 DTRSn	Channel n Data Transfer Request Status. These bits mimic the corresponding eTPU DMA requests. DTRSn can be cleared by software writing a 1 to it or by the assertion of corresponding DMA completion acknowledge line. When read: 0.....Indicates that channel n has no pending data transfer request. 1.....Indicates that channel x has a pending data transfer request. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.

2.11.5 eTPU Channel Interrupt Overflow Status Register for Engine 1 (eTPU_GCR_CIOSR_ENG1)

This register groups the interrupt overflow status from all channels. These bits are mirrored from the Channel Status Control registers (see eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C*n*SCR_ENG1)) and the Host must write 1 to clear a status bit.

For details about interrupts, see Interrupts and data transfer requests.

Address: 0h base + 220h offset = 220h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CIOSn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CIOSR_ENG1 field descriptions

Field	Description
0–31 CIOSn	Channel n Interrupt Overflow Status When read: 0.....Indicates that no interrupt overflow occurred in the channel. 1.....Indicates that interrupt overflow occurred in the channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.

2.11.6 eTPU Channel Interrupt Overflow Status Register for Engine 2 (eTPU_GCR_CIOSR_ENG2)

This register groups the interrupt overflow status from all channels. These bits are mirrored from the Channel Status Control registers (see eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C*n*SCR_ENG2)) and the Host must write 1 to clear a status bit.

For details about interrupts, see Interrupts and data transfer requests.

Address: 0h base + 224h offset = 224h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CIOSn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CIOSR_ENG2 field descriptions

Field	Description
0–31 CIOSn	Channel n Interrupt Overflow Status When read: 0.....Indicates that no interrupt overflow occurred in the channel. 1.....Indicates that interrupt overflow occurred in the channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.

2.11.7 eTPU Channel Data Transfer Request Overflow Status Register for Engine 1 (eTPU_GCR_CDTROSR_ENG1)

This register groups the data transfer request overflow status from all channels. These bits are mirrored from the Channel Status Control registers (see eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C*n*SCR_ENG1)) and the CPU must write 1 to clear a status bit.

For details about interrupts, see Interrupts and data transfer requests.

Address: 0h base + 230h offset = 230h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DTROSn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CDTROSR_ENG1 field descriptions

Field	Description
0–31 DTROSn	Channel n Data Transfer Request Overflow Status. When read: 0.....Indicates that no data transfer request overflow occurred in the channel. 1.....Indicates that data transfer request overflow occurred in the channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.

2.11.8 eTPU Channel Data Transfer Request Overflow Status Register for Engine 2 (eTPU_GCR_CDTROSR_ENG2)

This register groups the data transfer request overflow status from all channels. These bits are mirrored from the Channel Status Control registers (see eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C*n*SCR_ENG2)) and the CPU must write 1 to clear a status bit.

For details about interrupts, see Interrupts and data transfer requests.

Address: 0h base + 234h offset = 234h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DTROSn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CDTROSR_ENG2 field descriptions

Field	Description
0–31 DTROSn	Channel n Data Transfer Request Overflow Status. When read: 0.....Indicates that no data transfer request overflow occurred in the channel. 1.....Indicates that data transfer request overflow occurred in the channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.

2.11.9 eTPU Channel Interrupt Enable Register for Engine 1 (eTPU_GCR_CIER_ENG1)

This register groups the data transfer request overflow status from all channels. These bits are mirrored from the Channel Status Control registers (see eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C*n*SCR_ENG1)).

For details about interrupts, see Interrupts and data transfer requests.

Address: 0h base + 240h offset = 240h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CIEn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CIER_ENG1 field descriptions

Field	Description
0–31 CIEn	Channel n Interrupt Enable. 0 Interrupt disabled for channel n. 1 Interrupt enabled for channel n.

2.11.10 eTPU Channel Interrupt Enable Register for Engine 2 (eTPU_GCR_CIER_ENG2)

This register groups the data transfer request overflow status from all channels. These bits are mirrored from the Channel Status Control registers (see eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C*n*SCR_ENG2)).

For details about interrupts, see Interrupts and data transfer requests.

Address: 0h base + 244h offset = 244h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CIEn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CIER_ENG2 field descriptions

Field	Description
0–31 CIEn	Channel n Interrupt Enable. 0 Interrupt disabled for channel n. 1 Interrupt enabled for channel n.

2.11.11 eTPU Channel Data Transfer Request Enable Register for Engine 1 (eTPU_GCR_CDTRER_ENG1)

This register groups the data transfer request enable from all channels. These bits are mirrored from the Channel Status Control registers (see eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C*n*SCR_ENG1)).

For details about interrupts, see Interrupts and data transfer requests.

Address: 0h base + 250h offset = 250h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DTREn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CDTRER_ENG1 field descriptions

Field	Description
0–31 DTREn	Channel n Data Transfer Request Enable. 0 Data Transfer request disabled for channel n. 1 Data Transfer request enabled for channel n.

2.11.12 eTPU Channel Data Transfer Request Enable Register for Engine 2 (eTPU_GCR_CDTRER_ENG2)

This register groups the data transfer request enable from all channels. These bits are mirrored from the Channel Status Control registers (see eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C*n*SCR_ENG2)).

For details about interrupts, see Interrupts and data transfer requests.

Address: 0h base + 254h offset = 254h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DTREn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CDTRER_ENG2 field descriptions

Field	Description
0–31 DTREn	Channel n Data Transfer Request Enable. 0 Data Transfer request disabled for channel n. 1 Data Transfer request enabled for channel n.

2.11.13 eTPU Watchdog Status Register for Engine 1 (eTPU_GCR_WDSR_ENG1)

This register indicates the watchdog influence on each of the engine channels.

Address: 0h base + 260h offset = 260h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WDSn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_WDSR_ENG1 field descriptions

Field	Description
0–31 WDSn	Channel n Watchdog Status. For details about Watchdog mechanism, see Watchdog. **NOTE:** The WDSn bits set in case of a watchdog timeout independently of WMO. When read: 0.....Watchdog did not expire during execution of a channel x thread since reset or last WDSn clear. 1.....Watchdog expired during execution of a channel n thread. When written: 0.....Keep watchdog status bit unaltered. 1.....Clear watchdog status bit.

2.11.14 eTPU Watchdog Status Register for Engine 2 (eTPU_GCR_WDSR_ENG2)

This register indicates the watchdog influence on each of the engine channels.

Address: 0h base + 264h offset = 264h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WDSn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_WDSR_ENG2 field descriptions

Field	Description
0–31 WDSn	Channel n Watchdog Status. For details about Watchdog mechanism, see Watchdog. **NOTE:** The WDSn bits set in case of a watchdog timeout independently of WMO. When read: 0.....Watchdog did not expire during execution of a channel x thread since reset or last WDSn clear. 1.....Watchdog expired during execution of a channel n thread. When written: 0.....Keep watchdog status bit unaltered. 1.....Clear watchdog status bit.

2.11.15 eTPU Channel Pending Service Status Register for Engine 1 (eTPU_GCR_CPSSR_ENG1)

This register holds the status of the pending Channel Service Requests (see Functions and threads).

Address: 0h base + 280h offset = 280h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SRn																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CPSSR_ENG1 field descriptions

Field	Description
0–31 SRn	Pending Service Request n. Indicates a pending Service Request for channel n. Pending SR status is a logic OR of all service requests pending: if only HSR is active, SRn clears only at the end of the thread. SRn clear due to the other request sources is microcode dependent. **NOTE:** The pending service status bit for a channel is 1 when a Service Request is pending, even if the Channel is disabled (CPRn = 0). **NOTE:** There can be a delay of one clock between writing HSR greater than 0 in eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C*n*HSRR_ENG1) of a channel and its respective bit being asserted in eTPU_GCR_CPSSR_ENG1. 0 No Service Request pending for channel n. 1 Pending Service Request for channel n.

2.11.16 eTPU Channel Pending Service Status Register for Engine 2 (eTPU_GCR_CPSSR_ENG2)

This register is a read-only register that holds the status of the pending Channel Service Requests (see Functions and threads).

Address: 0h base + 284h offset = 284h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SRn																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CPSSR_ENG2 field descriptions

Field	Description
0–31 SRn	Pending Service Request n. Indicates a pending Service Request for channel n. Pending SR status is a logic OR of all service requests pending: if only HSR is active, SRn clears only at the end of the thread. SRn clear due to the other request sources is microcode dependent. **NOTE:** The pending service status bit for a channel is 1 when a Service Request is pending, even if the Channel is disabled (CPRn = 0). **NOTE:** There can be a delay of one clock between writing HSR greater than 0 in eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C*n*HSRR_ENG2) of a channel and its respective bit being asserted in eTPU_GCR_CPSSR_ENG2. 0 No Service Request pending for channel n. 1 Pending Service Request for channel n.

2.11.17 eTPU Channel Service Status Register for Engine 1 (eTPU_GCR_CSSR_ENG1)

This register holds the current channel service status on whether it is being serviced or not (see Functions and threads). Only one bit may be asserted in this register at a given time. When no channel is being serviced, the register read value is 0x0000_0000. This register is read-only and can be read during normal eTPU operation for monitoring the scheduler activity.

NOTE

Channel Service Status does not always reflect decoding of the CHAN register (see CHAN Register), since the latter can be changed by the service thread microcode.

Address: 0h base + 290h offset = 290h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SSn																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CSSR_ENG1 field descriptions

Field	Description
0–31 SSn	Service Status n. Indicates that channel n is currently being serviced. It is updated at the 1st microcycle of a Time Slot Transition (see Time Slot Transition), or when the microengine ends the thread. 0 Channel n is not currently being serviced. 1 channel n is currently being serviced.

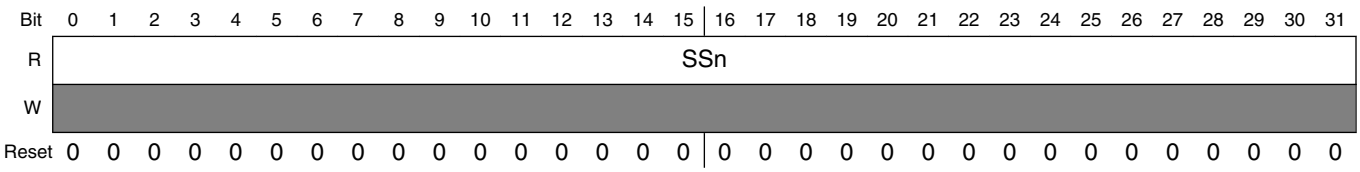
2.11.18 eTPU Channel Service Status Register for Engine 2 (eTPU_GCR_CSSR_ENG2)

This register holds the current channel service status on whether it is being serviced or not (see Functions and threads). Only one bit may be asserted in this register at a given time. When no channel is being serviced, the register read value is 0x0000_0000. This read-only register can be read during normal eTPU operation for monitoring the scheduler activity.

NOTE

Channel Service Status does not always reflect decoding of the CHAN register (see CHAN Register), since the latter can be changed by the service thread microcode.

Address: 0h base + 294h offset = 294h



eTPU_GCR_CSSR_ENG2 field descriptions

Field	Description
0–31 SSn	Service Status n. Indicates that channel n is currently being serviced. It is updated at the 1st microcycle of a Time Slot Transition (see Time Slot Transition), or when the microengine ends the thread. 0 Channel n is not currently being serviced. 1 channel n is currently being serviced.

2.12 Channel Configuration Control memory map and registers

Each channel has a group of 3 registers used to control, configure and check status of that channel as shown in the following table. This organization eases individual channel management.

Table 2-22. Channel registers structure

Channel offset	Register name
0x00	eTPU_CCCR_C*n*CR_ENG*m* - eTPU Channel Configuration Register
0x04	eTPU_CCCR_C*n*SCR_ENG*m* - eTPU Channel Status/Control Register
0x08	eTPU_CCCR_C*n*HSRR_ENG*m* - eTPU Channel Host Service Request Register
0x0C	Reserved

One contiguous area is used to map all channel registers of each eTPU engine as shown in the following table.

Table 2-23. Channel registers mMap

Offset	Registers structure
0x400	eTPU 1 Channel 0 Registers Structure
0x410	eTPU 1 Channel 1 Registers Structure
0x420	eTPU 1 Channel 2 Registers Structure
0x430	-----
0x5E0	eTPU 1 Channel 30 Registers Structure
0x5F0	eTPU 1 Channel 31 Registers Structure
0x600	RESERVED
0x800	eTPU 2 Channel 0 Registers Structure
0x810	eTPU 2 Channel 1 Registers Structure
0x820	-----
0x9E0	eTPU 2 Channel 30 Registers Structure
0x9F0	eTPU 2 Channel 31 Registers Structure
0xA00	RESERVED

There are 64 structures defined, one for each available channel in the eTPU System (32 for each Engine). The base address for the structure presented can be calculated by using the following equation:

Channel_Register_Base = eTPU_Engine_Channel_Base + (channel_number * 0x10), where:

- eTPU_Engine_Channel_Base = eTPU_Base + 0x400 for Engine 1.
- eTPU_Engine_Channel_Base = eTPU_Base + 0x800 for Engine 2.

eTPU_CCCR memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C0CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
404	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C0SCR_ENG1)	32	R/W	See section	2.12.2/178
408	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C0HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
410	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C1CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
414	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C1SCR_ENG1)	32	R/W	See section	2.12.2/178
418	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C1HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180

Table continues on the next page...

eTPU_CCCR memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
420	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C2CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
424	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C2SCR_ENG1)	32	R/W	See section	2.12.2/178
428	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C2HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
430	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C3CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
434	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C3SCR_ENG1)	32	R/W	See section	2.12.2/178
438	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C3HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
440	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C4CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
444	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C4SCR_ENG1)	32	R/W	See section	2.12.2/178
448	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C4HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
450	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C5CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
454	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C5SCR_ENG1)	32	R/W	See section	2.12.2/178
458	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C5HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
460	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C6CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
464	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C6SCR_ENG1)	32	R/W	See section	2.12.2/178
468	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C6HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
470	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C7CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
474	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C7SCR_ENG1)	32	R/W	See section	2.12.2/178
478	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C7HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
480	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C8CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
484	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C8SCR_ENG1)	32	R/W	See section	2.12.2/178
488	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C8HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
490	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C9CR_ENG1)	32	R/W	0000_0000h	2.12.1/175

Table continues on the next page...

eTPU_CCCR memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
494	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C9SCR_ENG1)	32	R/W	See section	2.12.2/178
498	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C9HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
4A0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C10CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
4A4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C10SCR_ENG1)	32	R/W	See section	2.12.2/178
4A8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C10HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
4B0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C11CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
4B4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C11SCR_ENG1)	32	R/W	See section	2.12.2/178
4B8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C11HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
4C0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C12CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
4C4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C12SCR_ENG1)	32	R/W	See section	2.12.2/178
4C8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C12HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
4D0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C13CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
4D4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C13SCR_ENG1)	32	R/W	See section	2.12.2/178
4D8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C13HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
4E0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C14CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
4E4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C14SCR_ENG1)	32	R/W	See section	2.12.2/178
4E8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C14HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
4F0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C15CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
4F4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C15SCR_ENG1)	32	R/W	See section	2.12.2/178
4F8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C15HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
500	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C16CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
504	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C16SCR_ENG1)	32	R/W	See section	2.12.2/178

Table continues on the next page...

eTPU_CCCR memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
508	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C16HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
510	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C17CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
514	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C17SCR_ENG1)	32	R/W	See section	2.12.2/178
518	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C17HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
520	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C18CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
524	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C18SCR_ENG1)	32	R/W	See section	2.12.2/178
528	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C18HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
530	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C19CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
534	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C19SCR_ENG1)	32	R/W	See section	2.12.2/178
538	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C19HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
540	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C20CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
544	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C20SCR_ENG1)	32	R/W	See section	2.12.2/178
548	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C20HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
550	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C21CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
554	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C21SCR_ENG1)	32	R/W	See section	2.12.2/178
558	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C21HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
560	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C22CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
564	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C22SCR_ENG1)	32	R/W	See section	2.12.2/178
568	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C22HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
570	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C23CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
574	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C23SCR_ENG1)	32	R/W	See section	2.12.2/178
578	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C23HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180

Table continues on the next page...

eTPU_CCCR memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
580	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C24CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
584	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C24SCR_ENG1)	32	R/W	See section	2.12.2/178
588	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C24HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
590	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C25CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
594	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C25SCR_ENG1)	32	R/W	See section	2.12.2/178
598	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C25HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
5A0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C26CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
5A4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C26SCR_ENG1)	32	R/W	See section	2.12.2/178
5A8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C26HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
5B0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C27CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
5B4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C27SCR_ENG1)	32	R/W	See section	2.12.2/178
5B8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C27HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
5C0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C28CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
5C4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C28SCR_ENG1)	32	R/W	See section	2.12.2/178
5C8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C28HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
5D0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C29CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
5D4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C29SCR_ENG1)	32	R/W	See section	2.12.2/178
5D8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C29HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
5E0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C30CR_ENG1)	32	R/W	0000_0000h	2.12.1/175
5E4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C30SCR_ENG1)	32	R/W	See section	2.12.2/178
5E8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C30HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
5F0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C31CR_ENG1)	32	R/W	0000_0000h	2.12.1/175

Table continues on the next page...

eTPU_CCCR memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5F4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C31SCR_ENG1)	32	R/W	See section	2.12.2/178
5F8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C31HSRR_ENG1)	32	R/W	0000_0000h	2.12.3/180
800	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C0CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
804	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C0SCR_ENG2)	32	R/W	See section	2.12.5/184
808	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C0HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
810	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C1CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
814	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C1SCR_ENG2)	32	R/W	See section	2.12.5/184
818	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C1HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
820	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C2CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
824	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C2SCR_ENG2)	32	R/W	See section	2.12.5/184
828	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C2HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
830	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C3CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
834	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C3SCR_ENG2)	32	R/W	See section	2.12.5/184
838	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C3HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
840	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C4CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
844	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C4SCR_ENG2)	32	R/W	See section	2.12.5/184
848	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C4HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
850	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C5CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
854	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C5SCR_ENG2)	32	R/W	See section	2.12.5/184
858	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C5HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
860	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C6CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
864	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C6SCR_ENG2)	32	R/W	See section	2.12.5/184

Table continues on the next page...

eTPU_CCCR memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
868	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C6HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
870	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C7CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
874	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C7SCR_ENG2)	32	R/W	See section	2.12.5/184
878	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C7HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
880	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C8CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
884	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C8SCR_ENG2)	32	R/W	See section	2.12.5/184
888	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C8HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
890	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C9CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
894	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C9SCR_ENG2)	32	R/W	See section	2.12.5/184
898	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C9HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
8A0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C10CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
8A4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C10SCR_ENG2)	32	R/W	See section	2.12.5/184
8A8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C10HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
8B0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C11CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
8B4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C11SCR_ENG2)	32	R/W	See section	2.12.5/184
8B8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C11HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
8C0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C12CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
8C4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C12SCR_ENG2)	32	R/W	See section	2.12.5/184
8C8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C12HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
8D0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C13CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
8D4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C13SCR_ENG2)	32	R/W	See section	2.12.5/184
8D8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C13HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186

Table continues on the next page...

eTPU_CCCR memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8E0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C14CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
8E4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C14SCR_ENG2)	32	R/W	See section	2.12.5/184
8E8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C14HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
8F0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C15CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
8F4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C15SCR_ENG2)	32	R/W	See section	2.12.5/184
8F8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C15HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
900	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C16CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
904	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C16SCR_ENG2)	32	R/W	See section	2.12.5/184
908	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C16HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
910	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C17CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
914	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C17SCR_ENG2)	32	R/W	See section	2.12.5/184
918	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C17HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
920	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C18CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
924	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C18SCR_ENG2)	32	R/W	See section	2.12.5/184
928	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C18HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
930	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C19CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
934	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C19SCR_ENG2)	32	R/W	See section	2.12.5/184
938	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C19HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
940	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C20CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
944	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C20SCR_ENG2)	32	R/W	See section	2.12.5/184
948	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C20HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
950	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C21CR_ENG2)	32	R/W	0000_0000h	2.12.4/181

Table continues on the next page...

eTPU_CCCR memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
954	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C21SCR_ENG2)	32	R/W	See section	2.12.5/184
958	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C21HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
960	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C22CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
964	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C22SCR_ENG2)	32	R/W	See section	2.12.5/184
968	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C22HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
970	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C23CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
974	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C23SCR_ENG2)	32	R/W	See section	2.12.5/184
978	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C23HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
980	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C24CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
984	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C24SCR_ENG2)	32	R/W	See section	2.12.5/184
988	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C24HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
990	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C25CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
994	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C25SCR_ENG2)	32	R/W	See section	2.12.5/184
998	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C25HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
9A0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C26CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
9A4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C26SCR_ENG2)	32	R/W	See section	2.12.5/184
9A8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C26HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
9B0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C27CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
9B4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C27SCR_ENG2)	32	R/W	See section	2.12.5/184
9B8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C27HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
9C0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C28CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
9C4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C28SCR_ENG2)	32	R/W	See section	2.12.5/184

Table continues on the next page...

eTPU_CCCR memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9C8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C28HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
9D0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C29CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
9D4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C29SCR_ENG2)	32	R/W	See section	2.12.5/184
9D8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C29HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
9E0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C30CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
9E4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C30SCR_ENG2)	32	R/W	See section	2.12.5/184
9E8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C30HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186
9F0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C31CR_ENG2)	32	R/W	0000_0000h	2.12.4/181
9F4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C31SCR_ENG2)	32	R/W	See section	2.12.5/184
9F8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C31HSRR_ENG2)	32	R/W	0000_0000h	2.12.6/186

2.12.1 eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1)

This register gathers configurations set individually per channel.

Address: 0h base + 400h offset + (16d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CIE	DTRE	CPR		T4DIS	SSMAE	ETPD	ETCS	0			CFS				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ODIS	OPOL	0			CPBA										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_CCCR_C*n*CR_ENG1 field descriptions

Field	Description
0 CIE	Channel Interrupt Enable (see Channel Interrupt and Data Transfer Requests). This bit is mirrored from eTPU Channel Interrupt Enable Register for Engine 1 (eTPU_GCR_CIER_ENG1). 0 Disable interrupt for this channel. 1 Enable interrupt for this channel.
1 DTRE	Channel Data Transfer Request Enable (see Interrupts and data transfer requests). This bit is mirrored from eTPU Channel Data Transfer Request Enable Register for Engine 1 (eTPU_GCR_CDTRER_ENG1). 0 Disable data transfer request for this channel. 1 Enable data transfer request for this channel.
2–3 CPR	Channel Priority. This field defines the priority level for the channel, used by the Hardware Scheduler (see Scheduler). 00 Disabled 01 Low 10 Middle 11 High
4 T4DIS	T4 channel action Disable. This bit forces the channel to work on T2 timing mode, regardless of the FCSS and CDFC fields in eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1), or of the TCR1CS field in eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1). For more details, see Channel timing modes. 0 Channel works either in T2 or T2/T4 timing modes, depending on the TCR1CS field in eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1) and the FCSS and CDFC fields in eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1). 1 Channel works in T2 timing mode, regardless of the TCR1CS field in eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1) or the FCSS and CDFC fields in eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1).
5 SSMAE	Shared Subsystem Microengine Access Enable. This bit enables the microengine access to the Shared Subsystem Area (see Shared Subsystems support). When this bit is negated, DIOB address values pointing to the Shared Subsystem Area actually access SDM mirrors, in legacy mode. **NOTE:** SSMAE value must only be changed when the engine is either stopped or halted. 0 An SDM mirror is accessed when the Shared Subsystem Area is addressed by the microengine. 1 Enables microengine access to the Shared Subsystem Area.
6 ETPD	Entry Table Pin Direction. This bit selects which channel signal, input or output, is used in the Entry Point selection. The ETPD value has to be compatible with the function chosen for the channel, selected in the field CFS. For details about Entry Table and condition encoding schemes, see Entry Points.

Table continues on the next page...

eTPU_CCCR_C*n*CR_ENG1 field descriptions (continued)

Field	Description
	0 Use PSTI for Entry Point selection. 1 Use PSTO for Entry Point selection.
7 ETCS	Entry Table Condition Select. This bit determines the channel condition encoding scheme that selects, according to channel conditions, the Entry Point to be taken in an Entry Table. ETCS value has to be compatible with the function chosen for the channel, selected in field CFS. Two condition encoding schemes are available. For details about Entry Table and condition encoding schemes, see Entry Points. 0 Select Standard Entry Table Condition encoding scheme. 1 Select Alternate Entry Table Condition encoding scheme.
8–10 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
11–15 CFS	Channel Function Select. This field defines the function to be performed by the channel (see Functions and threads). The Function assigned to the channel has to be compatible with the channel condition encoding scheme, selected by field ETCS.
16 ODIS	Output Disable. This bit enables the channel to have its output forced to the value opposite to OPOL when the output disable input signal corresponding to the channel group that it belongs is active (see eTPU channel output disable signals). 0 Turns off the output disable feature for the channel. 1 Turns on the output disable feature for the channel.
17 OPOL	Output Polarity. Determines the output signal polarity. The activation of the output disable signal forces, when enabled by the ODIS bit, the channel output signal to the opposite of this polarity. 0 Output active low (output disable drives output to high). 1 output active high (output disable drives output to low).
18–20 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
21–31 CPBA	Channel n Parameter Base Address. The value of this field times 8 specifies the SDM parameter base host (byte) address for channel n (2-parameter granularity; see SDM organization). As seen by the Host, the channel parameter base (byte) address is (without parameter sign extension) eTPU_Base + 0x8000 + CPBA * 8 or (with parameter sign extension), eTPU_Base + 0xC000 + CPBA*8.

2.12.2 eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C*n*SCR_ENG1)

This register gathers the interrupt status bits of the channel and the Function Mode definition (read-write). Bits CIS, CIOS and DTRS for each channel can be also accessed from eTPU Channel Interrupt Status Register for Engine 1 (eTPU_GCR_CISR_ENG1), eTPU Channel Interrupt Overflow Status Register for Engine 1 (eTPU_GCR_CIOSR_ENG1) and eTPU Channel Data Transfer Request Status Register for Engine 1 (eTPU_GCR_CDTRSR_ENG1). Host must write 1 to clear a status bit.

Address: 0h base + 404h offset + (16d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CIS	CIOS	0						DTRS	DTROS	0					
W	w1c	w1c							w1c	w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IPS	OPS	0	0											FM	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_CCCR_C*n*SCR_ENG1 field descriptions

Field	Description
0 CIS	Channel Interrupt Status. These bits are mirrored in eTPU Channel Interrupt Status Register for Engine 1 (eTPU_GCR_CISR_ENG1). Also, see Channel Interrupt and Data Transfer Requests. When read: 0.....Channel has no pending interrupt to the Host CPU. 1.....Channel has a pending interrupt to the Host CPU. When written: 0.....Keep interrupt status bit unaltered.

Table continues on the next page...

eTPU_CCCR_C*n*SCR_ENG1 field descriptions (continued)

Field	Description
	1.....Clear interrupt status bit.
1 CIOS	Channel Interrupt Overflow Status. These bits are mirrored in eTPU Channel Interrupt Overflow Status Register for Engine 1 (eTPU_GCR_CIOSR_ENG1). Also, see Interrupt and Data Transfer Request overflow. When read: 0.....Interrupt overflow negated for this channel. 1.....Interrupt overflow asserted for this channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.
2–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8 DTRS	Data Transfer Request Status. These bits are mirrored in eTPU Channel Data Transfer Request Status Register for Engine 1 (eTPU_GCR_CDTRSR_ENG1). Also, see Channel Interrupt and Data Transfer Requests. When read: 0.....Channel has no pending data transfer request. 1.....Channel has a pending data transfer request. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.
9 DTROS	Data Transfer Request Overflow Status. These bits are mirrored in eTPU Channel Data Transfer Request Overflow Status Register for Engine 1 (eTPU_GCR_CDTROSR_ENG1). Also, see Interrupt and Data Transfer Request overflow. When read: 0.....Data transfer request overflow negated for this channel. 1.....Data transfer request overflow asserted for this channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.
10–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16 IPS	Channel Input Pin State. This bit shows the current value of the filtered channel input signal state.
17 OPS	Channel Output Pin State.

Table continues on the next page...

eTPU_CCCR_C*n*SCR_ENG1 field descriptions (continued)

Field	Description
	This bit shows the current value driven in the channel output signal, including the effect of the external output disable feature (see eTPU channel output disable signals). If the channel input and output signals are connected to the same pad, OPS reflects the value driven to the pad (if OBE=1). This is not necessarily the actual pad value, which drives the value in the bit IPS.
18 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
19–29 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
30–31 FM	Channel Function Mode. Each function uses this field for a specific configuration. These bits can be tested by microengine code (see Conditional/unconditional branch). These bits are equivalent to the TPU/TPU2/TPU3 Host Sequence (HSQ) bits.

2.12.3 eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C*n*HSRR_ENG1)

This register is used by the Host to issue service requests to the channel.

Address: 0h base + 408h offset + (16d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																													HSR		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_CCCR_C*n*HSRR_ENG1 field descriptions

Field	Description
0–28 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
29–31 HSR	Host Service Request This field is used by the Host CPU to request service to the channel (see Host Service Requests). HSR value turns to 000 automatically at the end of microengine service for that channel, but only if the thread started due to an HSR. Host should write HSR greater than 0 only when HSR=0. Writing HSR=000 withdraws a pending request if scheduler did not begin to resolve the Entry Point yet, but it does not abort the service thread from that point on. For more details, see Entry Points and Host Service Requests. 000 No Host Service Request pending.

Table continues on the next page...

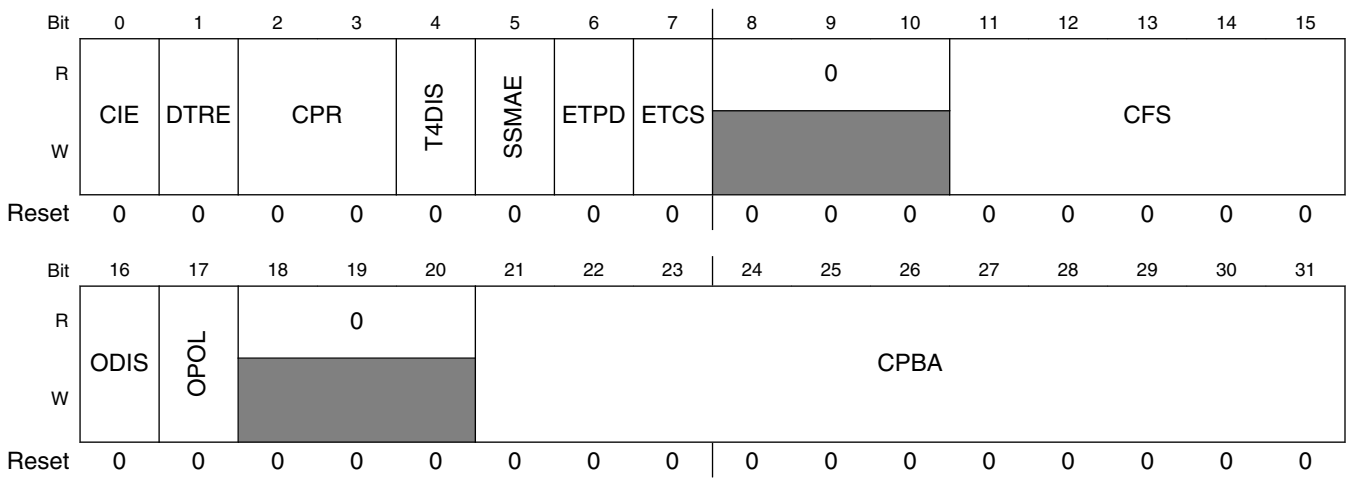
eTPU_CCCR_C*n*HSRR_ENG1 field descriptions (continued)

Field	Description
	001 Function-dependent Host Service Request pending. --- --- 111 Function-dependent Host Service Request pending.

2.12.4 eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2)

This register gathers configurations set individually per channel.

Address: 0h base + 800h offset + (16d × i), where i=0d to 31d



eTPU_CCCR_C*n*CR_ENG2 field descriptions

Field	Description
0 CIE	Channel Interrupt Enable (see Channel Interrupt and Data Transfer Requests). This bit is mirrored from eTPU Channel Interrupt Enable Register for Engine 2 (eTPU_GCR_CIER_ENG2). 0 Disable interrupt for this channel. 1 Enable interrupt for this channel.
1 DTRE	Channel Data Transfer Request Enable (see Channel Interrupt and Data Transfer Requests). This bit is mirrored from eTPU Channel Data Transfer Request Enable Register for Engine 2 (eTPU_GCR_CDTRER_ENG2). 0 Disable data transfer request for this channel. 1 Enable data transfer request for this channel.

Table continues on the next page...

eTPU_CCCR_C*n*CR_ENG2 field descriptions (continued)

Field	Description
2–3 CPR	Channel Priority. This field defines the priority level for the channel, used by the Hardware Scheduler (see Scheduler). 00 Disabled 01 Low 10 Middle 11 High
4 T4DIS	T4 channel action Disable. This bit forces the channel to work on T2 timing mode, regardless of the FCSS and CDFC fields in eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2), or of the TCR1CS field in eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2). For more details, see Channel timing modes. 0 Channel works either in T2 or T2/T4 timing modes, depending on the TCR1CS field in eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2) and the FCSS and CDFC fields in eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2). 1 Channel works in T2 timing mode, regardless of the TCR1CS field in eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2) or the FCSS and CDFC fields in eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2).
5 SSMAE	Shared Subsystem Microengine Access Enable. This bit enables the microengine access to the Shared Subsystem Area (see Shared Subsystems support). When this bit is negated, DIOB address values pointing to the Shared Subsystem Area actually access SDM mirrors, in legacy mode. **NOTE:** SSMAE value must only be changed when the engine is either stopped or halted. 0 An SDM mirror is accessed when the Shared Subsystem Area is addressed by the microengine. 1 Enables microengine access to the Shared Subsystem Area.
6 ETPD	Entry Table Pin Direction. This bit selects which channel signal, input or output, is used in the Entry Point selection. The ETPD value has to be compatible with the function chosen for the channel, selected in the field CFS. For details about Entry Table and condition encoding schemes, see Entry Points. 0 Use PSTI for Entry Point selection. 1 Use PSTO for Entry Point selection.
7 ETCS	Entry Table Condition Select. This bit determines the channel condition encoding scheme that selects, according to channel conditions, the Entry Point to be taken in an Entry Table. ETCS value has to be compatible with the function chosen for the channel, selected in field CFS. Two condition encoding schemes are available. For details about Entry Table and condition encoding schemes, see Entry Points. 0 Select Standard Entry Table Condition encoding scheme. 1 Select Alternate Entry Table Condition encoding scheme.
8–10 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

eTPU_CCCR_C*n*CR_ENG2 field descriptions (continued)

Field	Description
11–15 CFS	Channel Function Select. This field defines the function to be performed by the channel (see Functions and threads). The Function assigned to the channel has to be compatible with the channel condition encoding scheme, selected by field ETCS.
16 ODIS	Output Disable. This bit enables the channel to have its output forced to the value opposite to OPOL when the output disable input signal corresponding to the channel group that it belongs is active (see eTPU channel output disable signals). 0 Turns off the output disable feature for the channel. 1 Turns on the output disable feature for the channel.
17 OPOL	Output Polarity. Determines the output signal polarity. The activation of the output disable signal forces, when enabled by the ODIS bit, the channel output signal to the opposite of this polarity. 0 Output active low (output disable drives output to high). 1 output active high (output disable drives output to low).
18–20 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
21–31 CPBA	Channel n Parameter Base Address. The value of this field times 8 specifies the SDM parameter base host (byte) address for channel n (2-parameter granularity; see SDM organization). As seen by the Host, the channel parameter base (byte) address is (without parameter sign extension) eTPU_Base + 0x8000 + CPBA * 8 or (with parameter sign extension), eTPU_Base + 0xC000 + CPBA*8.

2.12.5 eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C*n*SCR_ENG2)

This register gathers the interrupt status bits of the channel and the Function Mode definition (read-write). Bits CIS, CIOS and DTRS for each channel can be also accessed from eTPU Channel Interrupt Status Register for Engine 2 (eTPU_GCR_CISR_ENG2), eTPU Channel Interrupt Overflow Status Register for Engine 2 (eTPU_GCR_CIOSR_ENG2) and eTPU Channel Data Transfer Request Status Register for Engine 2 (eTPU_GCR_CDTRSR_ENG2). Host must write 1 to clear a status bit.

Address: 0h base + 804h offset + (16d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CIS	CIOS	0						DTRS	DTROS	0					
W	w1c	w1c							w1c	w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IPS	OPS	0	0											FM	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_CCCR_C*n*SCR_ENG2 field descriptions

Field	Description
0 CIS	Channel Interrupt Status. These bits are mirrored in eTPU Channel Interrupt Status Register for Engine 2 (eTPU_GCR_CISR_ENG2). Also, see Channel Interrupt and Data Transfer Requests. When read: 0.....Channel has no pending interrupt to the Host CPU. 1.....Channel has a pending interrupt to the Host CPU. When written: 0.....Keep interrupt status bit unaltered.

Table continues on the next page...

eTPU_CCCR_C*n*SCR_ENG2 field descriptions (continued)

Field	Description
	1.....Clear interrupt status bit.
1 CIOS	Channel Interrupt Overflow Status. These bits are mirrored in eTPU Channel Interrupt Overflow Status Register for Engine 2 (eTPU_GCR_CIOSR_ENG2). Also, see Interrupt and Data Transfer Request overflow. When read: 0.....Interrupt overflow negated for this channel. 1.....Interrupt overflow asserted for this channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.
2–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8 DTRS	Data Transfer Request Status. These bits are mirrored in eTPU Channel Interrupt Status Register for Engine 2 (eTPU_GCR_CISR_ENG2). Also, see Channel Interrupt and Data Transfer Requests. When read: 0.....Channel has no pending data transfer request. 1.....Channel has a pending data transfer request. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.
9 DTROS	Data Transfer Request Overflow Status. These bits are mirrored in eTPU Channel Data Transfer Request Overflow Status Register for Engine 2 (eTPU_GCR_CDTROSR_ENG2). Also, see Channel Interrupt and Data Transfer Requests. When read: 0.....Data transfer request overflow negated for this channel. 1.....Data transfer request overflow asserted for this channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.
10–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16 IPS	Channel Input Pin State. This bit shows the current value of the filtered channel input signal state.
17 OPS	Channel Output Pin State.

Table continues on the next page...

eTPU_CCCR_C*n*SCR_ENG2 field descriptions (continued)

Field	Description
	This bit shows the current value driven in the channel output signal, including the effect of the external output disable feature (see eTPU channel output disable signals). If the channel input and output signals are connected to the same pad, OPS reflects the value driven to the pad (if OBE=1). This is not necessarily the actual pad value, which drives the value in the bit IPS.
18 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
19–29 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
30–31 FM	Channel Function Mode. Each function uses this field for a specific configuration. These bits can be tested by microengine code (see Conditional/unconditional branch). These bits are equivalent to the TPU/TPU2/TPU3 Host Sequence (HSQ) bits.

2.12.6 eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C*n*HSRR_ENG2)

This register is used by the Host to issue service requests to the channel.

Address: 0h base + 808h offset + (16d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																													HSR		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_CCCR_C*n*HSRR_ENG2 field descriptions

Field	Description
0–28 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
29–31 HSR	Host Service Request This field is used by the Host CPU to request service to the channel (see Host Service Requests). HSR value turns to 000 automatically at the end of microengine service for that channel, but only if the thread started due to an HSR. Host should write HSR greater than 0 only when HSR=0. Writing HSR=000 withdraws a pending request if scheduler did not begin to resolve the Entry Point yet, but it does not abort the service thread from that point on. For more details, see Entry Points and Host Service Requests. 000 No Host Service Request pending.

Table continues on the next page...

eTPU_CCCR_C*n*HSRR_ENG2 field descriptions (continued)

Field	Description
	001 Function-dependent Host Service Request pending. --- --- 111 Function-dependent Host Service Request pending.

2.13 Shared Subsystem Integration memory map and registers

Shared Subsystems Integration Registers configure aspects of how the shared modules interface with the eTPU. There is one integration register for each subsystem unit, which can be, depending on the specific subsystem, a whole IP module, a sub-module, or a channel within a module. For more details, see Shared Subsystems support.

eTPU_SSIR memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C00	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR0)	32	R/W	0000_0000h	2.13.1/188
C04	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR1)	32	R/W	0000_0000h	2.13.1/188
C08	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR2)	32	R/W	0000_0000h	2.13.1/188
C0C	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR3)	32	R/W	0000_0000h	2.13.1/188
C10	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR4)	32	R/W	0000_0000h	2.13.1/188
C14	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR5)	32	R/W	0000_0000h	2.13.1/188
C18	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR6)	32	R/W	0000_0000h	2.13.1/188
C1C	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR7)	32	R/W	0000_0000h	2.13.1/188
C20	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR8)	32	R/W	0000_0000h	2.13.1/188
C24	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR9)	32	R/W	0000_0000h	2.13.1/188

2.13.1 eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR*n*)

This register is used to configure the interface of a subsystem unit with the eTPU.

Address: 0h base + C00h offset + (4d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								LENG		0	LCHAN				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_SSIR_ETPUSSSIR*n* field descriptions

Field	Description
0–23 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
24–25 LENG	Link Engine. LENG specifies the engine that receives a Link Service Request when an interrupt is issued by the Shared Unit, or disables Shared Unit linking. The channel that receives the Link Service Request is specified in the LCHAN field. 00 Shared Unit Link Service Request disabled 01 Engine 1 receives Shared Unit Link Service Request 10 Engine 2 receives Shared Unit Link Service Request 11 Reserved
26 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–31 LCHAN	Link Channel. LCHAN specifies the channel linked when an interrupt is issued by the Shared Unit, if enabled in the LENG field. The field is meaningless if LENG disables Shared Unit linking. For more information on Link Service Requests, see Channel Link.

2.14 Functional description

2.14.1 Functions and threads

eTPU processing is event-driven, in the sense that eTPU microcode only runs to service a request from an event. Service Requests may result from the occurrence of any of the following events:

- Host CPU writing a non-zero value to the channel HSR (Host Service Request) field in eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C*n*HSRR_ENG1) and eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C*n*HSRR_ENG2).
- Occurrence of a time base match, an input signal transition, or a specific combination of them (depending on the Channel Mode currently configured).
- A Link Service Request.

A given event is always associated to only one Channel:

- There is one HSR register field for each Channel
- Each signal is associated with only one Channel, which has its own Match registers and independent mode configuration.
- Each Link Service Request can have only one Channel as a target.

Service Request processing is done by a set of microengine routines. A set of related routines that implement a specific channel application is called a **Function**. One or more Functions reside on SCM, limited only by the SCM space available, size of microcode Functions and the number of entry points available. Each Engine can be controlled by up to 32 Functions at a time.

A Function can be assigned to several channels, but only one Function can be assigned to a given Channel at a time. This is defined by the Host through the Channel Configuration Registers (see Channel Configuration Control memory map and registers).

The term **Thread** will be used hereafter to refer to a service routine of a Function, or its execution. A Thread is constructed of a specific number of microinstructions, typically the code necessary to calculate the next phase of waveform to be input to, or output from, a given channel. Once a Thread begins, its execution cannot be interrupted. A Thread normally finishes when an END microinstruction is executed.

A given Thread is selected and called by the Scheduler depending on the following:

- the type of event that generated the service request.

- the Function assigned to the target channel.
- target channel pin state.
- the state of the channel logic.
- the priority assigned to the target channel, relative to the priorities of other channels with pending service requests

The mechanism to select a thread based on the channel Function and type of event is described in the Entry Points.

The priority mechanism that determines the order of Thread execution amongst pending service requests is described in Scheduler.

2.14.1.1 Entry Points

2.14.1.1.1 Entry Table

Each Thread has its **Entry Point**, which contains the SCM address of its first instruction, besides other information. For a complete Entry Point description, see Entry Point format.

Once the Scheduler chooses a channel among pending Service Requests, the Entry Point is taken from an **Entry Table**, based on the Function assigned for the channel and other conditions. Entry Table layout is shown in the following figure.

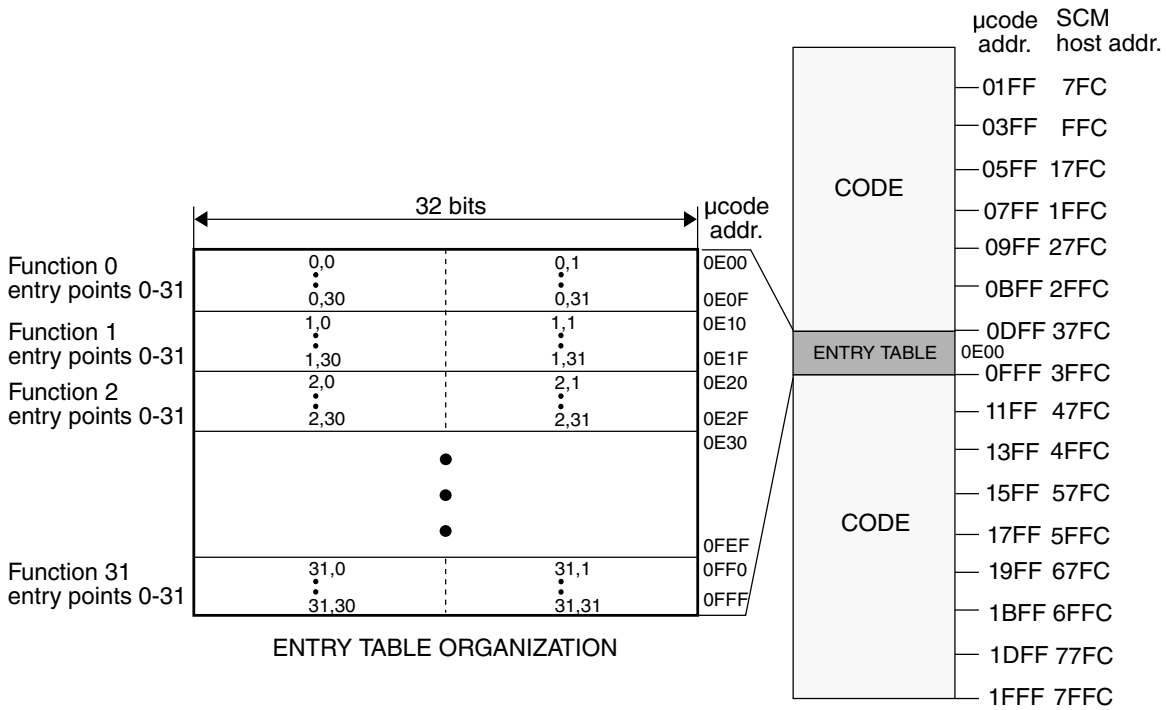


Figure 2-3. Entry Table

The Entry Table is organized by Functions. Each Function can have up to 32 Entry Points of 16 bits each, corresponding to 32 possible Threads per Function. Each Entry Point location in the table corresponds to a combination of events and channel states (see Entry Point address generation). A single Thread can be associated to more than one combination, having its Entry Point repeated in the table. Each 32-bit word in the Entry Table holds two Entry Points.

Note that the Entry Table can be placed in any SCM address multiple of the Entry Table size, determined by the ETB field in eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2). However, it is recommended to place the Entry Table at the start of the SCM to get continuous code memory and to ease the eventual migration of the code from larger parts down to smaller ones without rearranging the binary image, but this is not a restriction. Unused Entry Points may be used for microcode, so this organization extends the microcode continuous area to the unused area of the Entry Table. For this purpose, Function numbers should be selected from 0 up to 31. If, for example, only 8 functions are implemented, only the Entry Table locations for Functions 0 to 7 are used, and the Entry Table locations for functions 8 to 31 can be used as microinstruction memory (adding extra continuous 1536 bytes for microprogram usage).

One way of implementing different sets of Functions is having more than one Entry Table, and configuring the eTPU with the appropriate one for the application by changing the ETB field in eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2). Note that the Engines can use different Entry Tables, with or without the same set of Functions.

2.14.1.1.2 Entry Point address generation

The Entry Point address within the Entry Table is determined by the Function assigned to the Channel, the state of the Channel, the type of event, and the condition encoding scheme. Together with the Entry Table base address, they form the Entry Point Address at the SCM, as shown in the following figure.

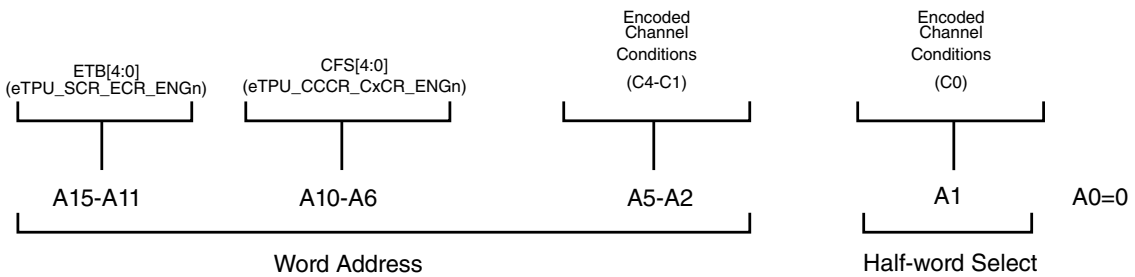


Figure 2-4. Entry Point Address (host address offset)

The type of event and channel state are coded in the Encoded Channel Conditions field C4-C0, according to one of two encoding schemes:

- Standard Entry Table Condition encoding scheme, shown in Table 2-24, which privileges Host Service Requests.
- Alternate Entry Table Condition encoding scheme, shown in Table 2-25, which focus on other events and state decoding.

The events that take part on condition encoding generate a Service Request, and have four origins:

1. Match Recognition (caused by greater/equal match, or equal-only, between the value TCR1/2 and the value stored in the channel match registers). eTPU channels support single and double match in various modes of match recognition; see Match Recognition.
2. Transition Detect Service Request (channel input signal transition detection of a selected edge). The eTPU channels support single and double transition, which together with the double match options provide various modes of transition detection; see Transition detection and Time Base Capture.

3. Channel Linking Service Request (microcode writing the channel number to the LINK register). Link service request allows one channel to activate another (see Channel Link). Links may also come from Shared Subsystem units (see Shared Subsystems support).
4. Host Service Request (Host writes a non-zero value to the HSR bits of the channel; see Host Service Requests).

Note

Even if a Transition or Match Service Request is inhibited (by channel mode/state or SRI), the Transition Detection and Match Recognition are taken into account for condition encoding. That is, the MRLA/B and TDLA/B flags are used, not their respective Service Requests.

Columns Host Request Bits, Link Request, MatchA/TransB, and MatchB/TransA determine the type of event. A non-zero value in these columns represents the recognition of the event, while "x" indicates that its recognition is irrelevant. Values 1 and 0 mean that event was recognized or not, respectively. Note that Match and Transition events may occur and not be recognized, and in this case it assumes value 0 for the condition encoding. The recognition of such an occurred event depends on the channel mode assigned and other conditions, as described in Enhanced Channels.

The **Host Service Request Bits** column refers to the value written by the Host CPU to the Host Service Request Registers (eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C*n*HSRR_ENG1) and eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C*n*HSRR_ENG2)) of the Channel being serviced. Note that the bits on this row are coded (3-bit representation). If the value of HSR is not zero, then the Host actually requested service.

The **Link Request** column refers to the occurrence of a Channel Link or SSS Link requests.

The **MatchA/TransB** column refers to the recognition of either a Match event specified by MatchA channel register or the detection of a channel input signal event specified by the IPACB configuration register (see Pin Control Registers).

The **MatchB/TransA** column refers to the recognition of either a Match event specified by MatchB channel register or the detection of a channel input signal event specified by the IPACA configuration register (see Pin Control Registers).

For the channel input signal, MatchA and MatchB provide double timeout conditions which depend on the channel mode programming (see Channel Modes). If the channel is used for output only, there are no transition detections, so the MatchB/TransA column

represents only Match B, and MatchA/TransB column the Match A. In this case Match A and Match B are separated to give better state resolution in double match output functions. For more information about channel requests refer to Enhanced Channels.

Besides those events, the following channel state conditions help to determine the Entry Point:

1. Channel Flags 0 and 1: these are channel-internal flags (not in SDM) associated with a channel. Their values are set by microcode (see Channel flags operations).
2. Input Pin state or Output Flip Flop: the state (0 or 1) of the channel input signal after the Enhanced Filter (see Enhanced Digital Filter - EDF), or the state driven to the output signal. Which one (input or output) is used is selected by the ETPD field in eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2).

The two Entry Table Condition encoding schemes combine events and state conditions differently, as detailed in the following sections.

2.14.1.1.3 Standard condition encoding scheme

In this scheme, shown in the following table, all 7 HSR combinations are used and other event type columns are marked "x" when HSR is non-zero, indicating that Host Service Request has priority over any other type of event. However, when an HSR service thread is called (entry numbers 0 to 9), other events may also have been recognized, and it is microcode responsibility to check them.

When HSR is 0, i.e., Host did not issue a Service Request to the channel, the other event conditions, the input signal state and channel flags determine the Entry Point. Note that channel flag 1 does not influence the encoding in this scheme.

Table 2-24. Standard channel condition encoding scheme

No.	Encoded channel conditions [C4-C0]	Host Service Request bits	Link request	MatchA/ TransB	MatchB/ TransA	In/output pin state[1]	Channel Flag1	Channel Flag0
0	001	00000	x	x	x	0	x	0
1	00001	001	x	x	x	0	x	1
2	00010	001	x	x	x	1	x	0
3	00011	001	x	x	x	1	x	1
4	00100	010	x	x	x	x	x	x
5	00101	011	x	x	x	x	x	x
6	00110	100	x	x	x	x	x	x

Table continues on the next page...

Table 2-24. Standard channel condition encoding scheme (continued)

No.	Encoded channel conditions [C4-C0]	Host Service Request bits	Link request	MatchA/ TransB	MatchB/ TransA	In/output pin state[1]	Channel Flag1	Channel Flag0
7	**00111**	**101**	**x**	**x**	**x**	**x**	**x**	**x**
8	**01000**	**110**	**x**	**x**	**x**	**x**	**x**	**x**
9	**01001**	**111**	**x**	**x**	**x**	**x**	**x**	**x**
10	**01010**	000	**1**	1	1	**x**	**x**	0
11	**01011**	000	**1**	1	1	**x**	**x**	1
12	**01100**	**000**	**0**	**0**	1	**0**	**x**	**0**
13	**01101**	000	**0**	**0**	1	0	**x**	1
14	01110	000	**0**	**0**	1	1	**x**	0
15	01111	000	**0**	**0**	1	1	**x**	1
16	**10000**	000	**0**	1	0	**0**	**x**	**0**
17	**10001**	000	**0**	1	0	0	**x**	1
18	**10010**	000	**0**	1	0	1	**x**	0
19	**10011**	000	**0**	1	0	1	**x**	1
20	**10100**	**000**	**0**	1	1	**0**	**x**	**0**
21	**10101**	000	**0**	1	1	0	**x**	1
22	**10110**	000	**0**	1	1	1	**x**	0
23	**10111**	000	**0**	1	1	1	**x**	1
24	**11000**	000	1	0	0	**0**	**x**	**0**
25	**11001**	000	1	0	0	0	**x**	1
26	**11010**	000	1	0	0	1	**x**	0
27	**11011**	000	1	0	0	1	**x**	1
28	**11100**	000	1	0	1	**x**	**x**	**0**
29	**11101**	000	1	0	1	**x**	**x**	1
30	11110	000	1	1	**0**	**x**	**x**	0
31	11111	000	1	1	0	**x**	**x**	1

Host Service Request

1. The ETPD field in CCR_ENG1 and CCR_ENG2 selects between input and output pin state.

2.14.1.1.4 Alternate condition encoding scheme

This scheme is shown in the following table. Because the HSR bits cannot be tested by microcode, only three distinct Host Service Request can be used:

1. HSR=010 or 011, which are coded into the same Entry Points (0 to 3)
2. HSR=100, 101 or 001, which are all coded into Entry Point 4

3. HSR=110 or 111, which are both coded into Entry Point 5

The remaining Entry Points use both channel flags for better state decoding, making this scheme better suited for Functions which need more states and/or faster state decoding, without needing many HSRs.

Table 2-25. Alternate channel condition encoding scheme

No.	Encoded channel conditions [C4-C0]	Host Service Request bits	Link request	MatchA/ TransB	MatchB/ TransA	In/Output pin state[1]	Channel Flag1	Channel Flag0
0	**00000**	**01x**	**x**	**x**	**x**	**0**	**x**	**0**
1	**00001**	**01x**	**x**	**x**	**x**	**0**	**x**	**1**
2	**00010**	**01x**	**x**	**x**	**x**	**1**	**x**	**0**
3	**00011**	**01x**	**x**	**x**	**x**	**1**	**x**	**1**
4	**00100**	**10x/001**	**x**	**x**	**x**	**x**	**x**	**x**
5	**00101**	**11x**	**x**	**x**	**x**	**x**	**x**	**x**
6	**00110**	000	**1**	0	0	0	x	x
7	**00111**	000	**1**	0	0	1	x	x
8	**01000**	**000**	**x**	1	0	0	**0**	**0**
9	**01001**	000	**x**	1	0	0	0	1
10	**01010**	000	**x**	1	0	0	1	0
11	**01011**	000	**x**	1	0	0	1	1
12	**01100**	**000**	**x**	1	0	1	**0**	**0**
13	**01101**	000	**x**	1	0	1	0	1
14	01110	000	**x**	1	0	1	1	0
15	01111	000	**x**	1	0	1	1	1
16	**10000**	000	**x**	0	1	0	0	0
17	**10001**	000	**x**	0	1	0	0	1
18	**10010**	000	**x**	0	1	0	1	0
19	**10011**	000	**x**	0	1	0	1	1
20	**10100**	**000**	**x**	0	1	1	**0**	**0**
21	**10101**	000	**x**	0	1	1	0	1
22	**10110**	000	**x**	0	1	1	1	0
23	**10111**	000	**x**	0	1	1	1	1
24	**11000**	000	**x**	1	1	0	**0**	**0**
25	**11001**	000	**x**	1	1	0	0	1
26	**11010**	000	**x**	1	1	0	1	0
27	**11011**	000	**x**	1	1	0	1	1
28	**11100**	000	**x**	1	1	1	**0**	**0**
29	**11101**	000	**x**	1	1	1	0	1
30	11110	000	**x**	1	1	1	1	0

Table continues on the next page...

Table 2-25. Alternate channel condition encoding scheme (continued)

No.	Encoded channel conditions [C4-C0]	Host Service Request bits	Link request	MatchA/ TransB	MatchB/ TransA	In/Output pin state[1]	Channel Flag1	Channel Flag0
31	11111	000	**x**	1	1	1	1	1
		Host Service Request						

1. The ETPD field in CCR_ENG1 and CCR_ENG2 selects between input and output pin state.

2.14.1.1.5 Entry Point format

Entry Point information includes a Preload-Parameter selection field, a Match Enable field, and the first microcode address of the thread. The Entry Point format is illustrated in the following figure.

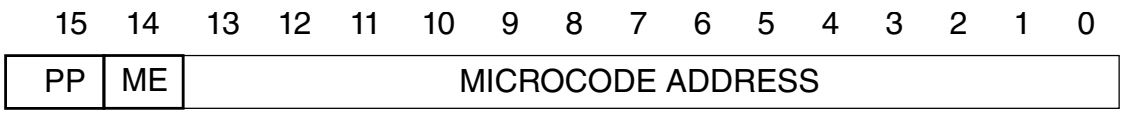


Figure 2-5. Entry Point format

PP - Preload Parameter

PP indicates which pair of channel parameters are loaded into registers P and DIOB from the SDM prior to the execution of a thread. Preloading occurs during the time-slot transition period (see Time Slot Transition).

1 = microengine register P is preloaded from parameter 2 and DIOB from parameter 3.

0 = microengine register P is preloaded from parameter 0 and DIOB from parameter 1.

The parameter numbers are offsets from the channel parameter base address. For more info, see Parameter Access.

ME - Match Enable

ME specifies whether match event recognitions are enabled or disabled for the thread associated with the entry point during the thread execution. If they are disabled, a match recognition can only occur after channel service. For more details refer to Match Recognition.

1 = Matches are enabled during the thread.

0 = Matches are disabled during the thread.

The disabling of Match A/B recognition by MEF is dependent on IPACA/B configuration on the serviced channel (see Pin Control Registers). If IPACA=1xx, Match A is not disabled by ME=0. Likewise, IPACB=1xx overrides the effect of ME on Match B to "always on" If IPACA/B=0xx, Match A/B is disabled for two microcycles during TST (see Time Slot Transition) and is re-enabled when Entry Point is loaded, if ME=1. Note that if the comparator is in equal-only mode and the time base reaches the value of the Match register during the time that recognition is disabled (beginning of TST, plus whole thread if ME=0), the match recognition is lost. If the comparator is in greater-equal mode, the match event may be recognized after the disabling period if it satisfies the "greater-than" condition.

MICROCODE ADDRESS - Starting Microcode Address

This field specifies the microcode address on which the thread is to begin execution.

2.14.1.2 Time Slot Transition

The Time Slot Transition period (also called TST for short) is the interval between the servicing of two channels, during which all Channel-specific context is loaded for the new serviced Channel. The primary tasks completed during this period include:

- Set MEF for the TST wait-states, if any.
- Reset MEF for the first two microcycles (after eventual wait-states).
- Update of the CHAN register with the number of the new channel to be serviced.
- Parallel update of ERTA and ERTB from CaptureA and CaptureB registers of the new serviced channel.
- Sampling of the branch conditions of the new channel to be serviced into the branch logic (this means flags TDLA/B, MRLA/B, LSR, FM[1], FM[0], and PSS). The branch conditions are coherent with the timebase capture values sampled into ERTA/B (if MRLA/B, TDLA/B are set at the same time of the sampling, either both old flag state and capture values are sampled, or both new values are sampled).
- Formation of the entry point address.
- Copy the ME bit in the Entry Point into MEF.
- Access to the entry point location and getting the first microinstruction address.
- Preload of two parameters from the SDM into P (32 bits) and DIOB (24 bits).
- Fetch the first instruction of the thread to be executed for the new channel.
- Preset the RAR register value (see RAR Register).

The preload operation is 32-bit wide for P and 24-bit wide for DIOB. The P register is loaded with all the 32-bit parameter. The DIOB register is loaded with the lower 24-bits of the parameter. The microcode can switch at any time to access the lower 24-bits, upper byte, or all the 32-bits of any parameter in the SDM. Preload of P-DIOB pair of parameters is atomic with respect to Host and CDC accesses, and so are coherent with their dual-parameter coherent transfers. For more details see Parameter sharing and coherency.

No instructions are executed at the Engine where the time slot transition period occurs, but the other Engine can execute normally. Match A/B is unconditionally disabled on the first two TST microcycles, if IPACA/B=0xx (respectively). During the rest of time slot transition, match recognition can be disabled or not, depending on IPACA/B field and ME. See Match Recognition .

Time Slot Transition takes a minimum of 3 microcycles (6 eTPU clocks), which may be extended due to SDM arbitration wait-states for the first preload access (see SDM arbitration). When no wait-states are received (Figure 2-6), DIOB is preloaded twice, one for each PP value, and the correct value remains in DIOB when the Entry Point is loaded. Figure 2-7 and Figure 2-8 show the timing for one and two wait-states, respectively.

Registers B, C, D and SR are not altered by TST and keep their values from the previous thread. The values of registers A, MACL and MACH are not guaranteed at the thread start.

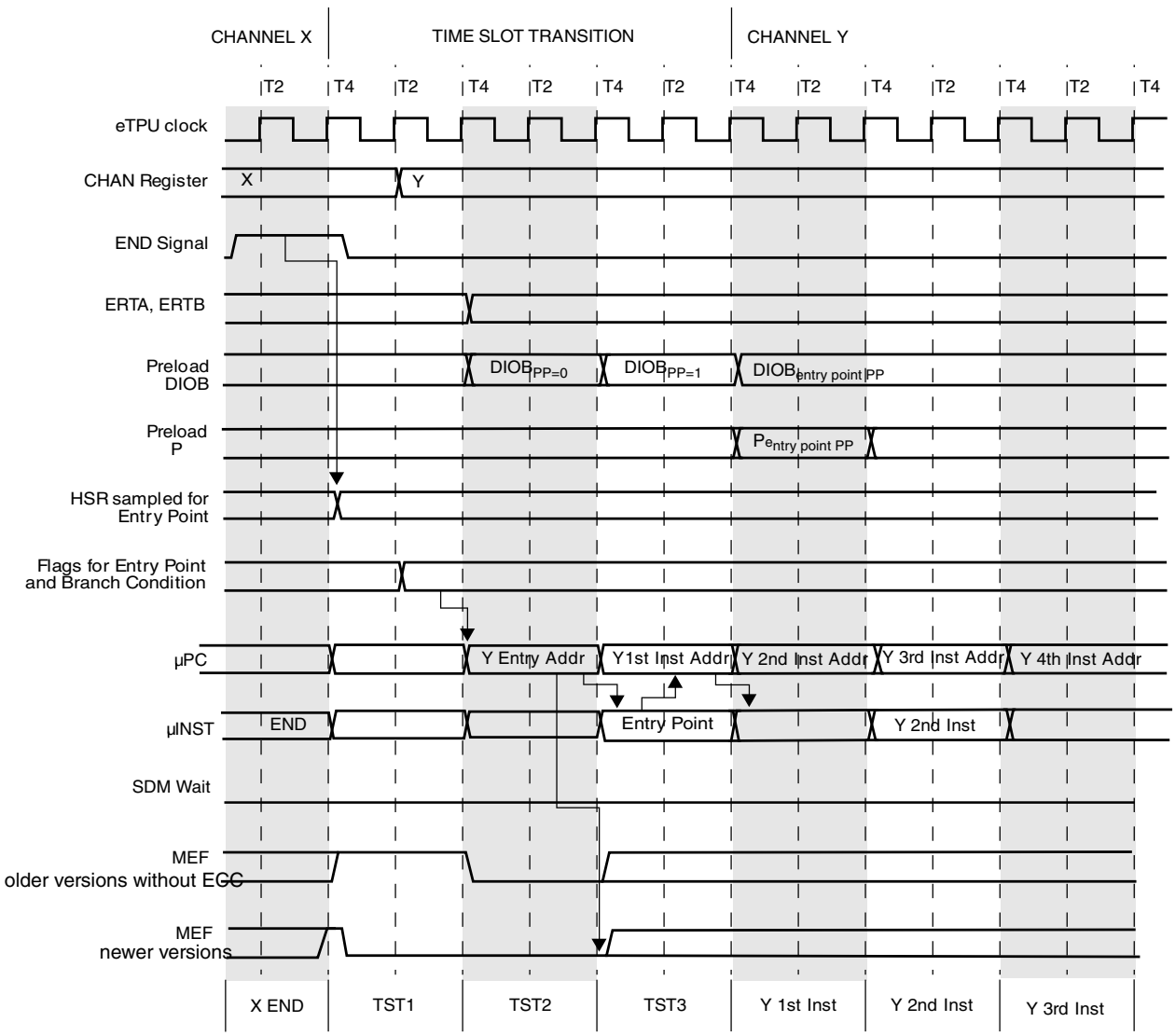


Figure 2-6. TST Timing - No Wait-states

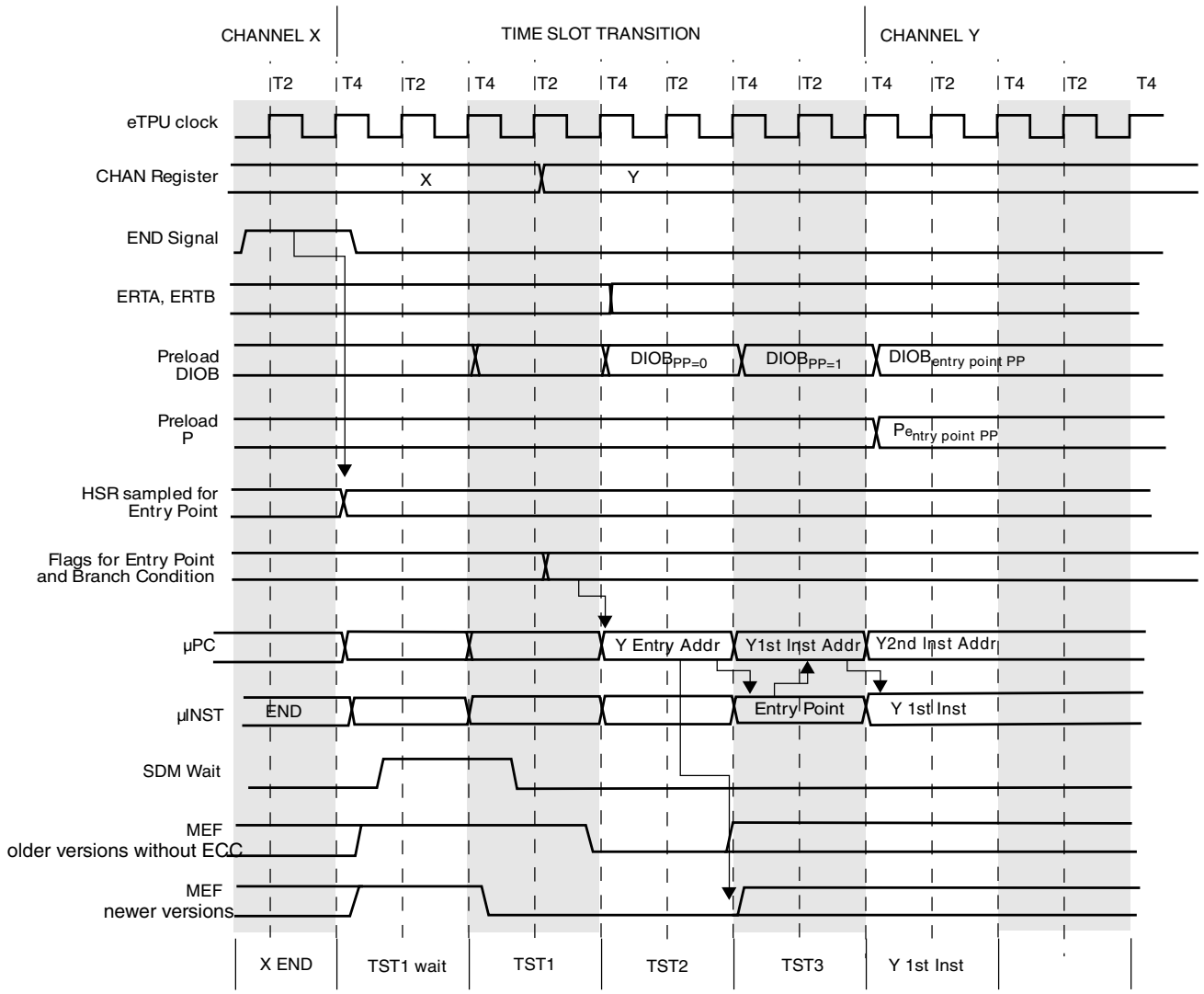


Figure 2-7. TST Timing - 1 Wait-State

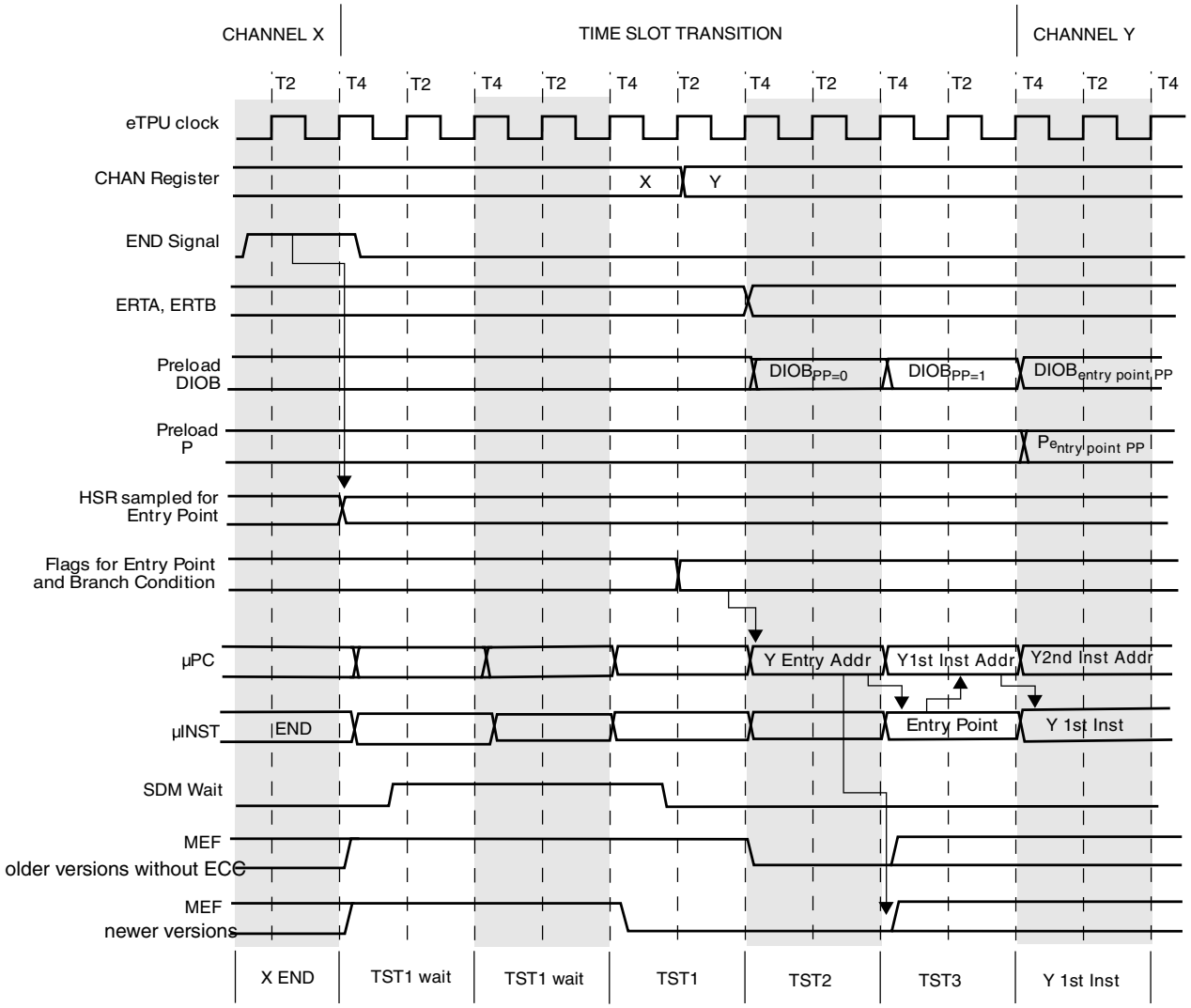


Figure 2-8. TST Timing - 2 Wait-states

For more information on Channel-specific registers and flags, refer to Enhanced Channels. For more information on P, ERTA/B and DIOB registers refer to Registers.

2.14.1.2.1 Thread gap

The MEF timing has changed from previous eTPU/eTPU2 versions, now being reset for two microcycles from the start of TST. Therefore, two consecutive threads with ME=0 serving the same channel could block any match until the end of the second thread (see Figure 2-6). This could cause latency problems and also break compatibility with legacy code, so the thread transition logic was modified to introduce a one-microcycle gap between consecutive threads servicing the same channel, as shown in the following figure. During the gap MEF is set to one, regardless of the previous or the next entry point ME field. The thread gap is inserted only between threads serving the same channel.

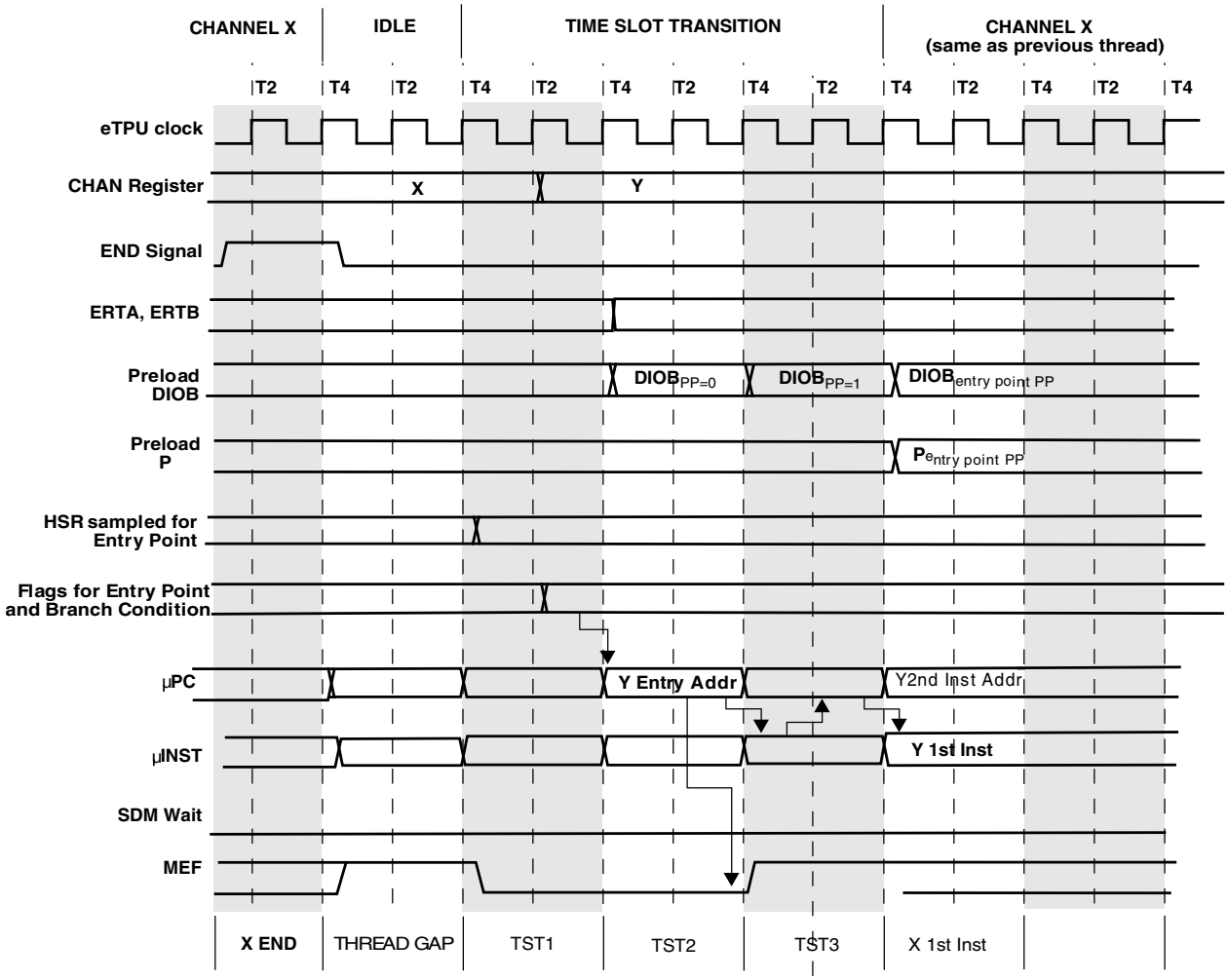


Figure 2-9. Thread gap

2.14.1.3 Thread ending

Threads can finish by either:

- an instruction with the END field active (see Ending Current Thread - END).
- a forced END by host writing to the FEND field in eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2).
- a forced END caused by Watchdog timeout (see Watchdog).
- a forced END caused by a non-correctable memory error (see Memory error support).

2.14.1.4 Watchdog

Each engine has a watchdog mechanism to prevent a thread or a sequence of threads from running too long, impacting the latency of the other channel services. The watchdog is configured through eTPU Watchdog Timer Register for Engine 1 (eTPU_ER_WDTR_ENG1) and eTPU Watchdog Timer Register for Engine 2 (eTPU_ER_WDTR_ENG2). When the watchdog is enabled, an internal counter increments on each microcycle when a thread is executing. If the count is greater than the value specified in the WDCNT field in eTPU Watchdog Timer Register for Engine 1 (eTPU_ER_WDTR_ENG1) and eTPU Watchdog Timer Register for Engine 2 (eTPU_ER_WDTR_ENG2) and a thread is still executing, the watchdog:

1. If the WMO field in eTPU Watchdog Timer Register for Engine 1 (eTPU_ER_WDTR_ENG1) and eTPU Watchdog Timer Register for Engine 2 (eTPU_ER_WDTR_ENG2) is negated, forces an END of the thread, allowing other channels to be serviced
2. Sets the WDS status bit of the serviced channel in eTPU Watchdog Status Register for Engine 1 (eTPU_GCR_WDSR_ENG1) and eTPU Watchdog Status Register for Engine 2 (eTPU_GCR_WDSR_ENG2).
3. If the WMO field in eTPU Watchdog Timer Register for Engine 1 (eTPU_ER_WDTR_ENG1) and eTPU Watchdog Timer Register for Engine 2 (eTPU_ER_WDTR_ENG2) is negated, the channel is disabled, not initiating any thread until its WDS bit is cleared (however, its CPR field in eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2) is not changed).
4. Issues a Global Exception and sets WDTO1 (engine 1) or WDTO2 (engine 2), regardless of the WMO bit state.

If the WMO field in eTPU Watchdog Timer Register for Engine 1 (eTPU_ER_WDTR_ENG1) and eTPU Watchdog Timer Register for Engine 2 (eTPU_ER_WDTR_ENG2) is asserted, the watchdog can be used to monitor the execution time threshold of a thread or thread sequence without interfering with the eTPU code execution.

The watchdog can be configured in one of the following modes, defining how the internal watchdog count is reset:

- **Thread Length Mode**: the watchdog count is reset at the end of each thread.
- **Busy Length Mode**: the watchdog count is reset when the microengine goes idle. A sequence of threads, one right after another, keeps the count running. The counter is also reinitialized when a thread is forced to end, so that a new count begins if another TST initiates at the following microcycle.

The following applies to the watchdog mechanism:

- microcycles during TST and SDM access wait-states (on TST or instruction execution) are counted.
- if the watchdog count equals WDCNT in the last microinstruction (with SDM wait-states or not) of a thread servicing a channel, its WDS bit is not set.
- if the watchdog count expires (gets greater than WDCNT) during the TST, the thread is forced end on its first instruction.
- the watchdog count does not wrap, so that a thread (in thread length mode) or a thread sequence (in busy length mode) that lasts for more than the maximum value of WDCNT does get a forced end.

Note

Watchdog must not be enabled when the microengine enters halt mode.

The counter does not run when the engine is stopped, and resets when the watchdog is disabled.

2.14.2 Host interface

2.14.2.1 System configuration

System Configuration Registers are described in System Configuration memory map and registers. Detailed explanation of the configured functionalities is found throughout Functional description and a specification for the initial configuration sequence is found in Configuration sequence.

2.14.2.2 Interrupts and data transfer requests

2.14.2.2.1 Interrupt types and sources

Each one of the eTPU channels can be a source of two requests: **Channel Interrupt** request and **Data Transfer Request**. Channel Interrupts are targeted to a Host CPU. Data Transfer Requests may be targeted to a data transfer module (for example, a DMA controller). Interrupt and Data Transfer registers are used by the Host to enable interrupts and data transfer requests, indicate their status and service them. Interrupt and Data Transfer requests have the same sets of registers and external signals, and are handled in the same way. They differ only by the fact that Data Transfer Requests are also cleared by the assertion of respective DMA completion acknowledge line.

Note

Interrupt and Data Transfer requests can be cleared even when Engines are in Module Disable Mode, through the Global Channel Registers, and also DMA completion for Data Transfer requests.

Channel Interrupts and Data Transfer Requests can only be issued by eTPU microcode, through one of the Channel Control instruction fields (see Channel Interrupt and Data Transfer Requests).

Both Channel Interrupt and Data Transfer requests can be individually enabled for each channel.

eTPU Interrupt and Data Transfer Registers are mirrored in two organizations: grouped by Channel and grouped by type (interrupt status, interrupt enable, data transfer status, data transfer enable). This allows either "channel-oriented" or "bundled channel" Host interrupt service schemes, or a combination of them. For a detailed description, refer to the Channel Register Layout and Global Channel Registers sections.

eTPU can also assert a **Global Exception** interrupt indicating a global illegal state. There are six possible sources for a Global Exception:

- Execution of an illegal instruction by the microengine (see Illegal instructions). This Global Exception source is flagged by ILF1 and ILF2.
- An SCM signature mismatch detected by the Multiple Input Signature Calculator - MISC. See SCM Test - Multiple Input Signature Calculator. This source is flagged by SCMMISF.

- Microcode request, through microinstruction field CIRC (see Channel Interrupt and Data Transfer Requests). This Global Exception source is flagged by MGE1 (Engine 1) and MGE2 (Engine 2). The cause of this illegal state is application-dependent. The microcode may write an error code into the SDM to indicate the cause of the exception, for instance.
- A Watchdog timeout, flagged by WDTO1 and WDTO2 (see Watchdog).
- An SDM or SCM non-correctable error on a microengine or host access, flagged by SDMERR and SCMERR, respectively (see Memory error support).
- The latency measured by any of either engine's Latency Monitors is greater than the specified maximum, flagged by LTO1 (Engine 1) and LTO2 (Engine 2) (see Latency Monitor).

Global Exceptions cannot be directly disabled within eTPU, except by disabling its sources (Memory Error, SSS, MISC and microcode), and it is cleared by writing 1 to GEC. Clearing Global Exception clears all Global Exception source status bits (ILF1, ILF2, SCMMISF, MGE1, MGE2, SCMERR, SDMERR). If GEC is written 1 at the same time any of the sources issues a Global Exception, both the interrupt and the status bit of that source remains asserted. The assertion of Global Exception by one of the sources above does not prevent the others from asserting it too, so any number of them, in any combination, can be flagged.

Note

There can be a race between the clear of a Global Exception and occurrence of a new set condition, such that the set happens just before the clear and cannot be sensed by the Host. Therefore, Global Exception cannot be used as a normal interrupt source. It should only be used for emergency procedures.

Memory Error Correction support (see Memory error support) has a distinct interrupt source, which can activate on any correctable error occurrence, either to SCM or SDM accesses, but does not activate on non-correctable errors (which assert Global Exception instead). Memory Error Correction interrupt assert when it is enabled through MECIE (see eTPU Memory Error Status Register (eTPU_MESR_MESR)) and either DCERR or CCERR in eTPU Memory Error Status Register (eTPU_MESR_MESR) is asserted. The interrupt is negated when both DCERR and CCERR are negated, or MECIE is negated.

2.14.2.2.2 Interrupt and Data Transfer Request overflow

If a Channel Interrupt was issued, its status bit is still set, and microcode issues another Channel Interrupt, the Interrupt Overflow status bit is set for that channel. Interrupt Overflow status can be checked by the Host in the CIOS field of eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C*n*SCR_ENG1) and eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C*n*SCR_ENG2), mirrored in eTPU Channel Interrupt Overflow Status Register for Engine 1 (eTPU_GCR_CIOSR_ENG1) and eTPU Channel Interrupt Overflow Status Register for Engine 2 (eTPU_GCR_CIOSR_ENG2). Interrupt Overflow status is not cleared automatically when Interrupt Status is cleared. The same mechanism and respective registers (eTPU Channel Data Transfer Request Overflow Status Register for Engine 1 (eTPU_GCR_CDTROSR_ENG1) and eTPU Channel Data Transfer Request Overflow Status Register for Engine 2 (eTPU_GCR_CDTROSR_ENG2)) are available for Data Transfer Requests.

If interrupt is set and cleared at the same time, set prevails and overflow is not altered (keeps the same state as it was before, asserted or not).

Global Exception has no overflow status.

2.14.2.3 Parameter Access

2.14.2.3.1 Parameter Access Widths

From the Host side the SDM address space is mapped in bytes, and each 32-bit parameter occupies 4 contiguous, aligned bytes. The Host can read/write the SDM by 8-, 16-, or 32-bit accesses in aligned addresses.

In 32-bit access, Host can access all 32 bits or only the lower 24 bits with an automatic sign extension (see Parameter Sign Extension area).

2.14.2.3.2 Parameter Addresses and Endianness

To access parameter number xxx, eTPU Microengine(s) would select address xxx. The Host would add *(xxx*4)* to the SDM base address to access the same parameter. For example, parameter 0x101 is seen by the Host in *(SDM base address +0x404)* . An example of SDM memory map is shown in Figure 2-10 . The Host can access the SDM with a 32-bit-wide bus cycle to a four-byte aligned address, 16-bit-wide bus cycle to a two-byte aligned address, or 8-bit wide bus cycle to any byte address.

The address of the 24-bit parameters and the most significant byte depends on the endianness of the MCU. For more details, see the Endianness.

2.14.2.3.3 Parameter concurrency

Host accesses to parameters may occur in parallel with eTPU Microengine accesses. Readings taken from a group of parameters while they are being simultaneously updated may lack coherency. eTPU provides mechanisms to ensure parameter coherency in accesses from both Host side and Microengine side, including the use of a coherent dual-parameter transfer mechanism, described in detail on Parameter sharing and coherency.

2.14.2.3.4 Parameter Sign Extension area

The SDM address space to the Host is mirrored in a Parameter Sign Extension (PSE) area (see Table 2-3). Accesses from the Host to the PSE area differ from accesses to the standard SDM address space as follows:

- **Writes**: the most significant byte of the parameters is not written, and the SDM retains the old byte value, regardless of the Host access size.
- **Reads**: the most significant bit of the 24-bit parameter (that is, the MSB of the second most significant 32-bit parameter byte) is repeated in the 8 most significant bits of the read value on all 32-bit reads and most significant 16- and 8-bit reads.

The same parameters written in the standard SDM address space are read from the PSE area with the same offsets, and vice-versa. See Table 2-117 for a reference of the address offsets in big and little endian machines.

This feature reliefs the Host from extending the signal of 24-bit eTPU parameters before calculations, and from read-modify-write accesses to modify 24-bit parameters at the SDM.

2.14.2.4 SDM organization

The SDM internal partition for channel allocation is dynamic and programmed in the Channel Registers (see eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2)).

The Host application is responsible for allocating a different parameter base address to each channel during the initial eTPU configuration, and to allocate enough parameters for the selected Function, with no unintentional overlapping between parameters of different functions.

Besides channel parameters, global areas may have to be allocated for parameters that are shared by more than one channel, in one or both Engines. Also, temporary parameter areas should be reserved to be used by the coherent parameter transfer mechanisms described in Parameter sharing and coherency, if necessary. An SDM organization example is shown in the following figure.

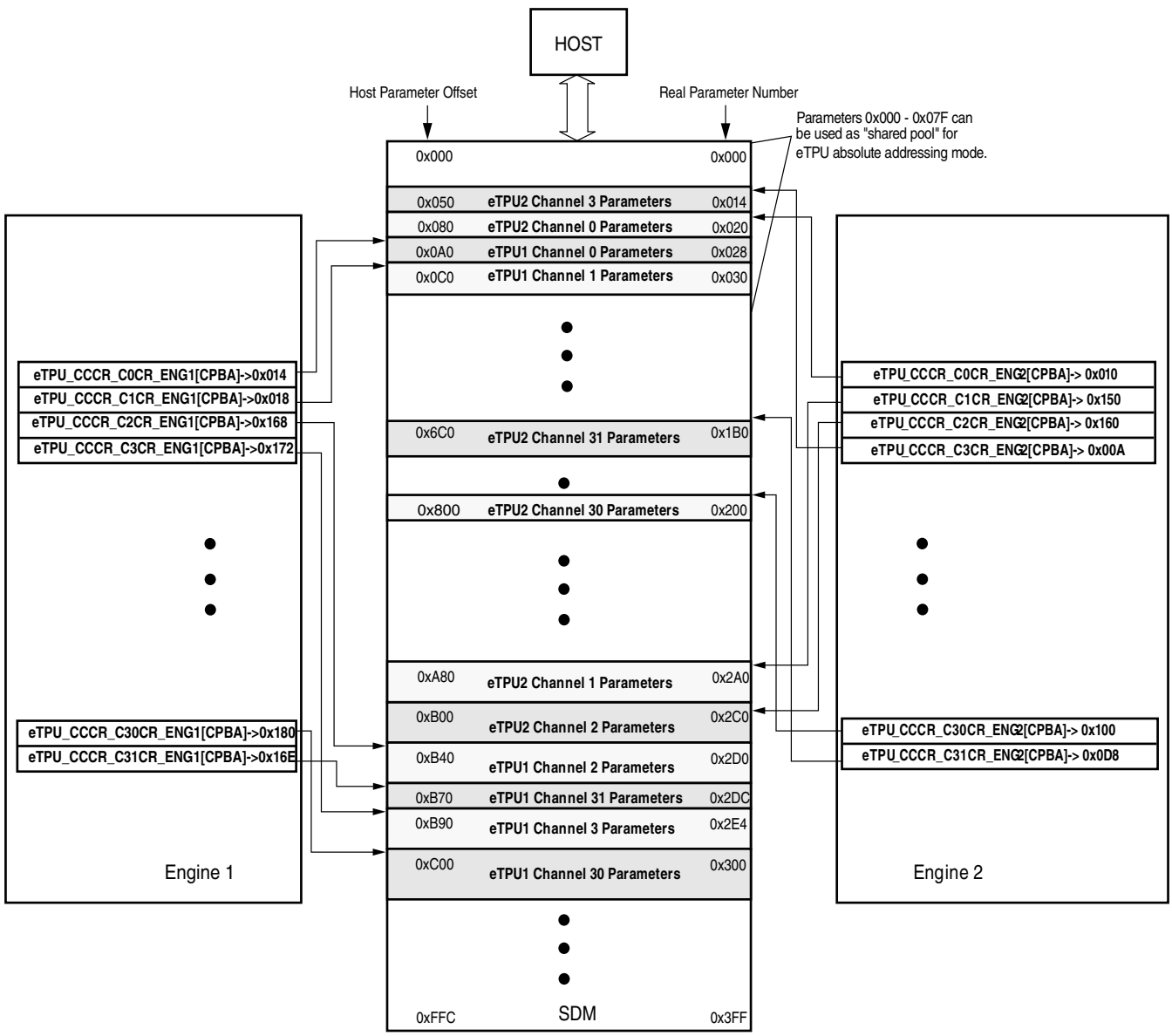


Figure 2-10. SDM Organization Example (4 KB)

Single-engine eTPU or dual eTPU system may require less parameters than the maximum number provided by the SDM. Since the SDM partition is fully dynamic, there is no limitation of fixed channel addresses, and the reduced array can be fully utilized.

2.14.2.5 Host Service Requests

Host CPU can request immediate service from a channel by writing a non-zero value to the Host Service Request register field - HSR (see eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C*n*HSRR_ENG1) and eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C*n*HSRR_ENG2)). There is one HSR field for each channel, so that writing to it generates a Service Request to the respective channel only. A zero value in HSR means no Host Service Request is pending for the channel.

HSR value turns to 000 automatically at the end of microengine service for that channel, but only if the thread started due to an HSR.

The meaning of a non-zero HSR value depends on the Function assigned for the channel. These bits are part of the conditions which select the Function entry point, and cannot be tested by microcode. For more details, refer to Entry Points.

If Host writes HSR=000 when a thread for the same channel is already running, the thread runs until the end and is not aborted. If Host writes HSR>000 when an HSR thread for the same channel is already running, HSR value resets at the end of the thread, and no new HSR will be pending. If HSR is written before its value is resolved by the scheduler during TST, the entry point will obey the new HSR value, and if this new value is 000, no service thread is executed for the HSR.

The scheduling of HSRs is completely asynchronous with Host accesses, and there is no race-free manner to change an HSR value before service thread execution, so generally the safe way is: write HSR>0 only when HSR=0. Error recovery or emergency host procedures may require one to the safely abort service and reset channel state when an HSR is already pending or executing. In these cases, the procedure below should be followed:

1. Disable the channel, writing CPR=00 in eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2). That will prevent any pending HSR to be serviced.
2. Check if the channel is currently being serviced, reading its service status bit in eTPU Channel Service Status Register for Engine 1 (eTPU_GCR_CSSR_ENG1) and eTPU Channel Service Status Register for Engine 2 (eTPU_GCR_CSSR_ENG2). If

it is, wait for the time necessary to finish the service pending, or check again until HSR == 0, or channel service bit in eTPU Channel Service Status Register for Engine 1 (eTPU_GCR_CSSR_ENG1) and eTPU Channel Service Status Register for Engine 2 (eTPU_GCR_CSSR_ENG2) is cleared.

3. Write HSR with the error recover value. This value should, possibly combined with other host-defined flags in SDM or FM bits, initiate a channel reset or error recovery procedure.
4. Re-enable the channel, writing CPR value > 0 in eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2).

2.14.2.6 SCM access

Only Host can access SCM as data. Depending on the specific device, SCM may be implemented as a RAM or ROM. This determines Host accesses to the SCM as shown below.

2.14.2.6.1 SCM RAM implementations

When SCM is implemented as RAM, the Host may read or write to SCM by setting VIS =1. If VIS=0 and Host tries to access SCM space, a bus error is issued, writes are ineffective and read data is meaningless. Both Engines must be stopped or halted to set VIS=1.

Note

It is necessary to turn VIS bit on to set software breakpoints (see Software breakpoints).

2.14.2.6.2 SCM low power

SCM turns off its internal clocks when both Engines are stopped (STF field in eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2) asserted), VIS, and MISC is not enabled (SCMMISEN=0). The SCM clocks are automatically turned on if either one of the STF bits is negated or VIS turns to 1, or SCMMISEN turns to 1.

SCM clocks are not turned off if any of the Engines is not stopped, even if they are both halted.

The conditions for SCM Clocks and MISC activation are summarized in the following table.

Table 2-26. SCM clocks and MISC activation

STF in eTPU_SCR_ECR_ENG1	STF in eTPU_SCR_ECR_ENG2	VIS in eTPU_SCR_MCR	SCMMISEN in eTPU_SCR_MCR	SCM clocks	MISC
0	x	0[1]	1	On	On
0	x	0[1]	0	On	off
x	0	0[1]	1	On	On
x	0	0[1]	0	On	off
1	1	0	0	off	off
1	1	0	1	On	On
1[2]	1[2]	1	0	On	off
1[2]	1[2]	1	1	On	off[3]
0	0	x	0	On	off
0	0	x	1	On	On

1. VIS cannot be written 1 if eTPU_SCR_ECR_ENG1[STF]=0 or eTPU_SCR_ECR_ENG2[STF]=0, and both HLTF bits are 0.
2. If VIS=1, neither MDIS can be written 0 nor the Engine leaves Stop Mode, regardless of device stop request .
3. MISC resets and stays so when VIS=1, restarting automatically when VIS goes 0 if SCMMISEN=1.

2.14.2.6.3 SCM Off-range data

When read accesses are made, either by the Host or by a microengine, to addresses above the limit corresponding to SCMSIZE, the value read comes from eTPU SCM Off-range Data Register (eTPU_SCR_SCMOFFDATAR). The Host can program the register at initialization with an opcode value with operations that try to protect or recover the system from runaway code, for instance: terminate the thread, clear channel flags, disable match and transition service requests, issue an interrupt, jump to an error recovery procedure[12] . Writes to unimplemented addresses do not return error and can write on unspecified mirror addresses, so they should be avoided.

2.14.2.7 Bus error conditions

Follows a summary of the possible causes of bus errors on host accesses to the eTPU memory space:

- Read or write access to the SCM with VIS =0.

12. Only part of these suggested operations can be parallelized in a single instruction, see Microinstruction formats.

- Read or write access to the channel registers with eTPU_SCR_ECR_ENG*n* bits MDIS=1, or bit STF=1 (see eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2)).
- Write access to the Time Base Registers with eTPU_SCR_ECR_ENG*n* bits MDIS=1, or bit STF=1 (see eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2)).
- Non-correctable errors in the SCM on reads of any size or writes less than 32-bits wide when CEDD =0.
- Non-correctable errors in the SDM on reads when DEDD =0.
- Accesses to the Shared Subsystem Area, when propagating a bus error issued by the attached subsystem.

2.14.3 Scheduler

Every Function is composed of one or more Threads. A Thread consists of a group of instructions that, once begins execution, cannot be interrupted by host or channel events. Each active channel intents to be serviced, being granted time for Thread execution. Since one microengine handles several channels operating concurrently, the Function threads must be executed serially.

The task of the Scheduler is to recognize and prioritize the channels needing service and to grant execution time to each channel. The time given to an individual Thread for execution or service is called a **Time Slot**. The duration of a time slot is determined by the number of instructions executed in the Thread plus SDM wait-states received, and varies in length.

At any time, an arbitrary number of channels can require service. To request service, channel logic, eTPU microcode or Host application notifies the Scheduler by issuing a Service Request.

2.14.3.1 Channel enabling and priority assignment

Every channel is assigned one of three priority levels - high, middle, or low - by the Host CPU, through the Channel Configuration Register field CPR (seeeTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n

Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2)). These registers are also used to disable the channel, which is equivalent to assigning it a "null" priority. In this case, the Scheduler does not grant any of its Service Requests.

It is possible to change the channel priority level or disable it dynamically. If the Host disables a channel when it is currently being serviced, channel service thread will complete. This means that it is possible for the output level of a channel signal to change, or a Host interrupt occur, even after its priority register was written to "null". For instance, if an output transition is scheduled, the transition will occur even after the channel is disabled.

Service requests previously pending or that occur while a channel is disabled remain asserted while the channel is disabled, and are serviced if the channel is enabled again, in due time determined by the priority scheme and concurrent requests from other channels. Channels are disabled after reset, and it is recommended to configure a Host Service Request for initialization of a channel before that channel is enabled to active priority (see Initialization/application information).

2.14.3.2 Channel priority schemes

The Scheduler holds a Service Grant register with one bit for each channel. Once the Scheduler grants a time slot to channel, the Service Grant bit for that channel is asserted in the Service Grant register. When the Service Grant bit of a channel is set, the channel may request new service but is not serviced again before its Service Grant bit is cleared.

When all channels in a same priority level are serviced, their Service Grant bits are cleared at the end of the thread, one eTPU clock before the next serviced channel is calculated, according to the scheme below[13] :

- Clear all grant bits of priority High if all channels of that priority that are requesting have their grant bits in 1.
- Clear all grant bits of priority Medium if all channels of that priority that are requesting have their grant bits in 1.
- Clear all grant bits of priority Low if all channels of that priority that are requesting have their grant bits in 1.
- Clear all grant bits of disabled channels.

13. Grant bits are also cleared in the next clock, when the service channel is chosen, or when the microengine is idle, using the same scheme.

This scheme assures that no channel is left with its grant bit forever asserted (preventing it from being serviced again), even if the channel priorities are reassigned during the execution.

Priority level is determined based on the maximum latency desired for each channel. A channel having a Function that requires the most frequent or more immediate service should be allocated a high priority level.

The eTPU employs a **primary** and a **secondary priority scheme**. These two schemes ensure frequent servicing of high-demand Functions and ensure a minimum time allocation to all channels requesting service, regardless of their priority level. The primary scheme prioritizes requesting channels that have different priority levels; the secondary scheme prioritizes requesting channels that have the same priority level.

Initially, a channel requests service and is granted a time slot by the Scheduler: Service Grant bit is asserted. If only high-level channels constantly receive service first because of their priority level, middle- and low-level channels would only be serviced by default, i.e., if no high-level channels request service. To ensure that each priority level receives an opportunity for servicing, every time slot has a fixed priority level that the Scheduler honors first. Divided into sets of seven, time slots are numbered from one to seven. The figure below illustrates the numbered time slots in sets of seven (fields A and B) and identifies their assigned default priority level. The high level has more time slots than the middle and low levels. Out of every seven time slots available, four are assigned to honor high-level channels first, two are assigned to honor middle-level channels first, and one is assigned to honor low-level channels first. Only one request (in each Engine) is serviced per time slot. When no channel requests service and the microengine is idle the priority scheme is initialized to time slot one, to prevent priority inversion on the next request[14].

14. Priority inversion would occur in the following situation: no channel is requesting service, and the current time slot is primarily assigned to a low-priority channel. If the Scheduler was not reset to time slot one and two channels requested service at the same time, one with high priority and the other with low priority, the channel to be serviced would be the low-priority channel.

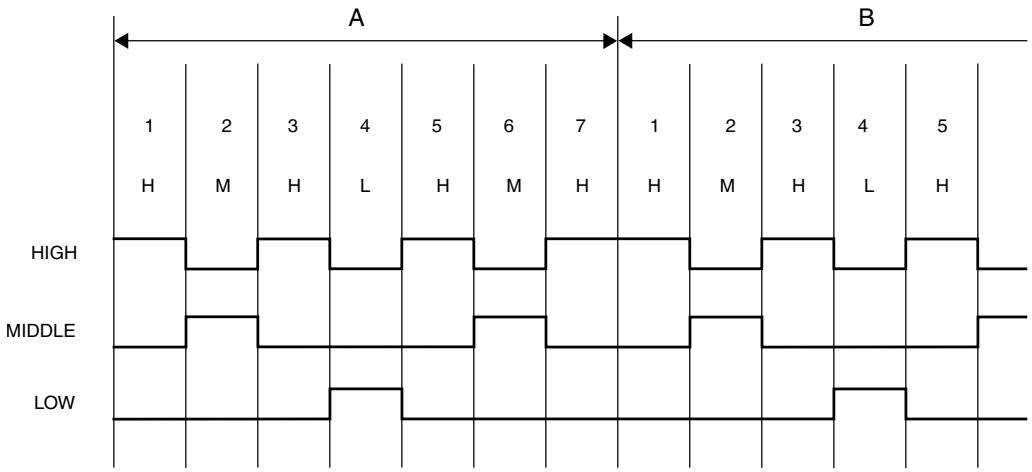


Figure 2-11. Time Slot Priority levels

2.14.3.2.1 Primary Scheme - Priority Among Channels on Different Levels

Although time slot priority assignment is fixed, the servicing priority is not. The primary scheme acknowledges the priority level assigned to a time slot, granting service first to a channel having the same priority. In Channel priority schemes, time slot one has a high-level assignment; therefore, a high-level channel requesting service is recognized first. However, if no high-level channel requests service, the Scheduler recognizes a requesting middle-level channel. If this level has no request, the Scheduler continues to the low-level. If no requests occur, the Scheduler truncates the seven state cycle and starts a new cycle at time slot one, waiting for the first request. Granting service to a different-level channel is called priority passing. The order of passing always gives the highest priority to the assigned level, and the second priority to the higher of the remaining requesting priority levels as shown in the following table.

Table 2-27. Priority Passing

Assigned Priority Level	Next Priority Level	Next Priority Level
High	Middle	Low
Middle	High	Low
Low	High	Middle

When priority is passed to another level, that level is serviced and the fixed-priority-level sequence is resumed with the next time slot.

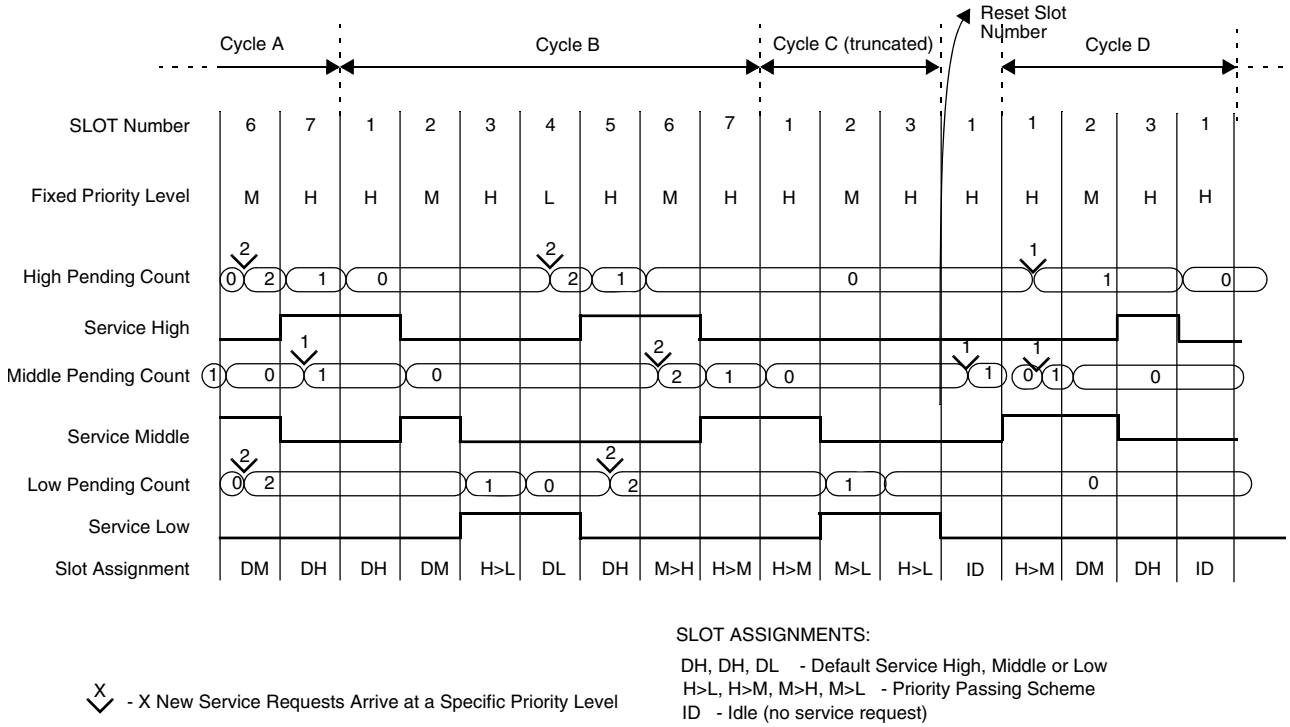


Figure 2-12. Priority Passing Example

Examples of priority passing are shown in the above figure. Each cycle contains seven time slots (or less if no service request exist). In cycle B, no high-level or middle-level service requests are present before time slot three which is assigned by default to high-level priority. Thus, time slot three is passed to the low level. In cycle B there are also no middle-level service requests before time slot six, so it passes the priority to a requesting high-level channel. During time slot six no more high level requests are left, but two new middle-level requests arrive, and there are also three low level pending service requests. Thus, time slot seven of cycle B and time slot one of cycle C are passed to the middle-level which is the next priority level after high. Time slots two and three of cycle C are passed to the low level which contains the three remaining channel service requests. At time slot three of cycle C the last low level request is serviced, and the Scheduler passes to idle state. At this point the cycle C is truncated and the Scheduler passes to time slot one of cycle D.

2.14.3.2.2 Priority passing disabling

The priority passing scheme allows a case where a high priority channel looses to a lower priority one right after another lower priority has been serviced, exemplified in the Cycle D on Figure 2-12. A middle priority channel wins time slot 1 due to priority passing from high to middle. While it is being serviced, two new service requests arrive, one high and one middle priority. The high priority request looses to the middle one on next time slot 2 by default priority assignment.

This priority inversion can be avoided by setting the SPPDIS field in the eTPU_SCR_ECR_ENG*n* registers (see eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2)), which disables the priority passing mechanism. When priority passing is disabled, at the end of the thread the slot number is incremented until a time slot that matches the priority of one of the requesting channel(s). The time slot advance takes no extra clocks. If no channel requests service, the time slot counter stays at time slot 1. The priority selection scheme with disabled priority passing is summarized in the following table.

Table 2-28. Priority Passing Disabling

At the endof time slot	servicing priority	if any request of priority	service it on time slot	else if any request of priority	service it on time slot	else if any request of priority	service it on time slot
1	High	Medium	2	High	3	Low	4
2	Medium	High	3	Low	4	Medium	6
3	High	Low	4	High	5	Medium	6
4	Low	High	5	Medium	6	Low	4
5	High	Medium	6	High	7	Low	4
6	Medium	High	7	Medium	2	Low	4
7	High	High	1	Medium	2	Low	4

An example of the priority passing disabling scheme is illustrated in the figure below. The sequence of service requests is the same as in the example of Figure 2-12, and although the time slot incrementing differs, the priorities granted are the same for cycle B. Cycle C has one of the low priority channels serviced before the second middle one. Cycle D, however, no longer has the priority inversion.

In cycle B, after the time slot 2 only a low priority request remains, so the time slot count advances directly to 4, which has a low priority assigned. Time slot keeps on 4 for the next service, as only a low priority request remains also, and only time slot 4 is assigned to low. Two high priority services contend for the next time slot 5 (assigned to High). The second high priority channel is serviced on the next time slot, jumped to 7 because there is no middle request, ending cycle B. Cycle C starts with time slot 2, as there are no high priority requests and two middle and two low ones. After the first middle service, time slot count skips 3 assigned to high (no high requests), and services a low priority channel on time slot 4. It follows the same scheme until there are no other requests and cycle C is truncated, resetting the time slot counter to 1.

Cycle D begins with a middle request, jumping to time slot 2. During this service two requests arrive, one high and one middle. Unlike what happened with priority passing, the next serviced is the high priority channel, as the time slot increments to 3. The second middle priority channel request in cycle D is finally serviced next, on time slot 6.

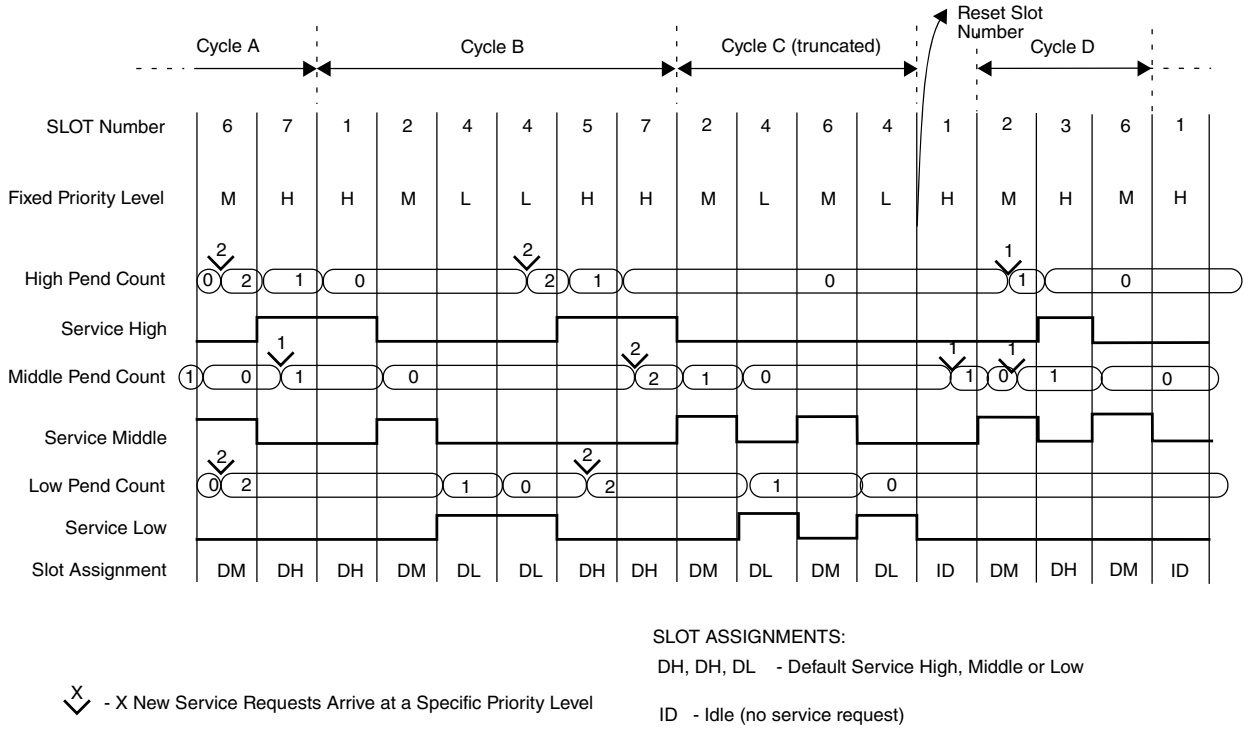


Figure 2-13. Priority Passing Disabling Example

2.14.3.2.3 Secondary Scheme - Priority Among Channels on the Same Level

Because channels can randomly request service, channels having the same priority level will inevitably request service simultaneously. A secondary scheme prioritizes these requests. The Scheduler services channels on each of the three priority levels, beginning with the lowest numbered channel on that level.

2.14.3.2.4 Priority Scheme Example

The overall priority scheme simultaneously incorporates both primary and secondary schemes. Combining both schemes in the following example conveys their correlation.

1. One high-priority and one low priority channels request service, while the Scheduler is in time slot one. Having its service request bit asserted, a single high-level channel is granted the time slot, which has high-level priority (primary scheme) and its service grant bit is asserted. At the end of the thread, the service grant bit is negated (no more requests of high priority level channels).

2. The Scheduler proceeds to time slot two, which has middle-level priority; however, no middle-level channel is requesting service. Priority is passed to the high level, but no high-level channel is requesting service; therefore, priority is passed again, and service is granted to the single requesting low-level channel. Once serviced, this channel's grant bit is negated (no more low-level requests).
3. The Scheduler resumes with the fixed-priority sequence on time slot three; however, no channels are requesting service. The Scheduler returns to time slot one, waiting for requests.
4. Two high-level and two middle-level channels simultaneously request service. Being in time slot one which is assigned high priority, the Scheduler finds the lowest numbered high-level channel (secondary scheme) and selects it for service. This channel's service grant bit is asserted.
5. The Scheduler continues to time slot two, which has middle priority (primary scheme), and allocates the slot to the lowest numbered middle-level channel requesting service (secondary scheme). The Scheduler notes the still unserviced middle-level channel and proceeds to time slot three.
6. Time slot three is allocated for high priority. The slot is allocated to the remaining unserviced high-priority channel, and the channel's service grant bit is asserted. The Scheduler checks again at the end of the thread. All service grant bits of high-level requested channels are asserted; therefore, all high-priority channels that requested have been allocated execution time. Under this condition, all service grant bits of the high-level serviced channels are negated. The Scheduler proceeds to time slot four.
7. Time slot four is allocated for low-priority channel; however, no low-level channel is requesting service. Priority is passed to the high level, but no high-level channel is requesting service; therefore, priority is passed again, and service is granted to the remaining middle-level channel which requests service. This channel's service grant bit is asserted. The Scheduler checks again at the end of the thread. All grant bits of middle-level requested channels are asserted; therefore, all middle-priority channels have been allocated execution time. Under this condition, all service grant bits of the middle-level serviced channels are negated. The Scheduler proceeds to time slot five. Meanwhile a low priority channel requests service.
8. Time slot five is allocated for high-priority channels, but there are no more requests from high-priority or middle priority channels. The single low-level channel which required service is granted time slot five. Once serviced, the channel's service grant bit is asserted. Next, the service grant bit is negated (no more requests of low priority level channels).

9. The Scheduler resumes with the fixed-priority sequence on time slot six; however, no channels are requesting service. The Scheduler returns to time slot one and waits for requests.

2.14.3.3 Time slot latency

Latency is the amount of time between a service request and the beginning of service on that channel. The following factors affect latency:

- Number of active channels
- Number of channels on a priority level
- Number of available time slots on a priority level
- Number of microcycles required to execute a thread of a Function
- Number of SDM accesses during execution of a Function thread
- eTPU clock frequency

Each time slot may require a different number of microcycles, depending on the thread of a Function to be executed. This variation is shown in the figure below.

For more details on latency evaluation, see Estimating Worst Case Latency.

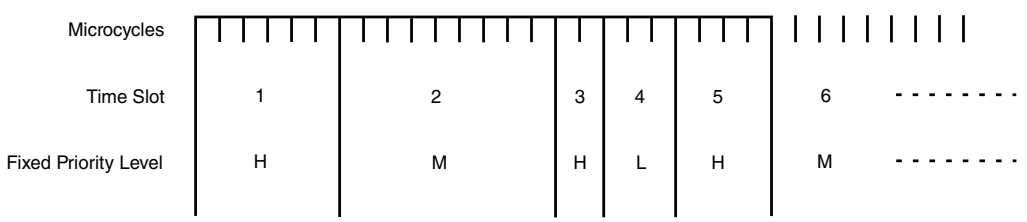


Figure 2-14. Time-slot variation

2.14.4 Parameter sharing and coherency

SDM can be concurrently accessed by Host and Microengines (two in a Dual eTPU Engine system). In general, there is no guaranteed order by which a group of parameters is accessed, which may lead to a lack of internal consistency if two or more related parameters are read when only part of them is updated.

eTPU provides mechanisms to guarantee parameter coherency. The most generic mechanisms for Host-eTPU coherency, suitable for any number of parameters, are:

- the use of Transfer Service Thread mechanism.
- the mailbox (or "software semaphore") mechanism.

These mechanisms, described in Multiple parameter coherency methods, use microcode to transfer parameters from temporary buffers in SDM to their definitive locations (or vice-versa). These methods have the disadvantage of wasting processing and code memory resources.

eTPU also provides a **Coherent Dual-parameter Controller - CDC** - mechanism. It is used by Host to coherently transfer pairs of parameters from/to a parameter buffer located on SDM to/from the locations on SDM where parameters are accessed directly by the channels. Coherency is guaranteed by SDM access arbitration. Although limited to two parameters only, it has lower latency and wastes no microengine resources[15]. CDC usage is described in Coherent Dual-parameter Controller (CDC).

For parameters shared by both Engines, eTPU provides **Hardware Semaphores**. Coherency is assured given the semaphores are used to prevent concurrent access to the changing parameters. Microengine can request semaphores using specific microinstructions (see Semaphore Operations). Hardware Semaphores are described in detail in Hardware Semaphores.

Neither Host nor CDC have access to the hardware semaphores, but they can be combined with microcode transfer mechanisms if Host must coherently access parameters which are also shared by both Engines.

In order to ensure coherent access to a group of parameters by two or more contenders, **each contender must have atomic access** to the shared parameters. Atomicity conditions are discussed in Host side atomic access and Microengine Side Atomic Accesses.

2.14.4.1 Host side atomic access

Host side atomic accesses can be achieved by either of following ways:

- For one parameter, the SDM should be accessed by 32-bit-wide data transfers to ensure coherency

15. A microengine access to the SDM in the moment CDC is performing the transfer may suffer a maximum of two wait-states.

- For two parameters only, using the Coherent Dual Parameter Controller.
- Indirectly, for any number of parameters, by requesting microcode to coherently access SDM in its behalf. The host side atomicity problem becomes, then, a microengine side atomicity problem. Some methods that use this approach to achieve coherency are described in Multiple parameter coherency methods.

2.14.4.2 Microengine Side Atomic Accesses

2.14.4.2.1 Microengine Single Parameter Atomicity

SDM should be accessed by 32-bit-wide data transfers to ensure atomicity for 32-bit parameters. This applies either to Host-Microengine coherency or Microengine-Microengine coherency in a dual eTPU Engine system.

2.14.4.2.2 Microengine Dual Parameter Atomicity

Microengine has the ability to access two parameters coherently in back-to-back accesses, at random addresses: once it accesses SDM, it has priority over Host for another access in the next microcycle (see SDM arbitration). Note that it applies **only to Microengine-Host** coherency. For Microengine-Microengine coherency in a dual eTPU Engine system, one must use Hardware Semaphores (see Hardware Semaphores).

Microengine dual back-to-back accesses are guaranteed to be atomic in relation to Host slave accesses or Coherent Dual-parameter Controller, regardless of semaphore usage: Host or CDC accesses cannot break-up a back-to-back Microengine access, neither Microengine can break a CDC transfer, due to the SDM arbitration mechanism described in SDM arbitration.

Atomicity is not guaranteed if microengine enters halt state in the middle of a back-to-back access (see Microengine Halt State): Host can access SDM while microengine is halted in the middle of a back-to-back access.

2.14.4.2.3 Microengine Side Multiple Atomicity

Hardware Semaphores must be used for Microengine-Microengine coherency (more than 1 parameter) since two or more accesses from one Microengine are not atomic with respect to the other.

For multiple Microengine-Host coherency, the software methods described in Multiple parameter coherency methods, or similar ones, must be used. Some of these methods rely on the fact that parameter access of a thread is atomic in relation to another thread in the same Engine, since a thread cannot be suspended (preempted).

For 1 parameter coherent access, or dual parameter coherency between only one Microengine and Host, the alternatives shown in previous sections apply.

2.14.4.3 Coherent Dual-parameter Controller (CDC)

Dual-parameter coherency is supported by a Coherent Dual-parameter Controller hardware (CDC), which contends with microengine for SDM access. CDC atomically transfers, upon Host's command, two parameters from one area of the SDM to another. One area is a temporary (buffer) area, where the two parameters are directly read or written by the Host. This temporary area has to begin in an SDM address multiple of 2 words, and the two parameters must be sequential. The other area is the channel parameter area where the microcode normally accesses the parameters, usually with the channel relative address mode (see SDM Addressing Modes). In this area, the parameters transferred by CDC don't have to be sequential. A transfer from the temporary area to the channel area, when the Host sends data to the channel, is called a **write transfer**. Inversely, in a **read transfer** the parameters are copied from the channel area to the temporary area (channel to Host).

Coherency is guaranteed by the SDM access contention rules implemented in the SDM arbiter (see SDM arbitration). CDC transfers are coherent in respect to the two Engines, so the target parameters in the channel area may be shared by channels on them both. During CDC operation, the Host may suffer from 3 up to 11 eTPU clocks wait states[16], and the Microengine(s) may suffer up to 2 microcycle wait-states[17]. CDC accesses are atomic with respect to Microengine(s) accesses to the SDM. Even when neither engine is in TST, CDC may suffer up to 4 eTPU clock internal wait-states from SDM arbiter, meaning 9 slave wait-states to Host, so that it does not break atomic back-to-back accesses from microengine(s). CDC also cannot break TST preload accesses. Host can initiate CDC back-to-back transfers: there is no need of idle slave cycles between two transfers.

16. The maximum number of Host wait states on CDC occurs when both microengines overlap their TSTs, delayed 3 eTPU clocks from each other.
17. One microcycle takes two eTPU clocks. Microengines get wait-states in multiples of microcycles, while Host and CDC wait-states are multiples of eTPU clocks.

Note

SDM memory error detection and correction does not affect the number of wait-states to host due to CDC (see Memory error support).

CDC can be used to atomically fix SDM soft errors. For more details, see Error fixing.

2.14.4.3.1 CDC programming

The Coherent Dual-parameter Controller Register (see eTPU Coherent Dual-Parameter Controller Register (eTPU_SCR_CDCR)) is used to configure and initiate CDC transfers between the temporary area and channel parameter area. Host asserts STS bit in order to start the data transfer. CDC then contends for the SDM and starts the transfer. When the data transfer is complete, STS returns to 0. Host receives wait-states for writing STS=1 while CDC contends for SDM and during the transfer. The write access ends when CDC finishes the transfer. Host receives wait-states during the CDC transfer. If Host writes eTPU Coherent Dual-Parameter Controller Register (eTPU_SCR_CDCR) with STS=0 or does not write the STS byte, the CDC transfer does not occur. CDC programming can be summarized as follows:

1. If it is a write transfer, i.e., from Host to channel, write the two parameters into temporary area.
2. Write STS =1 and the remaining CDC programming parameters: parameter width (32 or 24 bits, field PWIDTH), transfer direction (read or write, field WR), temporary parameter area base address (field PBBASE), and the absolute addresses of the parameters to be transferred (concatenation of the fields CTBASE and PARM0/1).
3. If it is a read transfer -- that is, from channel to host -- read the two parameters from the temporary area into Host memory/registers.

2.14.4.4 Hardware Semaphores

eTPU provides Hardware Semaphores accessible by the Microengine only. It is the responsibility of the application to ensure proper use of the semaphores (i.e., agree upon a specific semaphore and use it properly, to ensure coherency).

The eTPU microinstruction set has support for locking and freeing the semaphores, described in Semaphore Operations, and this is the only way to access them.

There are four semaphores available, which reduces the amount of collisions by assigning unrelated data transfers to different semaphores. Semaphores are used for parameters which can be shared by channels in different Engines, and for Engine-to-Engine synchronization. Semaphores are also the only way to ensure coherent access to parameters shared between the two Microengines.

Attempting to lock one semaphore (even not successfully) frees the other locked by the same Engine, ensuring one can lock just one semaphore at a time. That prevents deadlock conditions between the two Engines.

Microcode END command or Engine being in idle state (no thread executing) automatically releases all semaphores from one Engine side, even if a semaphore lock is done in parallel. However, it is recommended to write the microcode in a way which locks semaphores for the shortest required period, and frees them without waiting for the END command, to improve the performance of the other microengine. Semaphores are free after reset. An Engine can only free a sempaphore locked by itself.

Semaphore lock requests are always **non-blocking**, in the sense that they do not suspend the requester in case the semaphore is already locked. The semaphore status after the lock request - indicating if it was successfully locked or not - must be tested through the SMLCK microengine branch condition (see Branch conditions).

2.14.4.5 SDM arbitration

Up to four entities can access SDM:

- Two Microengines (in a dual eTPU Engine system)
- The Coherent Dual-parameter Controller (CDC)
- The Host CPU (direct memory-mapped access)

The following rules specify the access priorities for contended access. They keep compatibility with the TPU3 dual parameter access atomicity, but only between the microengine and CDC (not Host accesses through slave bus).

1. Microengine accesses from the two eTPU Engines are interleaved between each other, but not with Host or CDC accesses;
2. The eTPU microengine(s) gives priority for SDM accesses to either the Host CPU or the CDC under any of the following conditions:
    a. The microengine has completed accessing the second parameter in a back-to-back SDM access[18].

b. The SDM was not accessed during the last arbitration slot for the microengine and the host does not loose the access to the other engine in the current arbitration slot[19]

c. CDC is transferring data, after its first (read) access. Note that the CDC can be in middle of a data transfer of another pair of parameters, unrelated to the ones that microengine tries to access.

3. The eTPU microengine takes priority for SDM accesses under either of the following conditions:

a. the Host CPU or CDC has done a data transfer during the last access arbitration slot for the engine[19]. Also, the Host CPU does not hold a pending access against the other eTPU microengine.

b. the microengine is arbitrating for the access of its second parameter in a back-to-back access[1]. All pairs of back-to-back parameter accesses are coherent with respect to Host and CDC (not to the other microengine).

The direction (read or write) of any individual access by Host or microengine is irrelevant to the arbitration. The use of Normal or PSE SDM area by the Host is also irrelevant to the arbitration.

The first parameter preloading in a TST is considered first access by the arbiter, regardless of any access made at the END microinstruction of the previous thread, that is, the last access of a thread and the first preload are never considered a back-to-back access. On the other hand, the TST preload accesses are considered back-to-back and are, therefore, atomic with respect to Host or CDC.

Note

The Zero SDM operation (see Zero SDM Operation) is considered an SDM access for arbitration purposes both on writes and reads; the fact that read SDM data is discarded is irrelevant for arbitration.

18. If microengine tries to access the SDM in the following microcycles, the third and fourth consecutive accesses are considered the first and second of a new back-to-back dual access.

19. The microengine access slot is between its own T4 and T2 edges (see Microcycle and I/O timing).

2.14.5 Enhanced Channels

Enhanced Channels comprise hardware support for input digital signal processing and output signal generation. Each Channel is associated with one input and one output signal. Enhanced Channel logic is combined with Function microcode (and optionally Angle Mode logic) to implement Channel I/O functionality.

eTPU's Enhanced Channels are capable of **dual action**, meaning that each channel logic can handle two events at different times and/or cause two separated actions - these actions and events can be mutually dependent (with the first either blocking or enabling the other), or both independent, depending on the programmed **Channel Mode**.

Each Enhanced Channel contains event logic containing two **Event Register sets**, each set supporting one input and/or output action, the pair implementing dual action support. Each Event Register set contains two 24 bit registers: Match and Capture. The **Match register** holds the pending match value which is compared against one of the two time bases by an equal-only/greater-equal comparator. The **Capture register** captures one of the two time bases as a result of a Match or Transition detection. Service Requests are issued on particular combination of match and capture events, defined by the selected Channel Mode.

In the context of the eTPU channels, a **Match** is a comparison between a time base value and a channel Match register. If those two values are coincident, or the time base value is greater than the value of the Match register, a **Match Event occurs**. Depending on the channel mode of operation and current state of the channel logic, the match event may be recognized, i.e., change the state of the channel, or be ignored. A match event recognized by the channel logic is called a **Match Recognition**. Match Recognitions can cause, also depending on Channel Mode and current state, the channel to request service, configuring a **Match Service Request**.

eTPU uses two kinds of comparator to assert a Match Event: an **Equal comparator**, in which both the Match Register and the value of the selected time base must match exactly, and a **Greater-Equal comparator**. The Greater-Equal comparator considers any time base value between the range [N : N+0x800000-1] as a valid match against the value of N in the Match Register, even when the value N+0x800000-1 wraps around the point of origin (0x0). See the following figure for an illustration of the matching values on a Greater-Equal comparator.

The second source of events for the eTPU channel is a **Transition** detected at the corresponding channel's input signal. Two distinct Transition detections can be programmed individually for each channel, allowing recognition of all possible combinations of edge detection. It is also possible to check the sampled state of an input

signal upon the occurrence of a Match: the sampling of the expected value is treated as a Transition, even if the input signal did not necessarily toggled at the time of the Match, or at any time at all.

Like Match Events, **Transitions Events** may or not be recognized by the channel logic. When they are, a **Transition Detection** occurs. As well as Match Recognitions, Transition Detections can issue a Channel Service Request, depending on Channel Mode and current state.

Transition Detections and Match Recognitions are sometimes simply called Transitions and Matches throughout this document, for short.

Input and output signals can be processed separately by the channel logic and microcode, and can also be combined such that Matches and Transitions are used to cause **output signal actions**. The output signals can also be directly controlled by microcode. Many event combinations are allowed for a channel, given the possibility of configuring pairs of matches and transitions for the dual action logic, where each event is able to block or enable the next one. There is a full set of Channel Modes described in Channel Modes, exploring all the capabilities mentioned here.

Each channel has its own set of registers and flags. They are selected, and made accessible to the Microengine, according to the value written into the microengine **CHAN Register** that points to the desired channel . Every time the CHAN register is written, even if with the same previous value, a channel is selected and its flags and registers are updated. For further detail, see the Channel Selection Register-CHAN topic in the General Channel Registers section .

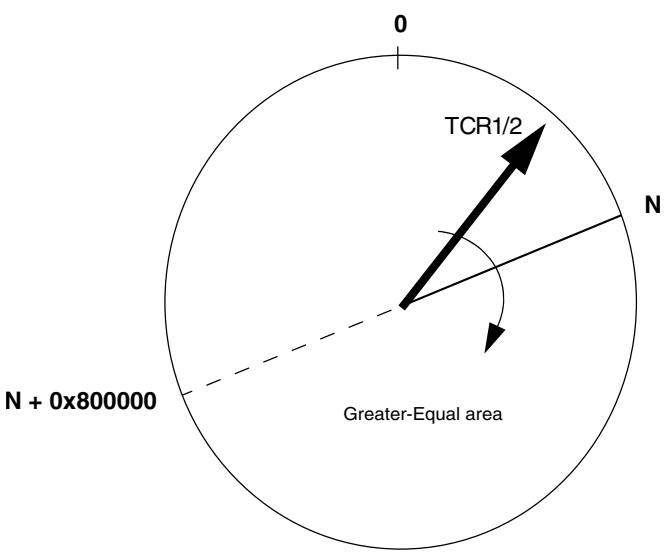


Figure 2-15. Greater-Equal Comparator

Beyond the request of services due to the signal and timing internal to each channel, one eTPU channel microcode can explicitly request service from another channel through the microengine **LINK Register**. A microcode write to the LINK Register asserts a service request to the channel whose number matches the contents of LINK. See Channel Link for a complete description of this mechanism.

These service requests originated in the eTPU Enhanced Channels (either time base match, input signal transition, or link service request) result in a call to the corresponding channel service routine, which is the sequence of microinstructions that is called a **Thread**. For further detail, see Functions and threads.

In addition to Event Logic, each Channel has an Output Buffer Enable signal, controlled by microcode, and an Enhanced Digital Filter, which eliminates spurious glitches on input pin signal. See the chip-specific eTPU information for how the Output Buffer Enable signal is controlled.

A high level diagram of Channel logic and registers is shown in the following figure.

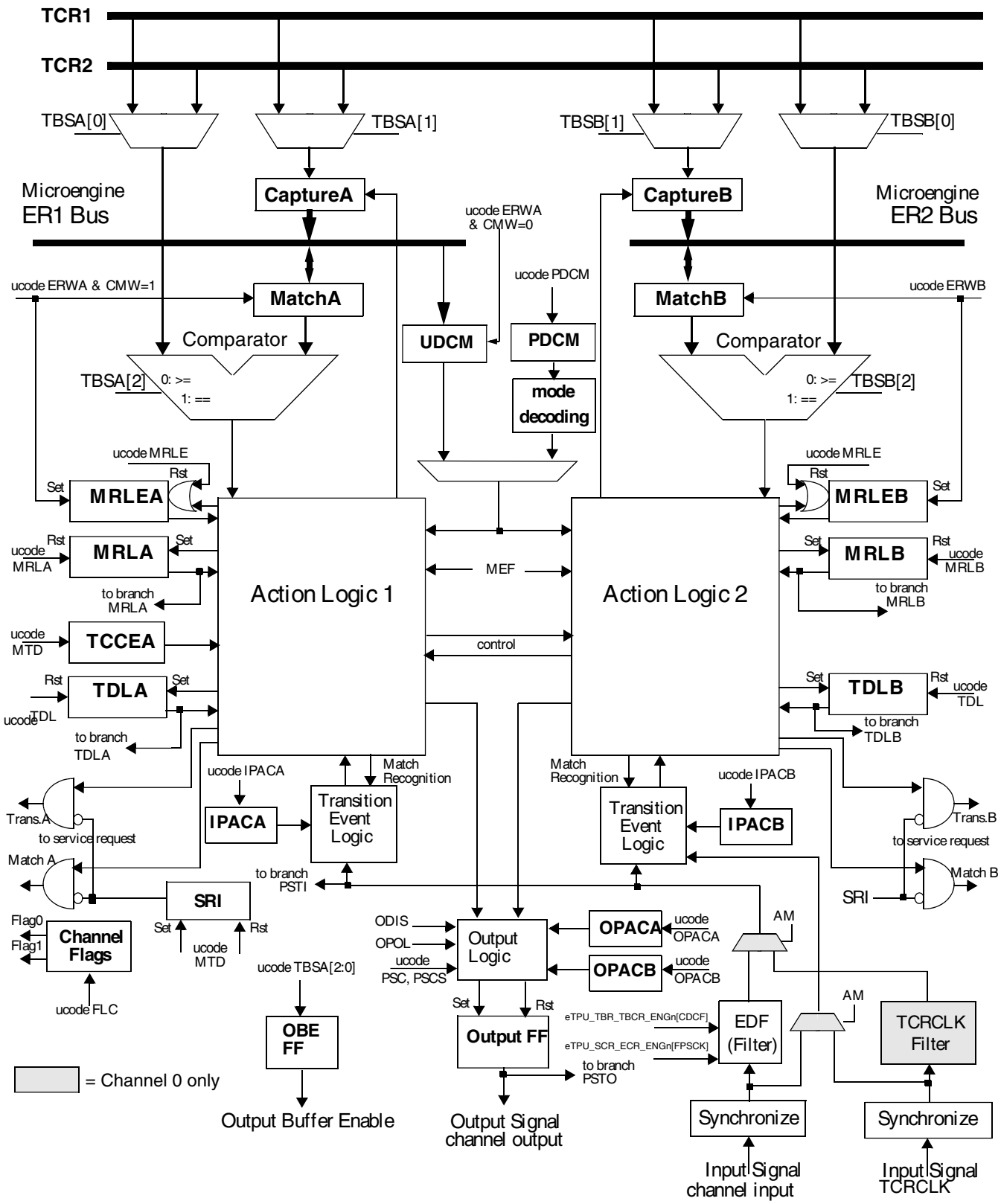


Figure 2-16. Channel logic block diagram

2.14.5.1 Channel registers and flags

Channel configuration and control registers can be divided in the following groups:

- **Host Configuration and Control registers**, which define channel Function and parameter allocation in SDM, input signal filtering, manage Host interrupts, and are used for Host Service Requests; they can only be accessed by Host, except for the Function Mode bits which can be also tested by microcode.
- **Event Registers**, which can only be accessed by eTPU Microengine, through dedicated Channel Control microinstruction operations (see Channel Control and Configuration Microoperations); these registers are directly used to implement channel functionality, and include channel event status latches which can be directly tested by Microengine branch instructions.
- **Pin Control registers**, which basically define pin state and transition polarity (but not input signal filtering); they are accessible only by dedicated Channel Control microinstruction operations.
- **Link registers**, which implement the channel link mechanism that allows one channel to request service to another one; they are accessible only by microinstruction operations.
- **General Channel registers**: CHAN, SRI, Flag0/1, PDCM, UDCM.

Most of those registers are channel exclusive, i.e., there is one copy of them for each channel. Microcode can access registers from only one channel at a time. The Channel Selection (CHAN) register (see Channel Selection Register - CHAN), accessible only by microcode, defines the channel whose registers are being accessed, with exception of link register and function mode. CHAN register assumes the value of the channel to be serviced at the beginning of TST.

The Service Request Inhibit - SRI - register controls the generation of Service Requests on matches and transitions, also affecting channel logic behavior. For a full description see SRI - Match/Transition Service Request Inhibit Latch. Flag0/1 are used to select channel service threads based on channel software state. See Flag1,Flag0 - Channel "state resolution" flags for more details.

Host Configuration and Control registers are described in Channel Configuration Control memory map and registers.

Time Base configuration is common to all channels, and described in Time bases. Time Base selection for matches and captures, however, is individual to each channel, and is part of the Event Registers.

Link registers are described in Channel Link.

The following sections describe the Event Registers and Pin Control registers.

2.14.5.1.1 ER - Event Registers

Each channel contains two identical Event Register sets, named ERA and ERB, corresponding to the two actions supported. Each Event Register set contains:

- a 24-bit Match register (Match A or Match B), which holds a match value. This value is compared against the selected match time base (TCR1 or TCR2).
- a 24-bit Capture register (CaptureA or CaptureB), which samples the selected capture time base (TCR1 or TCR2)
- a Time Base Selection register (TBSA or TBSB)
- a Match Recognition status flag (or latch) (MRLA or MRLB)
- a Match Recognition Enable latch (MRLEA or MRLEB)
- a Transition Detection flag (or latch) (TDLA or TDLB)
- a Transition Continuous Capture enable (TCCEA only)

ERA and ERB are associated with the first and second events in double action modes, not necessarily in that order. The order of Match events associated with ERA and ERB depends on the programmed channel mode, the MatchA and MatchB values, and the timebases selected by TBSA and TBSB. Similarly, the order of Transition events associated with ERA and ERB depends on the programmed channel mode, and the transition detection selected by IPACA and IPACB.

These registers are directly or indirectly accessed by the microcode. TBSA and TBSB registers are defined in TBSA and TBSB - Time Base Selection Registers. The other registers are explained in Match Recognition and Transition detection and Time Base Capture.

Access to the Event Registers is qualified by the channel currently selected by the microengine (i.e., the channel value currently in the CHAN register). During the channel transition period (automatic CHAN assignment), or whenever CHAN is written by microcode, Capture values of the new selected channel are sampled into Microengine registers ERTA and ERTB, therefore becoming visible to the microcode. At the same time, updated values of MRLA, MRLB, TDLA and TDLB are sampled into the branch logic, making the register values and the flags coherent with respect to each other and with the thread selected by the Scheduler. [20]

20. The thread selected is determined by the Entry Point which, in turn, is determined partially by the channel latches. See Entry Point address generation.

Note

The Function Mode bits are also sampled from the Host interface on Time Slot Transition, so that they remain constant to microengine even when Host changes them.

During service, the microcode can access updated values of the Event Registers of any channel by writing the channel number to CHAN. Writing CHAN with the same value (CHAN := CHAN) updates ERTA and ERTB with the new captured values, the branch logic with updated MRLA/B and TDLA/B flags. Writing CHAN with a different value does the same with the values from the newly selected channel.

Match values are also accessed through ERTA and ERTB Microengine registers, which are copied to/from the channel MatchA and MatchB registers by specific microinstruction operations.

Microcode writes to the flags and selections (MRLA/B, TDLA/B and TBSA/B) are immediately effective to the channel. The MRLA/B and TDLA/B branch conditions are also immediately reset when their correspondent flags are reset by microcode. Match registers are indirectly written by microcode through ERTA/B. MRLEA/B is unconditionally asserted when respective Match register is updated from ERTA/B, and its negation is immediate.

The following table summarizes Event Registers accesses.

Table 2-29. Event Registers microcode accesses

Register	Access type	Sampled from channel	Update to channel	Microcode fields[1]	Reset value[2]
CaptureA, CaptureB	read through ERTA/B	to ERTA/B on CHAN assignment	no	T2ABD	n.a.
MatchA, MatchB	read and write through ERTA/B	to ERTA/B by microcode	from ERTA/B by microcode	ERWA, CMW, ERWB, T4ABS	n.a.
MRLEA, MRLEB	write to 0 (negate) directly; write to 1 (assert) upon MatchA/B update from ERTA/B	no	immediate	MRLE, ERWA, ERWB	0, 0
TBSA, TBSB	write only	no	immediate	TBSA, TBSB	000, 000
MRLA, MRLB	flag test on branch, write to 0 (negate) only	on CHAN assignment	immediate	BCC (test) MRLA, MRLB (reset)	0, 0
TDLA, TDLB	flag test on branch, write to 0 (negate) only	on CHAN assignment	immediate	BCC (test) TDL (reset)	0, 0
TCCEA	write only	no	immediate	MTD	0

1. See Microinstruction Set.
2. n.a. means that value of the register is undetermined after reset.

2.14.5.1.1.1 MatchA and MatchB Registers

MatchA and MatchB registers hold a match value to be compared against the selected channel time base. A match value can only be written into the Match register by microcode, through ERTA/B microengine registers (see Write Channel Match and UDCM Registers). Microcode can also read the Match register as a special T4ABS source operation, when T4ABS=0101, and the source for T4ABS is selected from the second register set. In this operation, MatchA/B registers are copied into ERTA/B registers (see Selecting sources and destination). For more information on time base matches, see Match Recognition.

2.14.5.1.1.2 CaptureA and CaptureB Registers

CaptureA and CaptureB registers capture the selected channel time base. CaptureA/B registers cannot be directly written or read by microcode. During the Time Slot Transition (TST) or during CHAN assignment, CaptureA/B registers are copied into ERTA/B microengine registers. For more information, see Transition detection and Time Base Capture.

2.14.5.1.1.3 TBSA and TBSB - Time Base Selection Registers

TBSA/B are 3-bit registers which have the following effect on channel configuration:

- Selection of the timebase (TCR1 or TCR2) to be compared against the match values in MatchA and/or MatchB registers.
- Selection of the timebase (TCR1 or TCR2) to be captured in the CaptureA and/or CaptureB registers by a match or transition detection event.
- Selection of comparator mode to be used with MatchA and MatchB registers: equal-only or greater-equal.

After reset TBSA/B are 000. The following table shows values of TBSA and TBSB for configuration selection. Note that the time base selection for capture is independent of the time base selected for matches.

Table 2-30. TBSA/B Programming States

TBS bit	2	1	0
bit value	Comparator selection	Capture time base selection	Match time base selection
0	greater or equal	TCR1	TCR1
1	equal-only	TCR2	TCR2

TBSA/B are written through the microcode fields TBSA/B (see Comparator and Time Base Selection). Note that microcode field TBSA is also used to control the OBE pin control register (see Pin Control Registers), which is separate from the TBSA register.

2.14.5.1.1.4 MRLA/B - Match Recognition Latches

See MRLA/B - Match Recognition Latches.

2.14.5.1.1.5 MRLEA/B - Match Recognition Latch Enable

See MRLEA/B - Match Recognition Latch Enable.

2.14.5.1.1.6 TDLA/B - Transition Detection Latch

See TDLA/B - Transition Detect Latches.

2.14.5.1.1.7 TCCEA - Transition Continuous Capture Enable

See TCCEA - Transition Continuous Capture Enable.

2.14.5.1.2 Pin Control Registers

Pin Control Registers are replicated one per channel, accessed only by microcode and qualified by the CHAN register in the same way as Event Registers.

The following table lists Pin Control Registers, explained in following subsections, and their accesses.

Table 2-31. Pin Control Registers microcode accesses

Register	Access type	Sampled from channel	Update to channel	Microcode fields[1]	Reset value
IPACA, IPACB	write only	no	immediate	IPACA, IPACB	000,000
OPACA, OPACB	write only	no	immediate	OPACA, OPACB	000,000
PSTI	flag test on branch	no	no	BCC	0

Table continues on the next page...

Table 2-31. Pin Control Registers microcode accesses (continued)

Register	Access type	Sampled from channel	Update to channel	Microcode fields[1]	Reset value
PSTO	flag test on branch, write	no	immediate	BCC (test) PSC, PSCS (set)	0
OBE[2]	write only	no	immediate	TBSA values 1000,1001	0 (negated)
PSS	flag test on branch	on CHAN assignment	no	BCC	n.a.[3]
PRSS	flag test on branch	on CHAN assignment	no	BCC	n.a.[4]

1. See Microinstruction Set.
2. Although this register exists for code compatibility reasons, OBE functionality is not available on this chip.
3. PSS is PSTI or PSTO sampled on CHAN assignments and at thread start.
4. PRSS is PSTI sampled on channel service request.

2.14.5.1.2.1 IPACA/B and OPACA/B - Input and Output Pin Action Control Registers

These registers determine the transition detections and output pin actions which takes place due to match or transition events. Each field is three bit wide. After reset, the IPACA/B and OPACA/B registers are set to 000. IPACA and IPACB registers are mutually independent and have identical encoding, and so do OPACA and OPACB. The table below shows IPAC and OPAC encoding. Note that transition detections are independent from the output actions, but some options of output actions are triggered by transition detections. Output actions can also be triggered by Matches. For a detailed definition of Transition Detections, see Transition detection and Time Base Capture.

IPACA/B define the Transition events treated by channel logic. Although the name "transition" is generically used for the transition detections, IPAC options 100 and 101 do not really detect transitions: they actually sample the state of the input signal at the occurrence of the corresponding Match (Match A used for IPACA, Match B used for IPACB).

Table 2-32. IPACA/B and OPACA/B Encoding

Value	IPAC meaning	OPAC meaning
000	Do not detect transitions	Do not change output signal
001	Detect rising edge only	Match[1] sets output signal high
010	Detect falling edge only	Match[1] sets output signal low
011	Detect either edge	Match[1] toggles output signal
100	Detect input signal = 0 on Match[1]	Transition sets output signal low
101	Detect input signal = 1 on Match[1]	Transition detection sets output signal high

Table continues on the next page...

Table 2-32. IPACA/B and OPACA/B Encoding (continued)

Value	IPAC meaning	OPAC meaning
110	reserved	Transition detection toggles output signal
111	n.a.[2]	n.a.[2]

1. Match A is used for IPACA/OPACA, and Match B for IPACB/OPACB.
2. On the microinstruction fields IPACA/B and OPACA/B this value is neutral, meaning that IPAC/OPAC register values are not changed.

2.14.5.1.2.2 Output Pin Control Logic and Pin State Output Register - PSTO

The output signal control logic uses OPACA/B, the Pre-defined Channel Mode (PDCM) and the User Defined Channel Mode (UDCM), and the microcode Pin State Control (PSC and PSCS) fields. It is responsible for setting the Pin State Output (PSTO) register to the specified logic value required by microcode, by input events, or by Match A and/or Match B events. The PSTO register stores the driven pin state determined by the Pin Control logic. The Output Buffer Enable signal, if used at MCU integration, must be set by microcode (using TBSA field) to make the pad propagate the PSTO register output to the actual pin.

PSTO register output also goes to the microengine branch logic, enabling branching on the driven pin state. PSTO is set to 0 on reset.

The PSC and PSCS microcode fields are used for setting the PSTO register to a fixed value, or to the value specified by the OPACA or OPACB microcode field, as shown in the following table.

Table 2-33. PSC and PSCS encoding

PSC	Output pin action
00	Force pin state according to OPACA (PSCS=0) or OPACB (PSCS=1).
01	Force pin high.
10	Force pin low.
11	Do not change pin state.

See Channel Control and Configuration Microoperations for details.

On occurrence of match recognitions or transition detections, the pin control logic sets PSTO value according to the event number (Match A/Transition A or Match B/Transition B) and the contents of OPACA/IPACA or OPACB/IPACB registers. There are cases in which two match or transition events may occur at the same time, each of them trying to force a different pin action. The resolution of the selected match event which sets the

value depends on the Pre-defined Channel Mode (PDCM) register and the User Defined Channel Mode (UDCM). For details refer to Match/Transition Pin Action Conflict Resolution.

2.14.5.1.2.3 PSTI and PSS - Pin State Input and Pin Sampled State Registers

During the time slot transition period, or whenever the CHAN register is written by microcode, the filtered[21] input signal PSTI or output signal PSTO (selected by the ETPD field in eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2)) is sampled into the branch logic as the PSS flag (see Table 2-100). The microcode can then branch on either the currently driven (PSTO) or input (PSTI) pin state, or on sampled pin state (PSS, which is stable as long as CHAN does not change).

Note

If a transition occurs simultaneously (after the filter) with the CHAN assignment, PSS samples the new pin value. Therefore, if TDLA/B is cleared simultaneously with the assignment CHAN := CHAN, the occurrence of a transition at this very moment can still be tested with PSS.

2.14.5.1.2.4 PRSS - Pin Request Service Sample

Channel logic can, depending on its state and programmed mode, request service to the eTPU microengine (see Functions and threads). When the channel logic issues a service request, the filtered input signal PSTI is sampled into an internal channel flag PRSS. There is one such PRSS flag for each channel (see Pin Control registers). Channel PRSS keeps its value until all its service request sources are cleared and until a new channel service request rises.

Note

Only channel service requests update PRSS: it is updated neither by host service requests (HSRs) nor by link service requests. Moreover, a channel service request updates PRSS even if the channel is disabled (its CPR set to 0, see (xref to eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2)).

21. The filter can be bypassed.

The channel PRSS flag is sampled into the branch logic as the PRSS flag (see Table 2-100) during the time slot transition period, or whenever the CHAN register is written by microcode.

2.14.5.1.3 General Channel registers

These registers control other aspects of channel logic. Except for CHAN, they are unique per channel. The following table summarizes the registers and access options.

Table 2-34. General Channel registers microcode access

Register	Access type	Sampled from channel	Update to channel	Microcode fields[1]	Reset value
CHAN	read/write	n.a.[2]	n.a.[2]	T4ABS, T4BBS, T2ABD	defaults to serviced channel at thread start
PDCM	write only	no	immediate	PDCM	1100 (sm_st)
UDCM	write only	no	from ERTA by microcode	CMW, ERWA	parameter value defined at integration
SRI	write only	no	immediate	MTD	1
Flag1, Flag0	branch flag test, write	no	immediate	BCC,FLC	0, 0

1. See Microinstruction Set.
2. CHAN is common to all channels in the Engine.

2.14.5.1.3.1 Channel Selection Register - CHAN

CHAN is the register that holds the number of the channel that qualifies the context of most Channel Registers, including Pin Control and ER accesses, and is common to all channels in a same Engine.

When a thread starts to be executed, the contents of CHAN register is automatically updated on Time Slot transition to the number of the channel to be serviced. Serviced channel is constant during channel servicing, but the selected channel can be changed any time by microengine writing into CHAN register.

Some microinstructions are affected by the serviced channel instead of CHAN. These are:

- Conditional branch using LSR (see LINK Register) or Function Mode (see eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C*n*SCR_ENG1) and eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C*n*SCR_ENG2)).
- Negate channel flag LSR, Interrupt CPU and Data Transfer Request (see Channel Interrupt and Data Transfer Requests).

When CHAN register is written, accesses are qualified by the new CHAN register value from the instruction following CHAN assignment on, except CaptureA/B sampling into ERTA/B and Match register writing from ERTA/B (see CHAN assignment, Read Match and ERWA/B).

Writing CHAN (including with the same value, CHAN:= CHAN) updates ERTA and ERTB with the new captured values, the branch logic with updated MRLA/B and TDLA/B flags.

The following table shows the commands, flags and registers selected by the CHAN register value.

Table 2-35. CHAN-selected features

Feature used	Selected by CHAN
Channel-relative SDM access	YES
Branch using PSS, PRSS, PSTI and PSTO channel flags	YES
Branch using MRLA/B, TDLA/B, Flag0/1[1]	YES
Branch on all other conditions[2]	no
ERTA/B Value	YES
Configure (selected) channel	YES
Channel commands applied to: MRLA/B, TDLA/B, TBSA/B, IPACA/B,OPACA/B, PSC, PSCS, OBE, PDCM, MRLE, Flag0/1	YES
Channel command: set/reset SRI	YES
Channel command: write to Match registers (ERWA/B)	YES[3]
Channel command: read Match registers into ERTA/B	YES
Clear LSR	no
Channel interrupts and data transfer requests (CIRC)	optionally

1. In TPU, these conditions retained the old values.
2. See Branch Operations
3. If write Match (ERWA/B) occurs at the same time of a CHAN assignment, the channel which is target of the write is the one selected by the new CHAN value. See CHAN assignment, Read Match and ERWA/B.

2.14.5.1.3.2 PDCM - Predefined Channel Mode

PDCM determines the channel mode assigned to the channel. Channel mode defines much of the channel logic behavior, specially how matches blocks and enables transitions and vice-versa, as well as occurrence of time base captures and service requests based on matches and transitions. For a complete description see Channel Modes.

PDCM is a 4-bit register set by the microcode field of the same name (see Predefined Channel Modes), and cannot be read or tested in branch instructions. The table below relates the PDCM value with channel modes. The second column specifies the mnemonic used to reference the mode, introduced in Channel Modes Overview . There is one PDCM for each channel, initialized with 1100 on reset.

PDCM is also used to select the User Programmable Channel Mode. If this selection is made, the channel behavior is defined by the settings of the UDCM register (see UDCM - User Defined Channel Mode).

Table 2-36. PDCM encoding

PDCM	Channel mode
0000	em_b_st
0001	em_b_dt
0010	em_nb_st
0011	em_nb_dt
0100	m2_st
0101	m2_dt
0110	bm_st
0111	bm_dt
1000	m2_o_st
1001	m2_o_dt
1010	User Defined Channel Mode
1011	reserved
1100	sm_st[1]
1101	sm_dt
1110	sm_st_e
1111	n.a.[2]

1. This is the reset value, also compatible with TPU channel behavior.
2. This value is used as a neutral (do not change) value in the PDCM microinstruction field.

2.14.5.1.3.3 UDCM - User Defined Channel Mode

UDCM holds the control signals that define the channel logic behavior in terms of Match and Transition blocking and enabling, captures and service requests triggered by events. The register layout is defined in the figure below. The individual fields are defined in Channel Mode Logic and Event Flags. There is one UDCM register for each channel, which can be independently programmed.

UDCM can only be written into the Match register by microcode, through ERTA/B microengine registers (see Write Channel Match and UDCM Registers).

13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSR		MCAP	M1ET	M1EM2	M1BM2	M2BM1	M2BT	T1BM1	T2BM1	TBM2	T1ET2	TSR	TCAP

Figure 2-17. UDCM Register

2.14.5.1.3.4 SRI - Match/Transition Service Request Inhibit Latch

SRI blocks channel service requests due to the assertion of MRLA/B and/or TDLA/B. SRI does not affect recognition of Link Service Requests or Host Service Requests, neither MRLA/B or TDLA/B microcode branch tests nor Entry Table selection[22] . SRI is asserted during reset and is controlled by microcode field MTD.

To unburden the microengine, SRI asserted configures a channel "dumb" regarding the servicing of match and capture channel service requests. Even with SRI=1, TDLA/B and MRLA/B can still be asserted, and the level specified by the OPAC (Output Pin Action Control) registers will be output to the pin.

2.14.5.1.3.5 Flag1,Flag0 - Channel "state resolution" flags

Each channel has a pair of flags, simply called Flag0 and Flag1, that can be set/reset by microcode through microinstruction field FLC. FLC sets/resets Flag0/1 of the channel selected by CHAN. These flags can be tested by microcode, and are also used to resolve the microcode entry point for the channel service (see Entry Points"). Flag0 and Flag1 are, so, typically used for fast state resolution. FLC microinstruction field also allows Flag1,Flag0 to be copied from selected bits of P register high byte, which is also meant to be used to hold application state. Flag0 and Flag1 are both zero out of reset.

22. In TPU, SRI also blocked TDL and MDL branches and enabled any transition to capture time base.

2.14.5.2 Match Recognition

The match operation is performed every microcycle by comparing the channel MatchA and MatchB registers against the value of the TCR bus specified for each match. TCR1 or TCR2 bus is selected according to TBSA and TBSB fields. Both results have effect on the next clock cycle (see Microcycle and I/O timing).

A Match A/B event is qualified by a set of match enabling conditions to the Match Recognition Registers MRLA/B. To recognize the match, the following match enabling conditions are required:

- For IPACA/B=0xx, Match Enable Flag (MEF), qualified by the channel currently being serviced must be asserted. For IPACA/B =1xx, Match A/B is always enabled (even during Time Slot Transition - TST), regardless the state of the Match Enable Flag (MEF). See MEF - Match Enable Flag for the conditions of MEF assertion.
- Match Recognition Latch Enable 1/2 (MRLEA/B) is asserted. A match event recognition may only occur if its corresponding MRLEA/B bit is set, which only happens upon a write to a channel match register by the microcode, copied from ERTA/B. MRLEA/B is negated when the respective match occurs or, in some double match channel modes, when a match for the other Match register occurs. It ensures that the greater-equal comparison will not cause additional matches[23] .
- In selected modes (see Channel Modes), the particular conditions of MRL and TDL flags of the other event, i.e:
  - MRLA, TDLA enable or block MRLB;
  - MRLB, TDLB enable or block MRLA.
- The respective MRL is negated.
- In selected modes (see Channel Modes), the state of its respective TDL flag.

If the Match A and/or Match B conditions are met, the channel immediately forces the pin state if specified by the appropriate OPACA/B registers (Output Pin Action Control 1/2) and, in some cases, by IPACA/B registers. Refer to IPACA/B and OPACA/B - Input and Output Pin Action Control Registers.

If both Match A and Match B events occur at the same time, with conflicting pin actions, the priority over the pin action is mode dependent. For further details on pin action resolution refer to Match/Transition Pin Action Conflict Resolution.

23. Microcode can also negate MRLEA/B.

2.14.5.2.1 MRLA/B - Match Recognition Latches

MRLA/B indicate the recognition of a match event detected by the comparator. They can be asserted either on T2 or T4 (see Microcycle and I/O timing). Assertion of MRLA/B issues a match Service Request in specific channel modes, depending on previous events and state of SRI. After reset MRLA and MRLB are both negated.

When MRLA or MRLB is asserted, it may change the output signal level according to the Input and Output Pin Action Control registers (refer to IPACA/B and OPACA/B - Input and Output Pin Action Control Registers). Assertion of MRLA/B causes a capture of one or two time bases, according to the selected mode capturing scheme (see Transition detection and Time Base Capture)

A match recognition is self-blocking, regardless of Channel Mode: once MRLA (MRLB) has been asserted, it negates its associated MRLEA (MRLEB) register, preventing future match recognitions, until the associated match register is rewritten by microcode. The microcode has to enable new matches by updating the new match value in the MatchA (MatchB) register[24] . In addition, assertion of MRLA/B can block its twin MRLB/A, depending on the channel mode. In some double match blocking channel modes, Match A/B event blocks the occurrence of Match B/A in a "first win" scheme.

It is the transition from 0 to 1 in MRL that causes the Match actions, except the service request: apart from MRLEA/B negation(s) and service request, no action due to a Match occurs if MRL was already set to 1, even if the other MRL assert conditions are satisfied. However, if a Match and a microoperation negating its corresponding MRL occur at the same time, MRL negation by microcode overrides its assertion, but any dependable captures and pin action occurs anyway (if MRL was already negated before), and also the negation of MRLE(s) (the respective one and, in some channel modes, the other, regardless of MRL state before). Note that MRLE must have been set before (by writing a new Match value).

2.14.5.2.2 MEF - Match Enable Flag

MEF is a one-bit latch that is unique for all channels in an Engine.

MEF can selectively enable assertion of MRLA/B, depending on the IPACA/B field. For IPACA/B=0xx, MEF=1 enables assertion of MRLA/B for the scheduled channel during service. For IPACA/B=1xx, Match A/B is always enabled, regardless the state of the MEF, but it still depends on the other Match recognition conditions. Matches of channels not being serviced are never disabled by MEF.

24. Before that, microcode should also negate MRLA (MRLB), otherwise an old match may be recognized by the scheduler and serviced as a new one

MEF is not accessible by Microengine or Host. MEF is negated for two microcycles in the beginning of the time slot transition period. After two microcycles (plus wait-states) into TST, the ME bit in the entry point is copied to MEF to allow selective enabling of MRL for each thread (refer to Time Slot Transition). MEF is asserted unconditionally soon after a thread ends.

If a channel service needs to postpone a programmed match, MEF assures that microcode wins the race against match event after time slot transition (only for IPAC=0xx).

Note that a match event may be lost during the periods when MEF is negated only if:

- the match comparator is configured for "equal-only" behavior, and
- IPACA/B=0xx, and
- TCR increments at the rate of eTPU clock divided by 2 or faster.

When the comparator is configured as "greater-equal", the match event that occurred when MEF was negated may be recognized after MEF is asserted again, due to the "greater than" condition.

2.14.5.2.3 MRLEA/B - Match Recognition Latch Enable

MRLEA/B is negated upon the assertion of its respective MRLA/B. In blocking match channel modes it may also be negated together with the assertion of the twin MRLB/A. The MRLEA/B bits ensure that data captured due to the first match event will not be overwritten when MRLA/B is negated: due to greater-equal comparison, the match condition continues to be true, but should not cause another capture event.

In addition to negation by local match event, the microcode can negate both MRLEA and MRLEB, to block pending matches, and also MRLA/B, individually. This action will prevent future match events from the selected channel.

Writing the MatchA/B registers by microcode to schedule the next match values sets MRLEA and/or MRLEB and enables new matches. This setting overrides the MRLE negation conditions due to channel logic or microcode (see Channel Modes). By combining write to Match A/B with MRLEA/B negation microinstructions, the microcode can negate one MRLE while asserting the other.

Note

If the MRLE negation conditions continue after writing MatchA/B registers, the respective MRLE does not keep asserted. For instance, if MRL=1 and a match is programmed for a time value in the past during a thread with ME=1, then

MRLE will be cleared soon after MatchA/B is written, even though a match does not occur (because MRL was already asserted, neither captures nor pin toggles occur).

When the match register is updated (with MRLE already asserted before) and field MRLA/B = 1 (no clear, see Clear Transition/Match Event Registers) and MRLA/B flag is zero, the eTPU behaves exactly as the TPU, that is: a match that comes concurrently with the rewrite of the match register, matching the old value, sets the MRL, as if the setting of the MRLE due to match register write had precedence over its clear by the match at that moment. After this simultaneous operation, the MRLE value stays at 1, and the captured time base value, if any, reflects the match value.

When the match register is updated (with MRLE already asserted before) and field MRLA/B = 0 (clear MRL, see Clear Transition/Match Event Registers) and MRLA/B flag is zero, the match captures will occur, the MRLA/B flag will keep negated, and MRLE will stay asserted. Special care must be taken when a match is reprogrammed on a channel running at T2/T4 timing, please see the details in the “T2/T4 Channel Timing” section.

2.14.5.3 Transition detection and Time Base Capture

Time Base Capture(s) occur when the value of a specified TCR is sampled into the CaptureA and/or CaptureB register. TBSA[1] and TBSB[1] select which TCR will be captured in CaptureA and CaptureB, respectively.

A capture event may occur due to either of the following events:

- The assertion condition of Match Recognition Latch (MRL), even if MRL is simultaneously negated by microcode
- The assertion condition of Transition Detection Latch (TDL), even if TDL is simultaneously negated by microcode.
- Any Transition Event specified by IPACA if both the Transition Detection Latch TDLA and Transition Continuous Capture Enable TCCEA are asserted.

A capture event occurs together with the assertion of MRL or TDL either on T2 or T4 positive edges, and captures the time-base value that caused the match, even if TCR1/2 increments concurrently with the assertion (see Microcycle and I/O timing)[25] . MRLA/B and TDLA may, depending on the channel mode, inhibit the capture of the second event's TCR into CaptureA/B. As a general rule, values captured by signal transitions are not overwritten by values captured by match events.

25. In TPU3, when TCR1 was counting at maximum rate of eTPU clock divided by 2, the next value was captured.

When the enable bit TCCEA is asserted, captures due to Transition Events also occur after TDLA is asserted. Those captures happen on transition events specified by IPACA, and the TCR value is saved into CaptureA register only.

The capturing scheme is defined by the Channel Mode programmed at register PDCM, or at register UDCM when User Defined Channel Mode is selected . For more information on mode-dependent capture schemes refer to Channel Modes.

2.14.5.3.1 TDLA/B - Transition Detect Latches

TDLA/B indicate detection of specific transition occurrences on a channel input signal. TDLA and TDLB assertion causes service request in single and double transition predefined modes, respectively. TDLB assertion does not cause Service Request in single transition predefined modes, and TDLA assertion does not cause Service request in double transition predefined modes. In single transition channel predefined modes TDLB can be asserted on the second transition, but does not generate Service Request. Yet on predefined modes, TDLB assertion is enabled only if TDLA is asserted to detect an ordered input signal double transition. All the restrictions above, however, may be overridden by using the User Defined Channel Mode. The IPACA and IPACB registers indicate the programmed edges of the first and second detected transition, respectively.

The sampling of a determined value (0 or 1) on the input signal due the occurrence of a Match is also treated as a "transition", depending on IPACA/B programming (see IPACA/B and OPACA/B - Input and Output Pin Action Control Registers). When using a channel mode where the transition1 is initially blocked and IPACA is programmed to detect such "transitions", the occurrence of a Match A only unblocks the transition after the sampling. It means that the transition on the first Match (IPACA configurations 100 and 101) is not effective on predefined modes where Transition A is enabled by Match A (m2_st, m2_dt, m2_o_st and m2_o_dt) or user-defined modes with UDCM bit M1ET asserted . A Transition A can still happen after the Match A, however, if MatchA register is reprogrammed without clearing MRLA.

TDLA/B assertion conditions initiates a capture event of one or both selected TCR buses. TDLA or TDLB transition event generates a Service Request, depending on channel mode, previous events and the state of SRI. For more information on service request scheme refer to Entry Point address generation" and Channel Modes.

Assertion of TDLA/B occurs on either T2 or T4 positive edges. The capture event occurs on the same clock, and captures the time base value present when TDLA/B was asserted[1] . TDLA and TDLB are negated during reset and may also be negated independently by microcode. TDLA/B is reset by no way other than reset and microcode.

It is the transition from 0 to 1 in TDL that causes the Transition actions: even if TDL assert conditions are satisfied, no action due to a Transition occurs if TDL was already set to 1. However, if a Transition and a microoperation negating TDLs occur at the same time and TDL was already negated, TDL negation by microcode overrides its assertion, but any dependable captures and pin action occurs anyway.

2.14.5.3.2 TCCEA - Transition Continuous Capture Enable

TCCEA enables capture from transitions after the TDLA flag is set. TCCEA is negated on reset, so that a capture occurs only when TDLA asserts. TCCEA can be set and reset by microcode only, through the instruction field MTD (see Disable Match and Transition Service Requests). It can only be set together with inhibiting of the channel service requests (SRI = 1)[26].

When TCCEA is asserted, the transition events specified by IPACA that occur after TDLA is set also cause captures, into the CaptureA register only. However, output actions related to transition events are still blocked by TDLA.

2.14.5.4 Channel Modes

The Enhanced Channels support various modes of operation combining Match A/B recognition and transition detection events which set MRLA/B and TDLA/B. The channel mode is individually set for each channel by eTPU microcode, through the PDCM register (see PDCM - Predefined Channel Mode). The PDCM register selects among a set of 13 predefined channel modes, and also a user-defined channel mode.

The order in which events occur, combined with assigned channel mode, establish which following event detections are inhibited and/or enabled, as well as the actions taken: Time Base capture, flag setting (MRLA/B, TDLA/B), match disabling (MRLEA/B), output signal transition, and Service Request. Those channel mode characteristics are fixed in the predefined modes, but can be individually programmed in the user-defined channel mode.

A generic description of channel modes from the usage point of view can be found in Channel Modes Overview . Each mode is named with a mnemonic acronym for terse reference. The individual programmed attributes of the user-programmable channel mode are also described.

26. TCCEA provides compatibility with TPU when service request is inhibited.

The modes are used differently for input and output signals, as explained in Predefined Channel Modes on Input Signal Processing and Channel Modes on Output Signal Generation. Modes also allow combining input processing and output generation in a single channel, as exemplified in Combining Input and Output Signals.

Channel Mode Logic and Event Flags shows a structural definition of channel logic and its relation to channel modes. A dynamic, event-oriented view of each channel mode can be found in Channel Mode Summary.

2.14.5.4.1 Channel Mode Logic and Event Flags

Figure 2-18 shows a more detailed diagram of channel mode logic. The logic shown here is not necessarily identical to the actual channel logic implementation, but is equivalent with respect to conditions for event blocking, enabling, capture, and service requests.

Signals MSR, TSR, MCAP, TCAP, M1ET, M1EM2, M1BM2, M2BM1, M2BT, T1BM1, T2BM1, TBM2 and T1ET2 are decoded from programmed channel mode PDCM in predefined modes, and come directly from the UDCM register when user-defined mode is selected:

- TSR (1 bit) defines Service Requests issued by Transitions, as shown in Table 2-38.
- MSR (2 bits) defines Service Requests issued by Matches, as shown in Table 2-37.
- TCAP (1 bit) defines time base captures caused by Transitions, as shown in Table 2-38
- MCAP (1bit) defines time base captures caused by Matches, as shown in Table 2-39.
- M1ET, M1EM2, M1BM2, M2BM1, M2BT, T1BM1, T2BM1, TBM2, T1ET2 (1 bit each) define Match and Transition reciprocal blocking and enabling, as well as Transition ordering, as shown in Table 2-40 and Table 2-41.

Table 2-42 shows the decoded values of those control signals for each predefined channel mode. The first column shows the mnemonic reference for the predefined channel modes described in the following sections.

Changing PDCM or the UDCM when user mode is selected may set or reset any of the channel flags, or issue captures and service requests, so it is advisable to switch channel modes only in a quiescent channel state, with channel flags MRLEA/B, TDLA/B, MRLA/B cleared. Furthermore, an event flag asserted in one mode may keep asserted after the mode programming change, even if the flag is impossible to be set in the new mode.

Table 2-37. MSR signals - Match Service Requests

value	MSR
00	issue no Service Requests on Matches
01	issue Service Request on Match B only
10	issue Service Request on 2nd[1] Match
11	issue Service Request on both Matches

1. 2nd Match means the Match that happens after the 1st Match in time, either Match A or Match B.

Table 2-38. TCAP and TSR signals - Transition Captures and Service Requests

value	TCAP	TSR
0	1st[1] Transition captures Time Bases corresponding to Transition A and Transition B[2] Transition A captures corresponding Time Base if it is the second transition; Transition B captures corresponding Time Base if it is the second transition.	issue Service Request on the 1st[1] Transition
1	Transition A captures corresponding Time Base. Transition B captures corresponding Time Base.	issue Service Request on the 2nd[3] Transition

1. 1st Transition means the Transition that happens first in time, either Transition A or Transition B.
2. Match capture(s) never overwrites a Transition capture. Transition captures can always overwrite a Match capture.
3. 2nd Transition means the Transition that happens second in time, either Transition A or Transition B

Table 2-39. MCAP signal - Match Capture

value	MCAP
0	Match A captures corresponding Time Base; Match B captures corresponding Time Base
1	either Match captures Time Bases corresponding to Match A and Match B[1]

1. Match capture(s) never overwrites a Transition capture. Transition captures can always overwrite a Match capture.

Table 2-40. TBM2 signal - Transition Blocks Match B

value	TBM2
0	Transition A Blocks Match B
1	Transition B Blocks Match B

Table 2-41. M1ET, M1EM2, M1BM2, M2BM1, M2BT signals

signal	active value meaning
M1ET (Match A Enable Transitions)	Transitions are initially blocked, and Match A enables Transitions
M1EM2	Match B is initially blocked[1],

Table continues on the next page...

Table 2-41. M1ET, M1EM2, M1BM2, M2BM1, M2BT signals (continued)

signal	active value meaning
(Match A Enables Match B)	and Match A enables Match B
M1BM2[2] (Match A Blocks Match B)	Match B is initially enabled[1], and Match A blocks Match B[3]
M2BM1[2] (Match B Blocks Match A)	Match A is initially enabled,and Match B blocks Match A[3]
M2BT (Match B Blocks Transitions)	Match B blocks Transitions
T1BM1 (Transition A Blocks Match A)	Transition A blocks Match A
T2BM1 (Transition B Blocks Match A)	Transition B blocks Match A
T1ET2 (Transition A Enables Transition B)	Transition B is initially blocked, and Transition A enables Transition B

1. The initial condition of M1EM2 prevails over M1BM2, while M1BM2 blocking prevails over M1EM2 enabling, so that Match B stays always blocked when both M1BM2 and M1EM2 are active. This combination is used in single-match modes (sm_*).
2. Blocking of one Match by the other is done through MRLEs.
3. Matches always block themselves by resetting their own MRLEs (Match A always blocks Match A, Match B always blocks Match B)

Table 2-42. Predefined Channel Mode control signals decoding

Predefined Mode	MSR	MCAP	M1ET	M1EM2	M1BM2	M2BM1	M2BT	T1BM1	T2BM1	TBM2[1]	T1ET2	TSR[1]	TCAP[1]
em_nb_st	11	0	off	off	off	off	off	On	off	0	On	0	0
em_nb_dt	11	0	off	off	off	off	off	On	off	1	On	1	1
em_b_st	11	1	off	off	On	On	off	On	off	0	On	0	0
em_b_dt	11	1	off	off	On	On	off	On	off	1	On	1	1
bm_st	10	0	off	off	off	off	off	On	off	0	On	0	0
bm_dt[2]	10	0	off	off	off	off	off	off	On	1	On	1	1
m2_st	01	0	On	off	off	On	off	On	off	0	On	0	0
m2_dt	01	0	On	off	off	On	off	On	off	1	On	1	1
m2_o_st	01	0	On	On	off	off	On	On	off	0	On	0	0
m2_o_dt	01	0	On	On	off	off	On	On	off	1	On	1	1
sm_st	11	1	off	On	On	On	off	On	off	0	On	0	0
sm_dt[2]	11	1	off	On	On	On	off	off	On	1	On	1	1
sm_st_e[3]	11	0	off	On	On	On	off	On	off	0	On	0	0

1. signals TSR, TCAP and TBM2 replace the signal DTM used in previous eTPU versions.
2. bm_dt and sm_dt are exceptions in the match blocking logic by transitions. See Both Match Request Modes (bm_st, bm_dt) and Single Match Modes (sm_st, sm_dt).
3. sm_st_e is an exception in the capture scheme. See Single Match Enhanced Mode (sm_st_e).

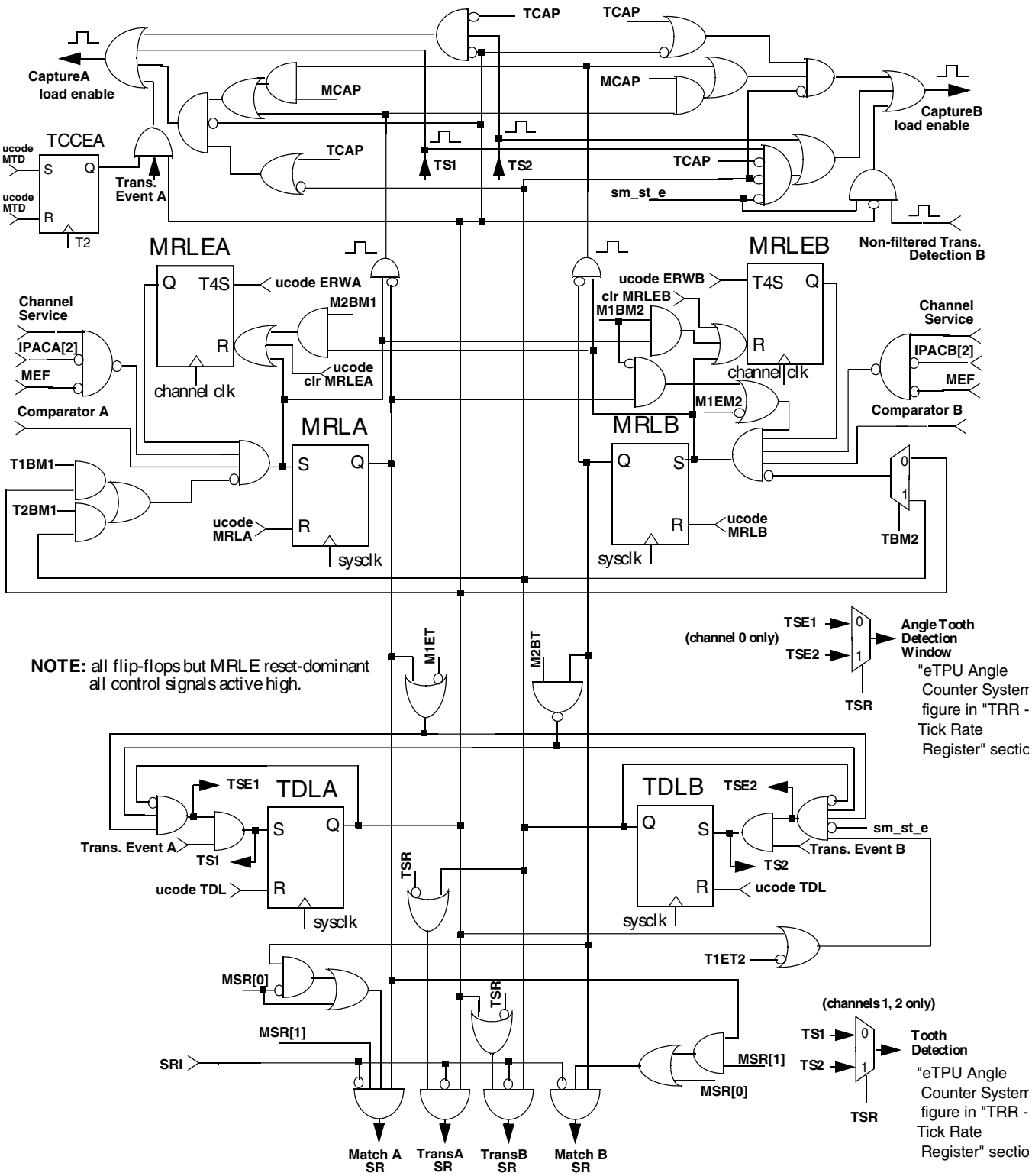


Figure 2-18. Channel Mode Logic and Event Flags

2.14.5.4.2 Channel Modes Overview

The predefined channel modes are divided, according to the way they treat transitions, in two basic modes:

- **Single Transition Modes** (mnemonic suffix **_st**): in these modes the first transition (flagged in TDLA) issues a service request, and captures both time bases (selected by TBSA[1] and TBSB[1]) except on sm_st_e. The second transition (flagged in TDLB) does not issue a service request, but it captures time base selected by TBSB[1], except on sm_st_e.
- **Double Transition Modes** (mnemonic suffix **_dt**): in these modes the second transition (flagged in TDLB) issues a service request, and each transition captures its own selected time base (Transition A and Transition B capture time bases selected by TBSA[1] and TBSB[1], respectively).

In predefined modes, Transition B is always (but not only) enabled by Transition A, so that transitions are always ordered: TDLA is set on the first transition and TDLB on the second. Unordered transitions are possible with user-defined mode, when UDCM bit T1ET2=0. Matches are generally not ordered, except on specific ordered match modes m2_o_st and m2_o_dt. Match capture(s) never overwrites a Transition capture, while Transition captures can always overwrite a Match capture, either in predefined or user-defined modes.

The following general rules apply to both predefined and user-defined modes:

- Blocking of one Match by the other, when it occurs, is done through MRLEs.
- Matches always block themselves by resetting their own MRLEs (Match A always blocks Match A, Match B always blocks Match B).

Predefined modes differ mostly by the way matches affects and are affected by other matches and transitions, as explained in next sections. However, some general rules on Match blocking apply:

- Match B is blocked by first transition (TDLA) in single transition modes, and by second transition (TDLB) in double transition modes.
- Both Matches are blocked by first transition in single transition modes.

Note

The rules above and in following sections may be overruled by the state of the channel latches if they are set/reset by microcode or if channel mode is changed. Care must be taken

to change channel modes, and is advisable to reset channel flags MRLA/B, TDLA/B and MRLEA/B before writing PDCM, or to UDCM when user-defined mode is selected.

2.14.5.4.2.1 Either Match, Blocking Modes (em_b_st, em_b_dt)

In these modes the first match recognition that occurs blocks the other match recognition and generates a service request. They end up with one service request for two programmed match recognitions where only the first match recognition has an actual effect. If both match recognitions occur at the same time, both MRLA and MRLB are set, before the mutual blocking takes effect. The associated logic is shown in the following figure.

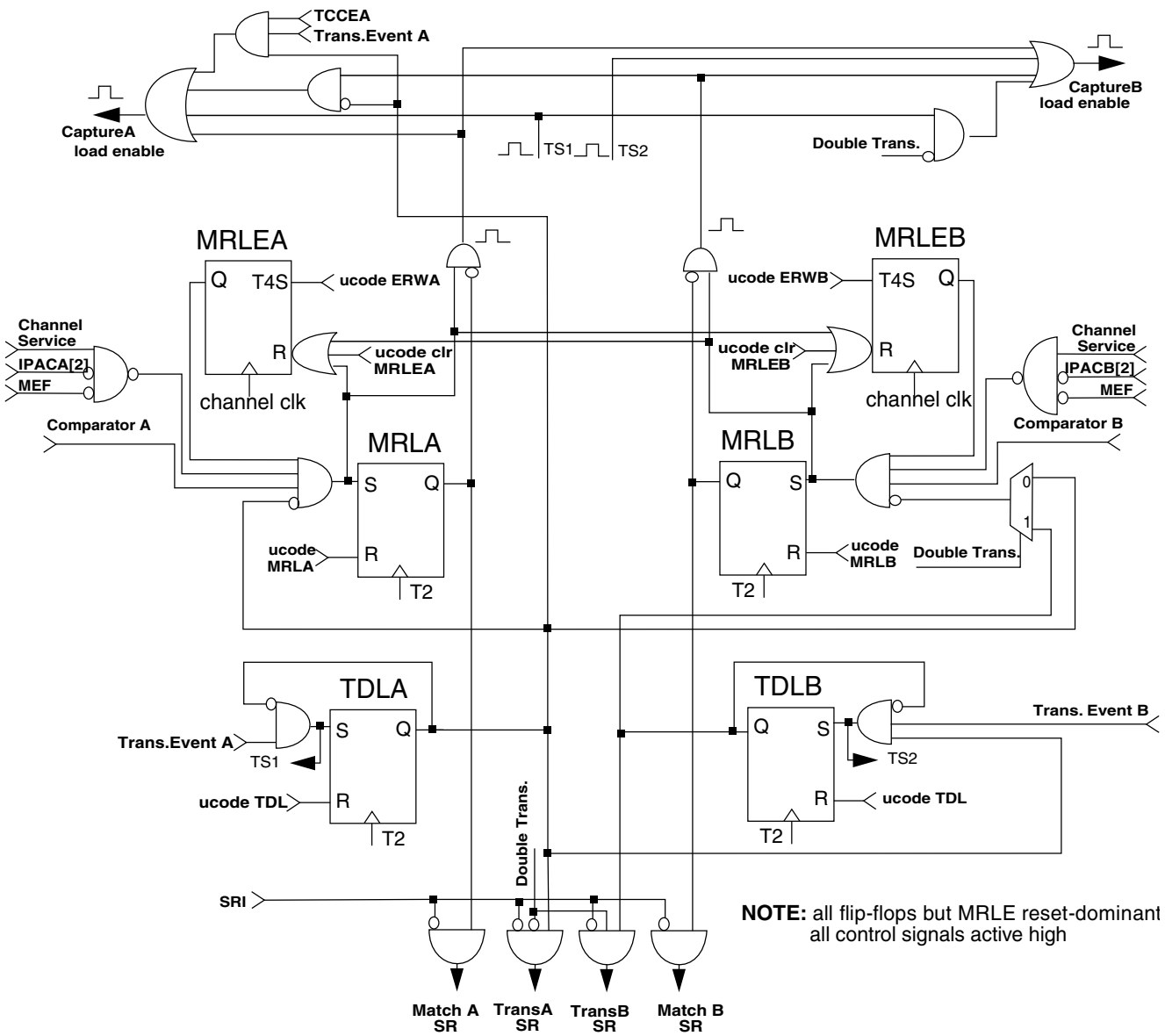


Figure 2-19. Either Match, Blocking Modes (em_b_st, em_b_dt)

2.14.5.4.2.2 Either Match, Non Blocking Modes (em_nb_st, em_nb_dt)

In these modes both match recognitions are independent and each of them generates service request. Each match recognition captures its related time base and does not block the other. The associated logic is shown in the following figure.

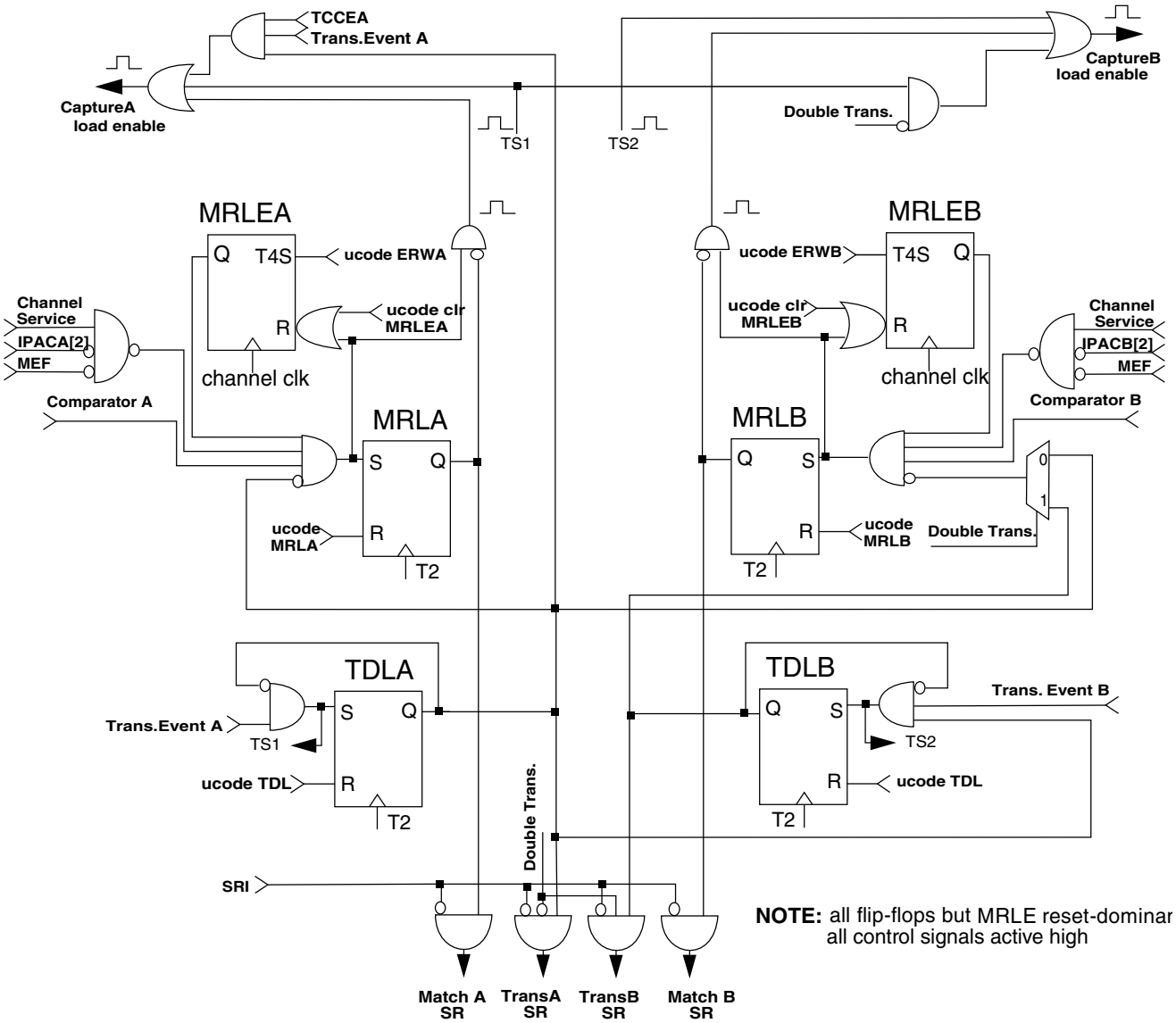


Figure 2-20. Either Match, Non Blocking Modes (em_nb_st, em_nb_dt)

2.14.5.4.2.3 Match B Request Modes (m2_st, m2_dt)

In these modes transitions are initially blocked, and are enabled by Match A. Match B recognition generates the match service request and disables Match A recognition. Each match recognition captures its own programmed timebase. In case of simultaneous match

recognition, both MRLA and MRLB are set, and OPACB register has priority over OPACA for selecting the pin action. The associated logic is shown in the following figure.

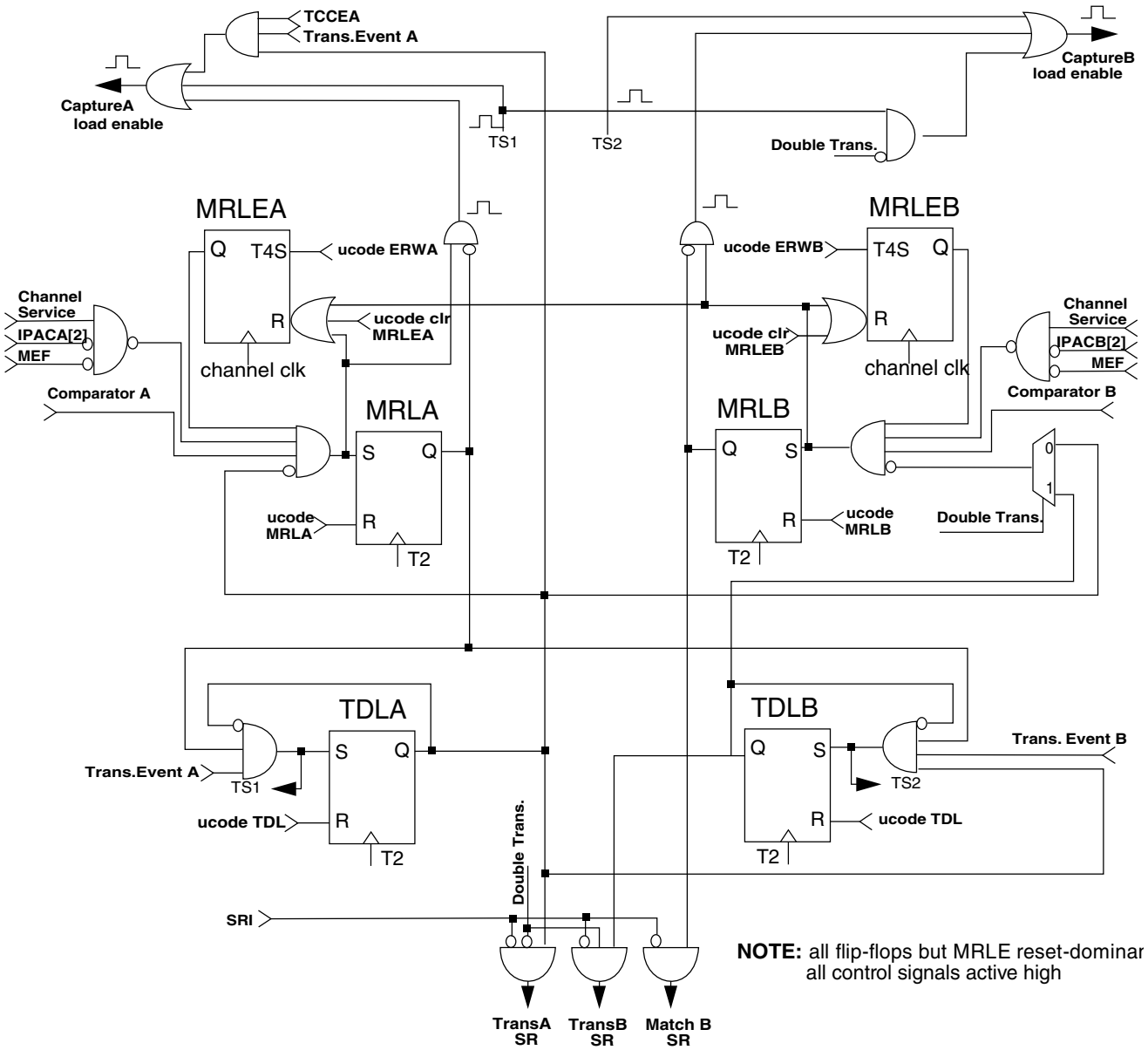


Figure 2-21. Match B Request Modes (m2_st, m2_dt)

2.14.5.4.2.4 Both Match Request Modes (bm_st, bm_dt)

In these modes, match service request is generated only after both match recognitions occurred. By definition this is a non-blocking match mode: match recognitions do not block each other, implementing a last-served scheme. Unlike other double transition

modes, bm_dt blocks Match A with Transition B (not with Transition A), so that the second transition blocks both matches. The associated logic is shown in the following figure.

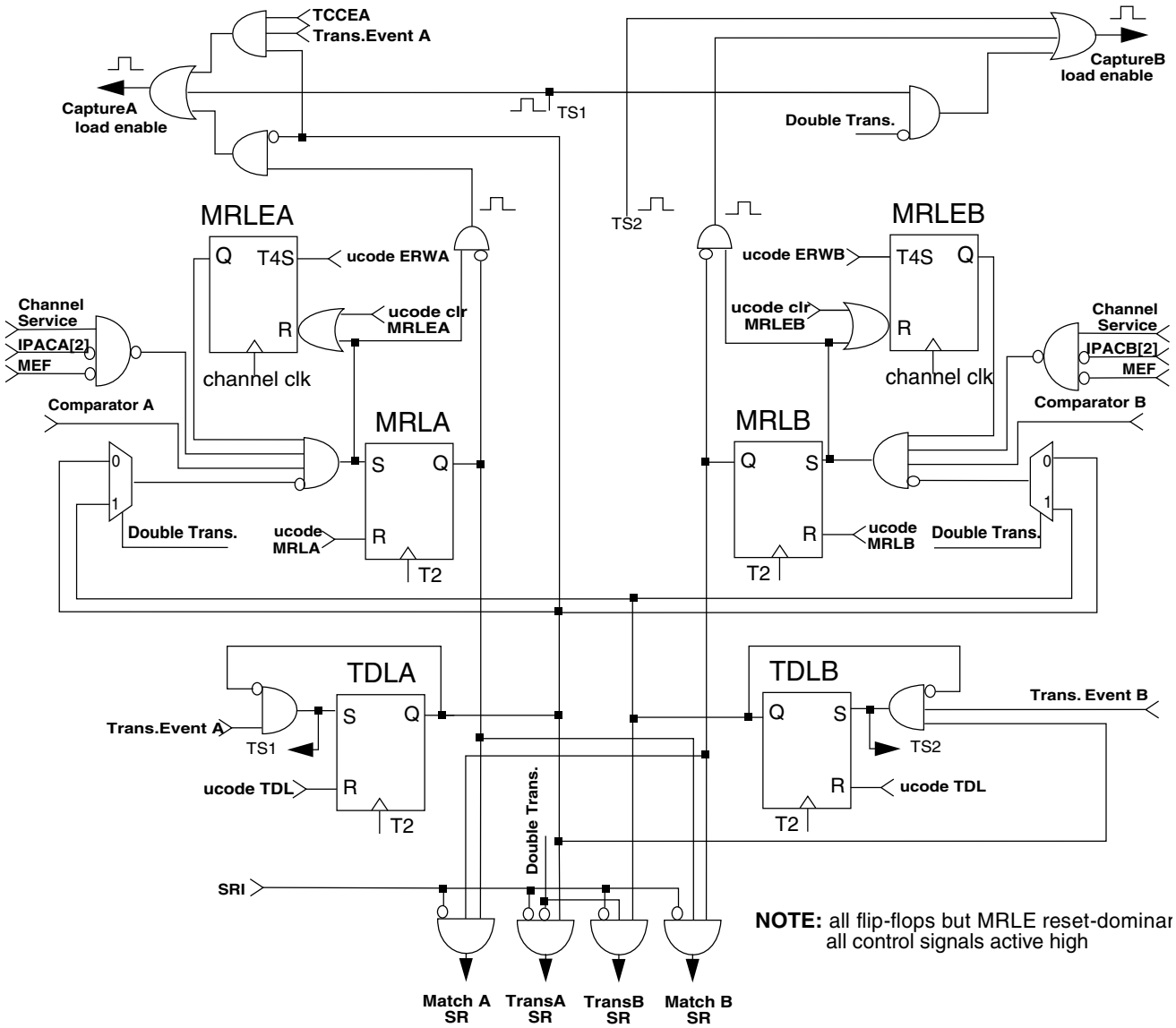


Figure 2-22. Both Match Request Modes (bm_st, bm_dt)

2.14.5.4.2.5 Ordered Modes with Match B Request (m2_o_st, m2_o_dt)

These are ordered match modes on which Match A recognition must precede Match B recognition (ordered 1->2). Match A asserts MRLA and enables Match B and transitions. Match B asserts MRLB, generates a match service request, and blocks both transitions. The associated logic is shown in the following figure.

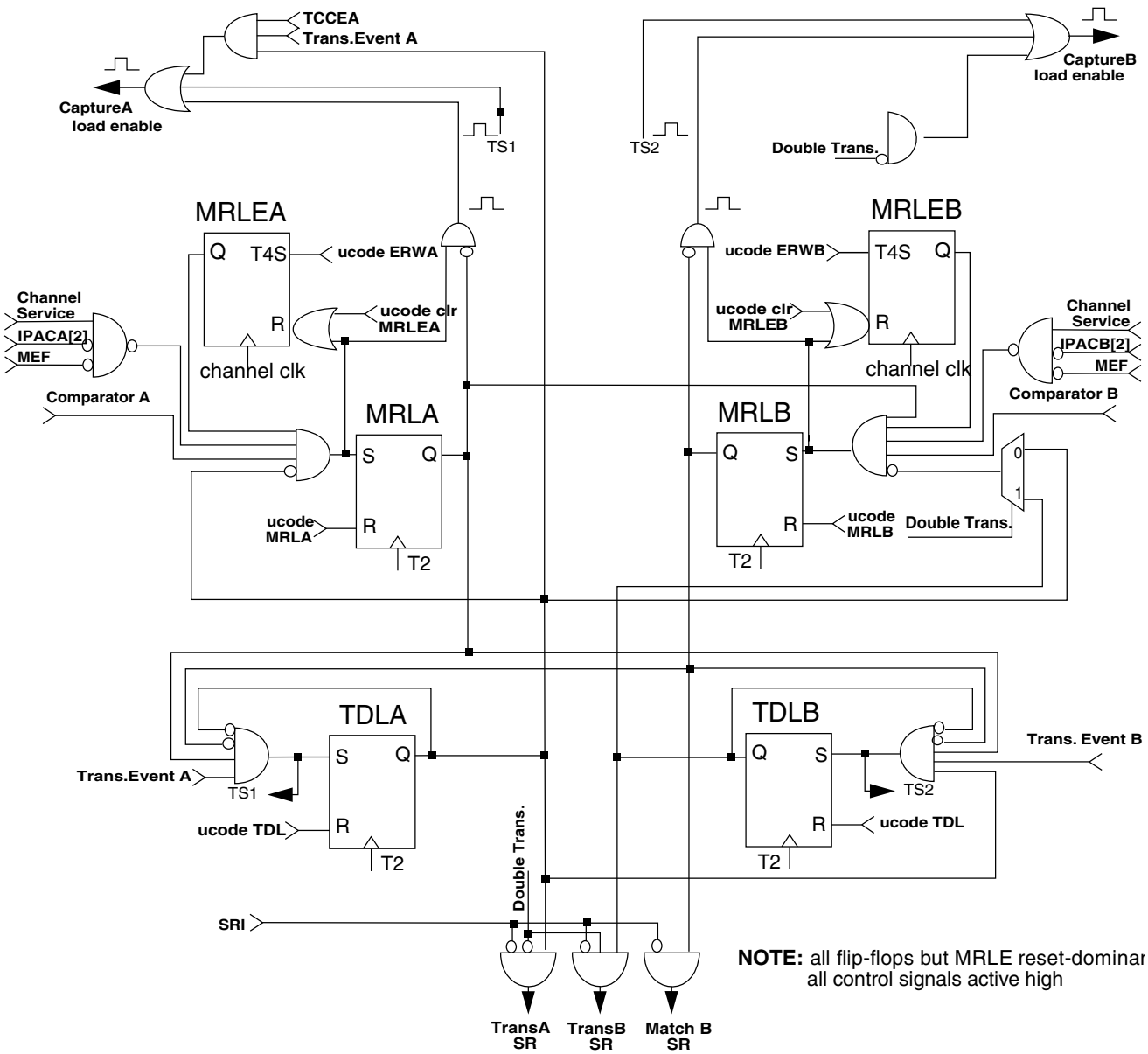


Figure 2-23. Ordered Modes with Match B Request (m2_o_st, m2_o_dt)

2.14.5.4.2.6 Single Match Modes (sm_st, sm_dt)

Single match modes support single or double transition with single match recognition. MRLB is never set, and MRLEB has no effect. The associated logic is shown in the following figure.

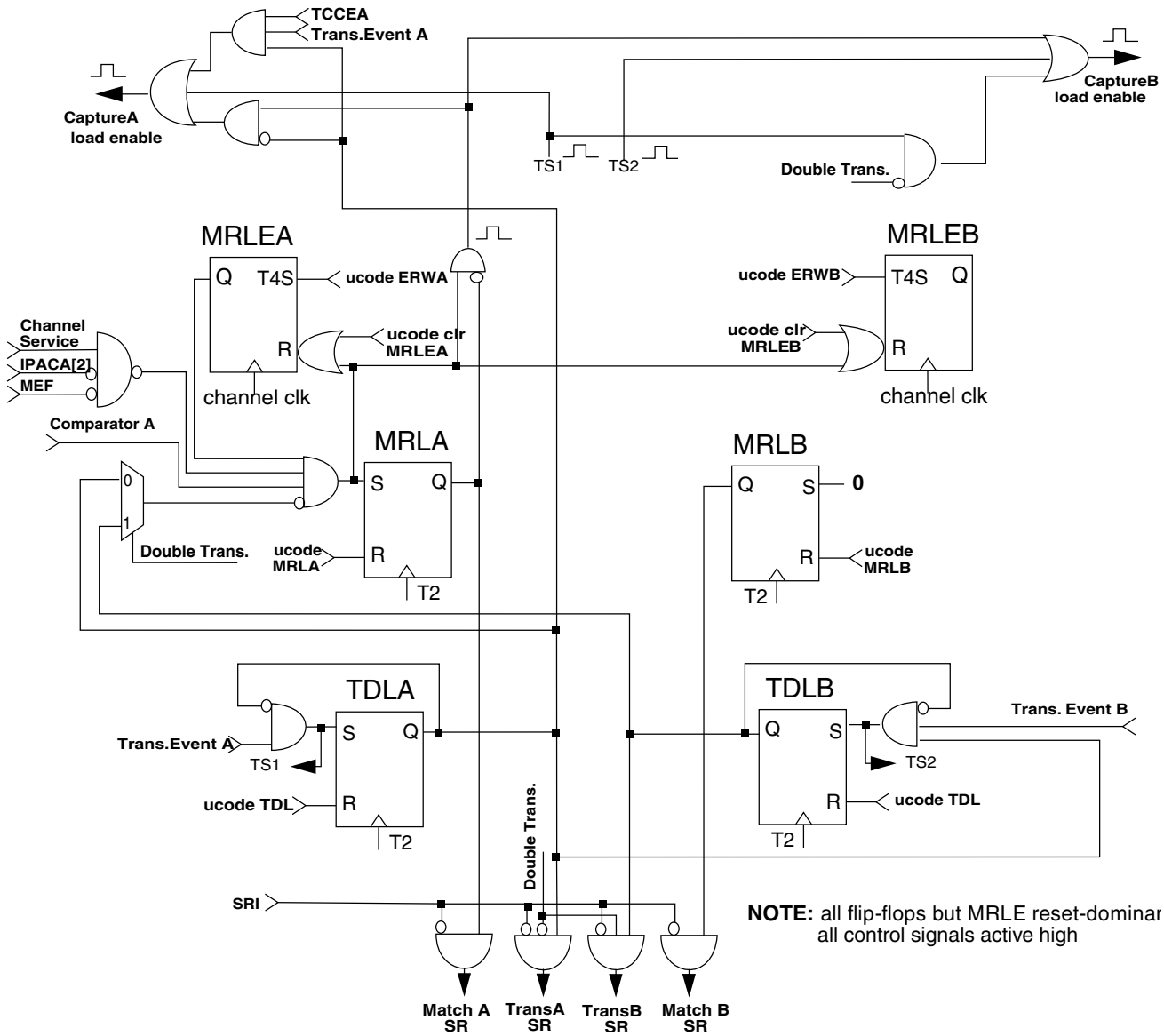


Figure 2-24. Single Match Modes (sm_st, sm_dt)

2.14.5.4.2.7 Single Match Enhanced Mode (sm_st_e)

This is an enhanced single transition and single capture mode, which provides non-filtered input captures in addition to the single capture, allowing one to measure the delay of the digital filter. In an output channel, it has the same functionality of sm_st (captures both time bases at once due to a match recognition). The associated logic is shown in the following figure.

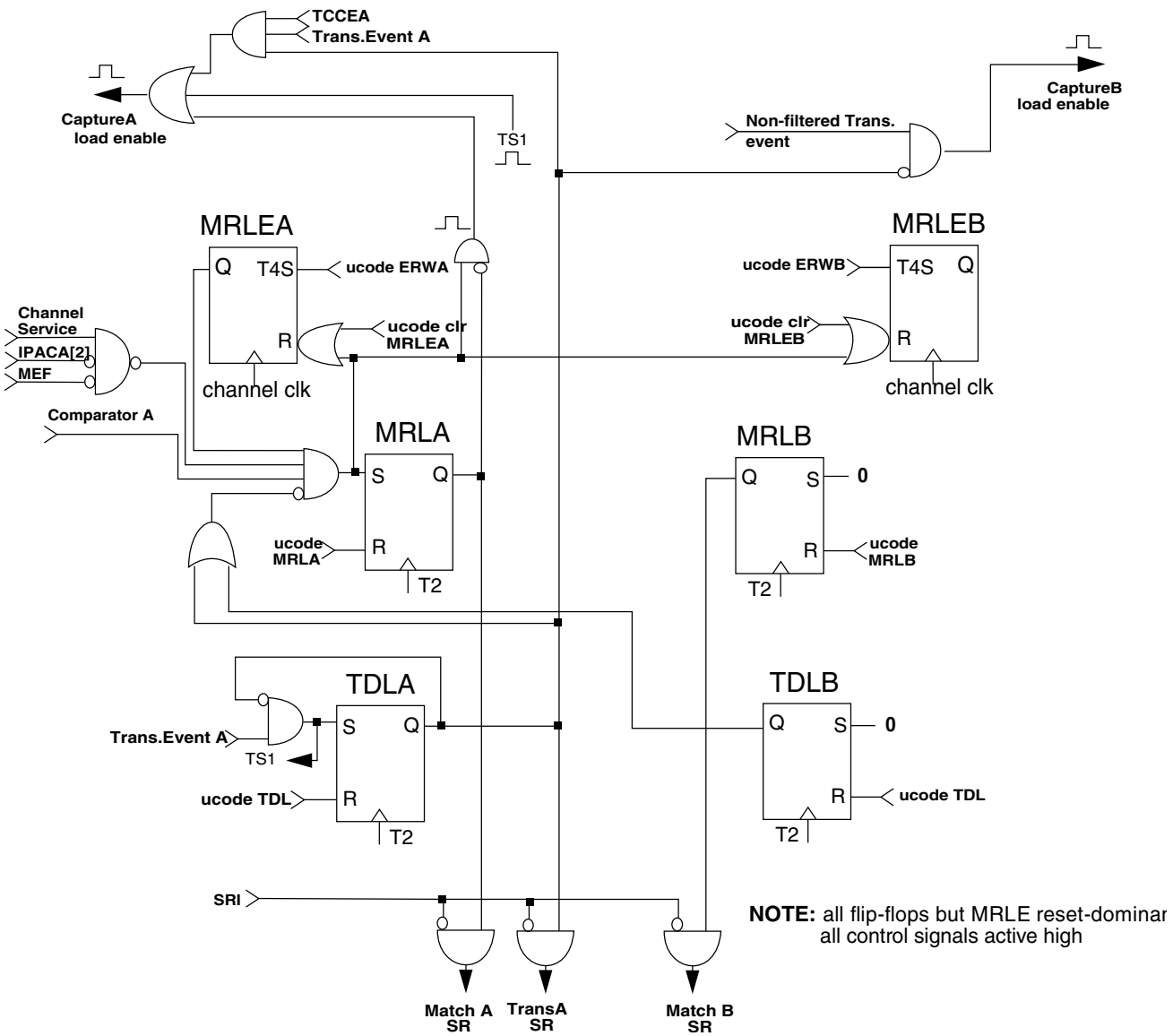


Figure 2-25. Single Match Enhanced Mode (sm_st_e)

2.14.5.4.3 Predefined Channel Modes on Input Signal Processing

When processing an input signal, the predefined channel modes can be classified in the following primary mode groups:

- Single Transition, Single Match: em_b_st, sm_st, sm_st_e
- Single Transition, Double Match: em_nb_st, bm_st, m2_st, m2_o_st
- Double Transition, Single Match: em_b_dt, sm_dt
- Double Transition, Double Match: em_nb_dt, bm_dt, m2_dt, m2_o_dt

In single transition modes, TDLA assertion may capture both time bases at once, while in double transition modes each transition captures its related time base in its related capture register.

Double transition is always ordered, i.e TDLB is enabled by TDLA and generates the service request.

The channel logic supports various input modes with combinations of single/double transition and single/double match, explained in the following subsections.

2.14.5.4.3.1 Either Match, Blocking, Single Transition (em_b_st)

On an input signal, this mode provides double timeout mechanism on a programmed transition edge with two timebases. The signal transition blocks both pending matches, indicating that no timeout condition occurred. The two match recognitions block each other, giving good separation in the entry table as to which match recognition caused the first timeout condition, and generating only one service request. Either match performs timebase captures which do not overwrite captures by transitions.

2.14.5.4.3.2 Either Match, Blocking, Double Transition (em_b_dt)

In double transition mode each transition is related to one match recognition. TDLA assertion captures its related timebase, blocks Match A and enables TDLB. TDLB assertion blocks Match B, captures its related timebase and generates a service request. Match recognitions block each other, so if there is a match timeout condition on TDLA, only one match service request is generated. This mode is good for qualifying two signal transitions by match timeout mechanisms, with one service request. Note that although TDLA assertion does not block Match B recognition, the value captured in CaptureA by TDLA assertion is not overwritten by this recognition. The second transition blocks Match B. Either match performs timebase captures which do not overwrite captures by transitions.

2.14.5.4.3.3 Either Match, Non Blocking, Single Transition (em_nb_st)

On an input signal, this is a double timeout mechanism of independent match recognitions of two different timebases. The match recognitions do not block each other, such that the microcode can check if one or two match recognitions occurred before their related signal transition. The signal transition detection (by IPACA) asserts TDLA, blocks both matches, captures both time bases and generates a transition service request, indicating that none of the two timeout conditions occurred. Any combination can be easily resolved by microcode (for example, signal transition after Match A and before Match B, or signal transition after both Match A and Match B).

Another possible use of this mode is allocating one match recognition for transition timeout and the other for another non-critical timed task, adding functionality to a single channel. Since the transition detection blocks both match recognitions, the match recognition of the other timed task is based on the fact that the comparator checks greater-equal conditions. It may be delayed if it occurs in the period between the signal transition detection (which blocks it) and the time TDLA is negated by microcode. If matches are enabled during the service, the same code can check if the match recognition of the timed task occurred in this period, by negating TDLA and writing to the CHAN register its own value (in order to update the MRLA flag in the branch logic).

2.14.5.4.3.4 Either Match, Non Blocking, Double Transition (em_nb_dt)

In this mode each transition is related to one match recognition, and the match recognitions are independent of each other. This mode can be used to give independent timeout conditions for the first and the second signal transition recognitions, and call service in any case of any timeout condition.

The first transition detection programmed in IPACA sets TDLA, captures its related timebase, blocks Match A recognition and enables TDLB assertion. The second transition detection programmed in IPACB sets TDLB, blocks Match B recognition, captures its related timebase and generates a service request. Any match recognition that occurs captures its related time base and generates a match service request, independent on the other match recognition.

2.14.5.4.3.5 Match B Request, Single Transition (m2_st)

On an input signal, this mode provides an open window filter for a single signal transition. MRLA assertion opens the window, and enables transition detection on TDLA from this time on. MRLB assertion blocks Match A (by negating MRLEA), providing conditional window opening, because transitions are indirectly blocked. It also generates service request, but if it happens after Match A it does not block transitions, providing a non-blocking timeout mechanism for the estimated signal transition time (typically it indicates a missing transition, or mis-prediction of the transition time).

Transitions can be detected from the microcycle following MRLA assertion. The Transition A detection asserts TDLA, blocks both matches, captures both timebases and generates service request.

Using this mode, the channel can replace software open window filtering of qualified transitions with the channel hardware window. The window opening and timeout can be scheduled for any of the two time bases or combination of them. Typically, Match A will be used to open a prediction window, and Match B will be used as a timeout condition

which does not close the prediction window. This configuration improves noise immunity from early signal transitions, and reduces the probability for blocking late signal transitions due to timeout mis-prediction.

Using these conditions, the microcode can easily resolve the state:

- If TDLA and MRLA are asserted and MRLB negated, signal transition is in the expected range.
- If MRLA and MRLB are both asserted, and TDLA is asserted, the signal transition had a timeout condition due to Match B mis-prediction.
- If MRLB is asserted and TDLA negated, a timeout condition occurred, and the expected signal transition had not occurred yet.
- If MRLA is negated and MRLB is asserted, the conditional window did not open at all (for example: a time window is open only after a specific angle, otherwise it is not opened).

2.14.5.4.3.6 Match B Request, Double Transition (m2_dt)

This mode is used as an open window filter for two signal transitions. In this case the Match A recognition opens the window (unless Match B recognition occurred first), and Match B recognition blocks Match A and generates a match service request. It is similar to m2_st, but in this case, it is the second transition that blocks Match B. MRLB assertion is a global timeout condition for the two pulses. Like m2_st, MRLB can conditionally eliminate the window from opening.

Using the TDLA, TDLB, MRLA and MRLB conditions, the microcode can easily resolve the state, in a similar manner as m2_st, with additional information on the second transition (TDLB).

2.14.5.4.3.7 Both Match Request, Single Transition (bm_st)

On an input signal, this is a double timeout mechanism on two different time bases. Both match recognitions must occur before the signal transition to generate a match timeout service request. Assertion of TDLA blocks both Match A and Match B recognitions, and captures both time bases, indicating there was no double timeout condition from both time bases.

Using the same timebase implements two timeout conditions, the first only sets its related MRL and the second generates a service request. Using these flags allows the microcode to check if one or both match recognitions precede the signal transition.

2.14.5.4.3.8 Both Match Request, Double Transition (bm_dt)

In this mode the first transition detection does not block matches, since both match recognitions are required to generate a match service request. The second transition detection asserts TDLB, blocks Match A and Match B, captures its related timebase and generates transition service request. In this mode, a Match A recognition which occurs after the assertion of TDLA does not capture a new value in CaptureA, to preserve the actual signal transition time. Assertion of TDLA, however, always captures its related timebase.

This mode allows putting a double match timeout condition on the second transition. Typically, a pulse trailing edge timing can be checked against two time bases, to indicate if the pulse has not ended when both MRLA and MRLB are asserted. When a transition service request is generated by TDLB assertion, the state of MRLA and MRLB indicates which timeout condition occurred, if any.

2.14.5.4.3.9 Ordered Mode with Match B Request, Single Transition (m2_o_st)

On an input channel, this mode provides a closing window filter for a single signal transition. Match A assertion captures its programmed time base in CaptureA, opens the filter window (enables assertion of TDLA), and enables assertion of MRLB. Match B recognition captures its related timebase, closes the window (disables assertion of TDLA) and generates a service request. Due to Match A and Match B ordering, the window is opened for at least one microcycle. Match B recognition indicates a window timeout condition which blocks late signal transitions, outside the prediction window. Transition detection blocks both matches, indicating the transition occurred inside the estimated window. Transitions can be detected from the microcycle following MRLA assertion until the microcycle on which MRLB is asserted. When TDLA is asserted inside the window range it disables both matches, captures both time bases and generates a transition service request.

Using this mode, the channel can replace software window filtering of qualified transitions with the channel hardware window. The window opening and closing time can be scheduled for any of the two time bases or a combination of them.

2.14.5.4.3.10 Ordered Mode with Match B Request, Double Transition (m2_o_dt)

In this mode the channel logic implements a window filter for two detected signal transitions. MRLA assertion captures its related timebase and enables the assertion of both TDLA and TDLB. MRLB assertion captures its related timebase and disables the assertion of both TDLA and TDLB. Transitions can be detected from the microcycle following MRLA assertion until the microcycle on which MRLB is asserted. The first

signal transition (following MRLA assertion) asserts TDLA, captures its related timebase and enables assertion of TDLB. The second signal transition detection asserts TDLB, blocks Match B, captures its related timebase and generates the service request.

If both signal transitions occur inside the scheduled window, Match B recognition is blocked. If one or both signal transitions do not occur inside the scheduled window, Match B recognition generates a match service request and blocks further transition detections. The microcode can resolve the state using MRLA, MRLB, TDLA and TDLB, which affect the entry point selection.

2.14.5.4.3.11 Single Match Enhanced Mode (sm_st_e)

This is an enhanced single transition and single match channel mode which provides timing information of the digital filter delay.

The CaptureA register captures the timebase selected by TBSA due to transition detection specified by IPACA or match recognition, as in sm_st mode. Initially, the CaptureB register continuously captures the unfiltered IPACB-selected signal transitions from the digital filter input, directly from the signal synchronizer. When an IPACA-qualified, filtered transition detection occurs, TDLA is set, MRLA assertion is blocked, and, in addition, captures into CaptureB are also blocked. On service, CaptureA and CaptureB (copied into ERTA and ERTB) holds the time of the qualified transition detection (ERTA), and the time of the last signal transition at the input of the digital filter (ERTB). Subtracting the time in ERTB from the time in ERTA provides the delay of the digital filter.

In a quiet environment, the two captures provide the accurate delay of the digital filter in granularity of two eTPU clocks. In a noisy environment, false transitions may be detected at the input of the digital filter due to the noise, and the delay measurement may be reduced, especially if IPACB selects both edge detection. The microcode can do sanity checks on this value to recognize noise effects (for example: calculated delay is less than the minimum delay of the digital filter).

Note

In Channel 0, if eTPU_TBR_TBCR_ENG*n*[AM]=01 (Angle Mode), the unfiltered input comes from TCRCLK input and the filtered input comes from the TCRCLK filter output. The edge is selected by IPACA/B, and is independent of the edge selection by eTPU_TBR_TBCR_ENG*n*[TCR2CTL].

2.14.5.4.3.12 Single Match, Single Transition (sm_st)

In this mode the channel logic is functionally back-compatible to a TPU3 single action channel, but a match or transition detection captures at once both timebases. The mode recognizes a single transition with single match timeout. Either TDLA or MRLA generates service request and captures both timebases. Assertion of TDLA blocks future assertions of MRLA.

2.14.5.4.3.13 Single Match, Double Transition (sm_dt)

In this mode, the first transition detection asserts TDLA, captures a timebase in CaptureA and enables TDLB. The second signal transition asserts TDLB, blocks Match A, captures a timebase in CaptureB and generates a service request.

Match A (before TDLB) captures into CaptureB the timebase selected by TBSA, in order not to overwrite the captured value of TDLA.

This mode is used for scheduling one timeout condition on two input signal transitions (pulse timeout).

2.14.5.4.4 Channel Modes on Output Signal Generation

Since channel logic can generate output signal transitions based on Matches, the channel can be viewed as working in the following primary mode groups for signal generation:

- Single Match: em_b_st, sm_st, sm_st_e, em_b_dt, sm_dt
- Double Match: em_nb_st, bm_st, m2_st, m2_o_st, em_nb_dt, bm_dt, m2_dt, m2_o_dt

The channel logic supports various match channel modes with single/double match, as explained in the following subsections.

2.14.5.4.4.1 Either Match, Blocking Modes (em_b_st, em_b_dt)

On an output signal these modes are useful when using two different time bases to set a required signal transition. The first match condition which is met sets a required pin action, captures both time bases, blocks any effects of the other recognition, and generates a service request. Because the first match recognition blocks the other, the microcode can get good separation in the function entry table as to which match caused the timeout first, and both time bases are captured, enabling the microcode to compare one timebase to the other at the moment of the match recognition. These modes can be used for:

- Scheduling a required pin action to the first match recognition of two different time bases.
- Canceling a programmed pin action scheduled on one time base by match on another timebase (as a consequence of Table 2-43). Microcode has to set the OPAC register of the canceling match to no-action and the OPAC register of the other match to the required pin action which may be blocked. If Match A is the canceling match, it blocks the pin action also in case of two matches at the same time, since it has priority in this case. If Match B is the canceling match, it does not block the pin action in case of two matches at the same time.

2.14.5.4.4.2 Either Match, Non Blocking Modes (em_nb_st, em_nb_dt)

On an output signal these modes are useful in combination with the ME bit set on the entry point, to define an interlaced operation. For example, each match recognition can set a pin action, and the second pin action is not sensitive to microcode latency (ME bit asserted). Example for usage is PWM interlaced function on which the latency is determined by the period and not the duty cycle.

Another possibility is using one match for pin actions and the other match for an unrelated timed task without pin action (double the functionality of a single channel).

2.14.5.4.4.3 Match B Request Modes (m2_st, m2_dt)

On an output signal, these modes can generate narrow pulses or do conditional pin actions. A conditional pin action means that the pin state is changed only if the match recognitions occurred in the correct order, since the Match B recognition which generates the service request also has priority over the pin action and blocks future Match A recognitions.

Setting OPACA to a desired pin action and OPACB to no-action, and using different time bases for Match A and Match B defines a conditional OPACA pin action which can be blocked by Match B recognition. For example, setting Match A on time and Match B on angle can limit the pin action to a maximum angle value.

When pulses are generated, the service is requested at the trailing edge of the pulse, after MRLB is asserted.

2.14.5.4.4.4 Both Match Request Modes (bm_st, bm_dt)

On an output signal, each match recognition can affect the pin state, and capture its programmed time base. This way the pin action can be programmed separately for both match recognitions. For example, both match recognitions can negate the signal, and

service request is generated after both conditions are met. This mechanism can set two conditions to do a required pin action, and the first recognition changes the signal, but service is called only after both conditions occur.

When using the same time base, these modes can generate narrow pulses in any required order. For example, in a PWM function, when duty cycle is below 50% the function can get service on the low time and program the pulse to the required duty cycle of the high time. When duty cycle is equal or above 50%, the function can get service on the high time and program a negative pulse with the width of the required low time. To switch between the two states the function can program once the same transition time to MatchA and Match B with a required pin action, and on the next service program double match for the new state.

Another usage is generating a required pin action on one programmed time and service request later on another time, after the second match recognition occurs, or capturing some timebase on one time and generating a required signal transition and service request later.

2.14.5.4.4.5 Ordered Modes with Match B Request (m2_o_st, m2_o_dt)

The order of the match recognitions imply that OPACA register programmed pin action always precede the OPACB register pin action. Setting OPACA to no-action, based on the greater-equal comparator, enables using Match A on one time base to delay the signal effect of Match B on the other time base. This method implements a conditional pulse extension or conditional delay on signal transition.

These modes can also be used for deferred pulse generation with microcode service request after its trailing edge (if Match A condition comes after Match B condition). Another option is having Match A recognition associated with output pin actions and Match B recognition for a timed microcode task which has to be scheduled at a programmed time which may be delayed by the Match A pin action.

2.14.5.4.4.6 Single Match Modes (sm_st, sm_dt, sm_st_e)

There is no difference between plain and enhanced Single Match modes on an output signal.

In this mode the channel logic is functionally back-compatible to a TPU3 single match output channel. Match A recognition generates service request and sets the pin state according to OPACA register. It captures at once the timebase selected by TBSA in CaptureA and the timebase selected by TBSB in CaptureB.

2.14.5.4.4.7 Match/Transition Pin Action Conflict Resolution

In output signals, matches and/or transitions automatically cause pin actions defined by the OPACA/B and/or IPACA/B channel control registers (see Pin Control Registers). Simultaneous matches/transitions may be associated with different, possibly contradictory, pin actions. These conflicts are resolved according to the table below.

If an OPACA/B=000 (no action) prevails over non-zero OPAC according to the table below, then if Match A/Transition A and Match B/Transition B occur simultaneously, no output pin action occurs, that is: a match on the action logic with OPAC=000 inhibits simultaneous actions of the other OPAC, if prevailing according to the following table. That also applies when output actions are caused by inputs (OPAC=1xx).

Table 2-43. Simultaneous Match Pin Action Priority

Channel Mode	Priority
em_nb_st / em_nb_dt	OPACA
em_b_st / em_b_dt	OPACA
bm_st / bm_dt	OPACA
m2_st / m2_dt	OPACB
m2_o_st / m2_o_dt	in these modes there is no possibility of simultaneous matches
user-defined	OPACB if M2BM1 = 1 and M1BM2 = 0, OPACA otherwise

2.14.5.4.5 Combining Input and Output Signals

The processing of input signal can be combined with output signal generation. A detected input transition can trigger an output signal edge, even without microcode intervention, by using OPAC options 1xx.

The channel set-up examples below show these two capabilities combined (see the figure below).

The **first example** implements a fast (no microcode intervention) short-circuit protection feedback mechanism for driving high-current output devices. The signal after the high-current driver feeds back to the channel input. The input signal is normally delayed from the output signal by the device turn-on delay. After the channel output turns on, the channel logic must check if the driver output (connected to the channel input) follows the driven value after the maximum device turn-on delay. If it does not, the driver output is probably shorted, and the channel output must be turned off immediately to avoid damaging the device.

Match A causes the output to be driven high (for simplicity the output and input signals are shown as positive logic). It also causes a transition A, because IPACA=100 and the input is still low. Match B occurs after the expected driver delay, and causes a service request. If the output is shorted, a Transition B occurs on Match B because IPACB=100. This will cause the output to go low immediately, because OPACB=100.

In the **second example**, an output pulse is generated from an input transition without microcode intervention. Match A opens a window for transitions and also enables Match B. A rising edge on input sets output high. On Match B the window closes, and input signal is checked: if sampled high, the output resets; otherwise it stays high.

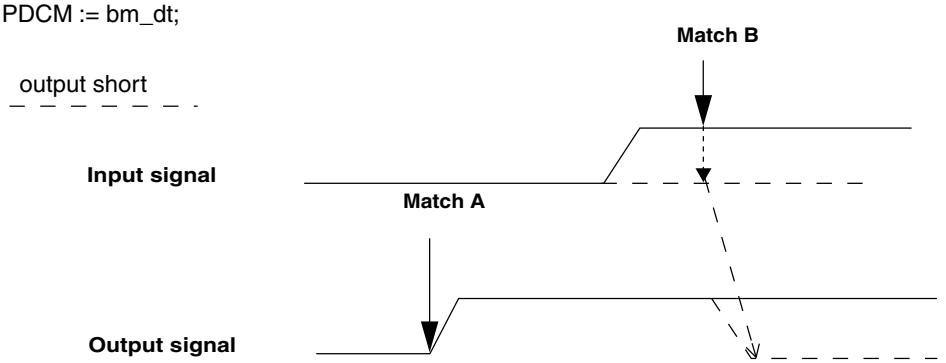
In the **third example**, a pulse is generated depending on the value sampled on the input signal at a predetermined time. Match A samples the input signal, causing a Transition. Low level on input sets output low, otherwise it stays high. Match B sets output high. In both cases a service request is issued (microcode intervention), at the beginning and at the end of the pulse (Match B), if required (SRI := 0).

Note

When IPAC = 1xx, a match event can cause simultaneously a Match recognition and a Transition detection. Depending on the Channel Mode, these Match and Transition may have conflicting effects on other Transition/Match blocking or enabling. In these cases, blocking always prevails over enabling, effective on the next microcycle.

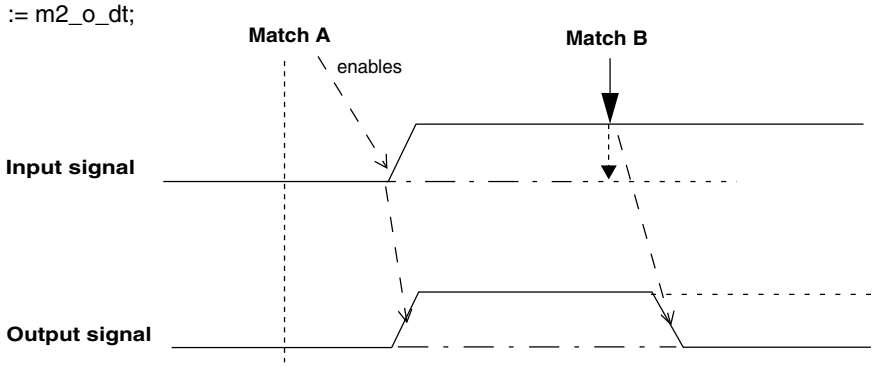
Example 1: Short-circuit protection feedback

IPACA := 100; OPACA := 001; MatchA := output activate time;
IPACB := 100; OPACB := 100; MatchB := MatchA + max. high-current driver turn-on delay;
PDCM := bm_dt;



Example 2: Pulse generation on windowed input transition

IPACA := 001; OPACA := 101; MatchA := window open time;
IPACB := 101; OPACB := 100; MatchB := window close time, input sampling;
PDCM := m2_o_dt;



Example 3: Pulse generation on input sampling

IPACA := 100; OPACA := 100; MatchA := window open time,input sampling;
IPACB := 000; OPACB := 001; MatchB := window close time = MatchA + pulse width;
PDCM := em_nb_dt;

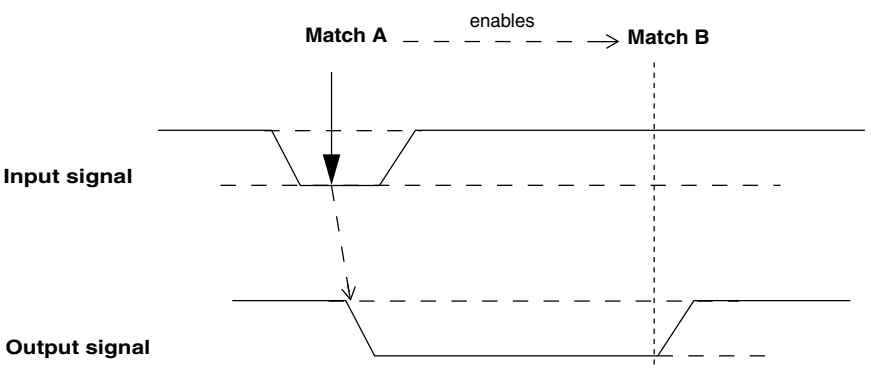


Figure 2-26. Input/Output combination

2.14.5.5 Channel Link

A channel can issue service requests to other channels through microcode, by assigning to the write-only microengine register LINK (refer to LINK Register) a value which specifies the target channel of the Link Service Request, as shown in the figure below.

Writing to the LINK register issues a link request to the target channel, setting its LSR flag. Each channel has its own LSR flag, which can be tested as a microcode branch condition (see Conditional/unconditional branch) and reset through the microcode field LSR (see Clear Link Service Request). The link branch condition samples, at the TST start, the value used to calculate the Entry Point.

Writing LINK with another channel target value in the same thread issues a Link Service Request to the new target, without negating the service request to the former one. This allows a channel to issue service requests to any number of channels, including itself. Neither LINK nor LSR microengine accesses are qualified by the CHAN register, i.e., they always access the serviced channel LINK and LSR, regardless of the value written in CHAN.

Channels can also receive link service requests from the Shared Subsystem Units (see Shared Subsystems support). These links are set by the interrupt signals from the shared subsystem and are level-sensitive, that is, they keep on requesting link service as long as the interrupt is active. Therefore, it is not enough to clear the link service request through the LSR instruction field: one must clear beforehand the respective shared subsystem unit interrupt source.

If microcode executes an instruction with field LSR=0 (clear Link Service Request), the link branch condition is cleared. However, the link service request itself is cleared only if no link from other channel was received by the serviced channel during the same thread[27] . Note that this applies only to links that come from the other engine or the same channel: if only Shared Subsystem link was received during the thread, the link service is also cleared with field LSR=0. If microengine clears LSR of its channel and, simultaneously, Link Service Request is issued to the current serviced channel, the branch condition is cleared but the link service request remains pending.

7	6	5	4	3	2	1	0
Engine Selection		Reserved[1]	Channel Number				

NOTES:
1 reserved bit must be written 0.

Figure 2-27. Microengine LINK Register

27. That can only happen if the link service request came from the other Engine, or from the serviced channel itself.

A channel can issue Link Service Requests to channels in any of two Engines, determined by the LINK register field Engine Selection (2 bits), as shown in the following table. In a Single Engine eTPU, Link is ignored when sent to the other engine, or engine 2.

Table 2-44. LINK Engine Selection

Engine Selection	Description
00	this Engine
01	Engine 1
10[1]	Engine 2
11[1]	the other Engine

1. ignored in Single Engine eTPU.

There is no explicit hardware mechanism to allow the thread to identify the link service request source, but it can be done by a software protocol using SDM parameters. All links are negated on reset.

2.14.5.6 Enhanced Digital Filter - EDF

The EDF eliminates passing of signal transitions which are caused by noise. Its purpose is to eliminate false transition service requests caused by noise pulses which are shorter than a programmed width.

See the chip-specific eTPU information for connections between eTPU and input glitch filters.

The EDF has three modes of operations, selected by the CDFC field in eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2). These modes offer selections of trade-off between noise immunity and signal latency. CDFC also allows the filter to be bypassed. Table 2-45 gives an example of minimum detected signal pulse and maximum filtered noise pulse in the three EDF operation modes. In Angle Mode, if AM=01, the EDF in channel 0 is replaced with the digital filter and synchronizer of the TCRCLK signal. In this mode, channel 0 works in combination with the Angle Counter logic, and their operation is fully synchronized.

Following subsections provide the functional description of the eTPU channel digital filter.

2.14.5.6.1 Two-Sample mode

In this mode the EDF works like the TPU2/3 digital filter. It uses the filter clock which is the eTPU clock divided by (2, 4, 8,.., 256) as a sampling clock. The filter clock is selected by the FPSCK field ineTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2). The EDF compares two consecutive samples. If both samples have the same value, the input signal state is updated. Note that when the FPSCK field selects the eTPU clock divided by two, the EDF works like the TPU1 four-clock digital filter.

2.14.5.6.2 Three-Sample mode

In this mode, like in the TPU2/3 mode, the EDF uses the filter clock as a sampling clock. The EDF compares three consecutive samples. If all three samples have the same value, the input signal state is updated.

The Three-Sample mode gives more signal latency than the Two-Sample mode, but also better noise immunity and better ratio between minimum detected signal pulse to maximum filtered noise pulse. When a certain filter clock frequency is selected for Two-sample mode, double filter clock frequency can be selected to get better latency in Three-sample mode.

2.14.5.6.3 Continuous mode

In this mode the EDF compares all the values sampled at the rate of eTPU clock divided by two, between two consecutive filter clock pulses. If the signal is continuously stable for the entire digital filter clock period (i.e all the samples have the same signal value), the input signal state is updated.

This method gives the same latency and the same ratio between minimum detected signal pulse to maximum filtered noise pulse, as the Two-Sample mode, as long as there is no noise. Each sampled noise delays the signal transition detection by at least a whole digital filter clock period.

The Continuous mode gives the best noise immunity by comparing multiple samples of the noise. On the other hand, when a short noise pulse appears in the middle of the filter clock period at the same time of a real signal transition, the Continuous mode may reject a real signal transition and delay the response to the first filter clock period in which the signal is continuously stable. This may add to the latency and also to the minimum detected signal pulse in a noisy environment.

2.14.5.6.4 Bypass Mode

In bypass mode the signal that feeds the edge detection comes directly from the output of the synchronizer, not filtered. Bypass mode automatically makes the channel logic work in T2/T4 timing mode (see Channel timing modes).

2.14.5.6.5 Filter clock prescaler

Each channel configured as an input has an associated synchronizer followed by a digital filter connected to the signal that samples signal transitions. After reset, the digital filter filters out high and low pulse widths smaller than the period of two eTPU clocks with eTPU_SCR_ECR_ENG*n* field FCSS=0, or 1 eTPU clock with FCSS=1 (see eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2)), preventing these transitions from being input to the transition detect logic. For FPSCK=0 and FCSS=0, the synchronizer and digital filter are guaranteed to pass pulses that are as wide as or wider than four eTPU clocks, meaning a minimum period of 8 eTPU clocks. These figures are halved by setting FCSS=1. By changing the FPSCK field in eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2) the user can select a lower clock rate for the filter signal to define wider valid pulses and filter out wider noise pulses. The filter prescaler clock control is a division of the eTPU clock. To guarantee pulse detection by the digital filter, the pulse must cover at least the stated number of samples at the filter clock rate. For example, a two sample digital filter must sample two points in the pulse to detect it. The table below shows the minimum guaranteed detected pulse width and the maximum filtered noise pulse width. The table refers only to the digital filter operation. The external pulses may have to be wider (to ensure detection) or narrower (to ensure filtering) depending on the rise/fall delay differences in the MCU receivers and internal logic. Delays introduced by synchronizer, filter and edge detection logic are explained in Input/Output Signal Delays.

The TCRCLK signal also has a synchronizer and a filter. The FCSS bit has no influence on the clock prescaler of the TCRCLK filter, which works like the channel filter with FCSS=0.

Table 2-45. Pulse widths and delays

Filter control (FPSCK)		Filter clock period[1]	Min. width guaranteed detected / max. width filtered (min. filter delay / max. filter delay)[1]	
FCSS = 0	FCSS = 1		Two-Sample or Continuous mode	Three-Sample or Integrator[2] mode
not avail.	000	1	2 / 1 (2 / 3)	3 / 2 (3 / 4)
000[3]	001	2	4 / 2 (3 / 3)	6 / 4 (5 / 5)

Table continues on the next page...

Table 2-45. Pulse widths and delays (continued)

Filter control (FPSCK)		Filter clock period[1]	Min. width guaranteed detected / max. width filtered (min. filter delay / max. filter delay)[1]	
FCSS = 0	FCSS = 1		Two-Sample or Continuous mode	Three-Sample or Integrator[2] mode
001	010	4	8 / 4 (5 / 7)	12 / 8 (9 / 11)
010	011	8	16 / 8 (9 / 15)	24 / 16 (17 / 23)
011	100	16	32 / 16 (17 / 31)	48 / 32 (33 / 47)
100	101	32	64 / 32 (33 / 63)	96 / 64 (65 / 95)
101	110	64	128 / 64 (65 / 127)	192 / 128 (129 / 191)
110	111	128	256 / 128 (129 / 255)	384 / 256 (257 / 383)
111	not avail.	256	512 / 256 (257 / 511)	768 / 512 (513 / 767)

1. This table shows pulse widths and delays in number of periods of the eTPU clock.
2. Integrator mode is available for TCRCLK filtering only, see TCRCLK digital filter.
3. These delays also apply to the TCRCLK filter when TCRCF selects eTPU clock/2 as filter clock.

Note

The TCRCLK filter is clocked as if FCSS=0, always dividing eTPU clock /2 using FPSCK, regardless of the FCSS field.

2.14.5.7 Channel timing modes

Channels can work on two different timing schemes, defining the period of channel clocking, tied to T2 and T4 microengine phases, as explained in subsections below. Microengine T2 and T4 phases are shown in Microcycle and I/O timing.

2.14.5.7.1 T2 channel timing

In T2 timing mode the channel event state can only be updated every two eTPU clocks (see Figure 2-55). Pin state, TDLs, MRLs and Capture registers are updated on the microengine's T2 clock phase. MRLE clears also happen on T2, but MRLE setting occurs on T4, together with the Match register write by microcode (see Write Channel Match and UDCM Registers).

Channels work in T2 timing mode when either the T4DIS field in eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2) is 1 or all the following conditions are true:

- The TCR1CS field in eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1) and eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2) is 0.
- The Enhanced Digital Filter is not configured as bypass (see Enhanced Digital Filter - EDF).
- The FCSS field in eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2) is 0 (see Filter clock prescaler).

2.14.5.7.2 T2/T4 channel timing

In T2/T4 timing mode the channel event state can be updated on any eTPU clock (see Figure 2-56). Pin state, TDLs, MRLs, MRLEs, and Capture registers are updated either on microengine's T2 or T4 clock phases. MRLE clears can happen on T2 or T4, but MRLE setting occurs on T4 only, together with the Match register write by microcode (see Write Channel Match and UDCM Registers).

Channels work in T2/T4 timing mode when the T4DIS field in eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2) is 0 and either one the following conditions are true:

- The TCR1CS field in eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1) and eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2) is 1.
- The Enhanced Digital Filter is configured as bypass (see Enhanced Digital Filter - EDF).
- The FCSS field in eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2) (see Filter clock prescaler).

T2/T4 timing mode demands special care on channel microcode that reprograms a match in a thread with matches enabled (entry point ME=1, see the “Entry Points” section). When the match register is written and the MRLA/B flag(s) is(are) cleared in the same instruction, a match can occur after MRLA/B is cleared, together with the set of respective MRLE. In this case, both MRL and MRLE will be set, and a match service request will occur if enabled. However, the match happened on the old match value, not on the new (reprogrammed) one. In order to prevent this ambiguity to the code that services the match, it is advisable to clear the MRLE (besides MRL) together with the

match reprogramming, avoiding the match on the old value to occur while the new match value is being written. The set of the MRLE due to match reprogramming prevails over the MRLE clear, thus allowing the new programmed match to occur in the future.

A similar case occurs when a match is blocked by a transition, the blocking TDL is cleared and the blocked match is reprogrammed at the same instruction: in T2/T4 timing mode: a match can occur on the old match value. The solution is the same as the case above, that is, clear the respective MRLE in the same instruction.

2.14.6 Time bases

Each eTPU engine has two Time Counter Registers, TCR1 and TCR2. They provide 24-bit time bases, shared by all 32 channels. Any channel can use both time bases to:

- Match channel's internal registers MatchA or MatchB;
- Capture time base value to channel's internal registers, CaptureA and/or CaptureB, when a match recognition or an Input transition detection occurs. For more information on channel events refer to Enhanced Channels.

The TCR1 and TCR2 counters are accessible by the microcode for read and write operations. Its current value is used for getting the current time, and the captured values are used for channel relative time calculations of future events. When they are written at the same time they are incremented from any clock source, the written value prevails.

TCR1 with eTPU_TBR_TBCR_ENG*n*[TCR1CS]=0 and TCR2 values are updated in T2 and read in T4 (see Microcycle and I/O timing). TCR1 can also work at full-speed eTPU clock, and so be updated on both T2 and T4, when eTPU_TBR_TBCR_ENG*n*[TCR1CS]=1. Both TCR1 and TCR2 values can be imported from or exported to the STAC bus. When their values are imported (STAC Clients), these registers are written from the STAC bus and can only be read by microcode.

The TCR2 counters between the two Engines are out of phase by 1 eTPU clock, even when Time Bases are shared between them through STAC. It also applies to TCR1 counters if eTPU_TBR_TBCR_ENG*n*[TCR1CS]=0, but they can be in phase otherwise.

2.14.6.1 Timer Count Register 1 - TCR1

TCR1 can be used in the following modes:

- Internally Clocked Mode

- Externally Clocked Mode
- STAC Bus Client Mode

The host program can read TCR1 time base through eTPU Engine 1 Time Base 1 (TCR1) Visibility Register (eTPU_TBR_TB1R_ENG1) and eTPU Engine 2 Time Base 1 (TCR1) Visibility Register (eTPU_TBR_TB1R_ENG2).

The TCR1 bus runs through all the local engine channels. In channels which select TCR1 as MatchA and/or MatchB source, when its value is greater or equal to the programmed match value, a Match A and/or Match B event occurs on that channel. A recognized match event sets its related Match Recognition Latch 1 or 2, and according to the Predefined Channel Modes (PDCM) it may generate a channel service request. For details on eTPU channels see Enhanced Channels.

2.14.6.1.1 Externally Clocked Mode

TCR1 can be driven externally by the TCRCLK input, after the digital filter. The TCR1 clock source is configured by the TCR1CTL bit, as shown in the following figure. For more information on clock source selection, see eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1) and eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2).

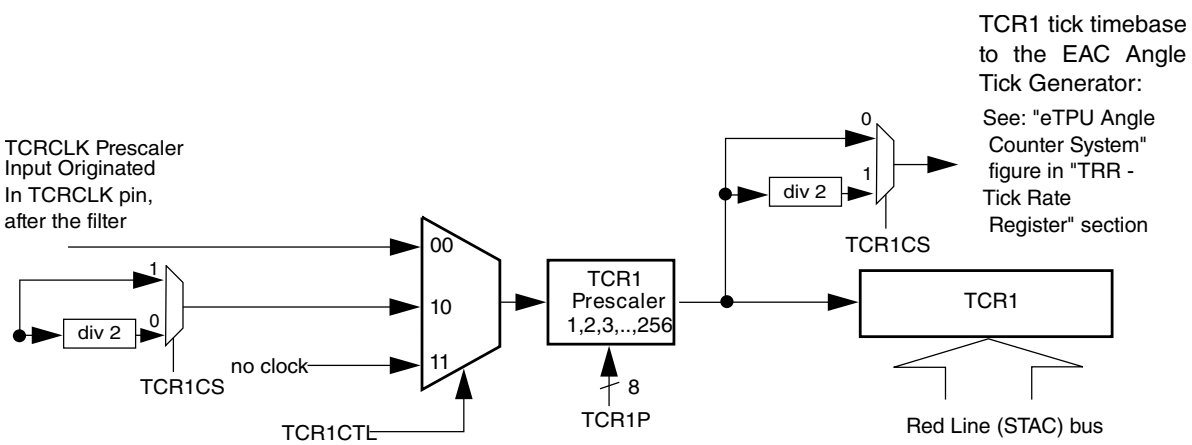


Figure 2-28. TCR1 clock selection

2.14.6.1.2 Internally Clocked Mode

TCR1 can be driven by the eTPU clock or eTPU clock divided by 2, before the prescaler. TCR1CTL can also be used to freeze TCR1 clock independently of TCR2 (unlike GTBE).

2.14.6.1.3 TCR1 clock prescaling

Any clock source selected by TCR1CTL is prescaled by a factor of 1 to 256, selected by the TCR1P field in eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1) and eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2). For more information on prescaler configuration see eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1) and eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2). The TCR1 Prescaler resets when the eTPU GTBE input is negated. After reset, it starts counting up to TCR1P when the eTPU GTBE input is asserted. When TCR1 increments (the eTPU GTBE input = 1), the prescaler starts a new count and the new TCR1P becomes effective. When TCR1 is written by microcode, the prescaler is reloaded with TCR1P and it becomes effective, if the eTPU GTBE input is asserted.

2.14.6.1.4 STAC Bus Client Mode

In this mode the TCR1 register is continuously updated from the STAC bus, and the clock selection and prescaling logic becomes ineffective. It is not writable by the microcode, and when read, it reflects the STAC bus imported value. The use of EAC is not allowed in client mode. This mode is configured through eTPU Engine 1 STAC Configuration Register (eTPU_TBR_STAC_ENG1) and eTPU Engine 2 STAC Configuration Register (eTPU_TBR_STAC_ENG2).

2.14.6.1.5 STAC Bus Server Mode

TCR1 bus can be exported to the STAC bus as a server, providing time information to other peripherals. This mode is configured through eTPU Engine 1 STAC Configuration Register (eTPU_TBR_STAC_ENG1) and eTPU Engine 2 STAC Configuration Register (eTPU_TBR_STAC_ENG2).

2.14.6.2 Timer Count Register 2 - TCR2

The TCR2 is a 24-bit counter which can be used in the following modes:

- Pin Transition Mode: Count the rise, fall or both transitions of TCRCLK signal.
- Angle Clock Mode: Count internal tooth angle in combination with the eTPU Angle Counter (EAC) hardware which implements an Angle PLL, and generates angle information to the channels. This mode is targeted for angle based applications.
- STAC (STAC) Bus Client Mode: TCR2 is driven by an external source (see STAC Bus Client mode).

- Gated Mode: Count with rate derived from the eTPU clock divided by eight. The TCRCLK signal is used to gate this count, enabling pulse accumulator operations.
- Internally Clocked Modes: TCR2 is driven by internal clock, with count rate of eTPU clock divided by eight.

All clock sources pass through a prescaler. In addition, the TCR2 count can be originated from the EAC which is a hardware angle clock and angle counter. Figure 2-29 shows the diagram for TCR2 clock control. When TCR2 is not driven by the EAC or STAC, the eTPU_TBR_TBCR_ENG*n*[TCR2CTL] field selects the clock source, also allowing TCR2 to be frozen independently of TCR1 (see eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1) and eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2)). When in Angle Mode, TCR2CTL selects the TCRCLK edge sensitivity.

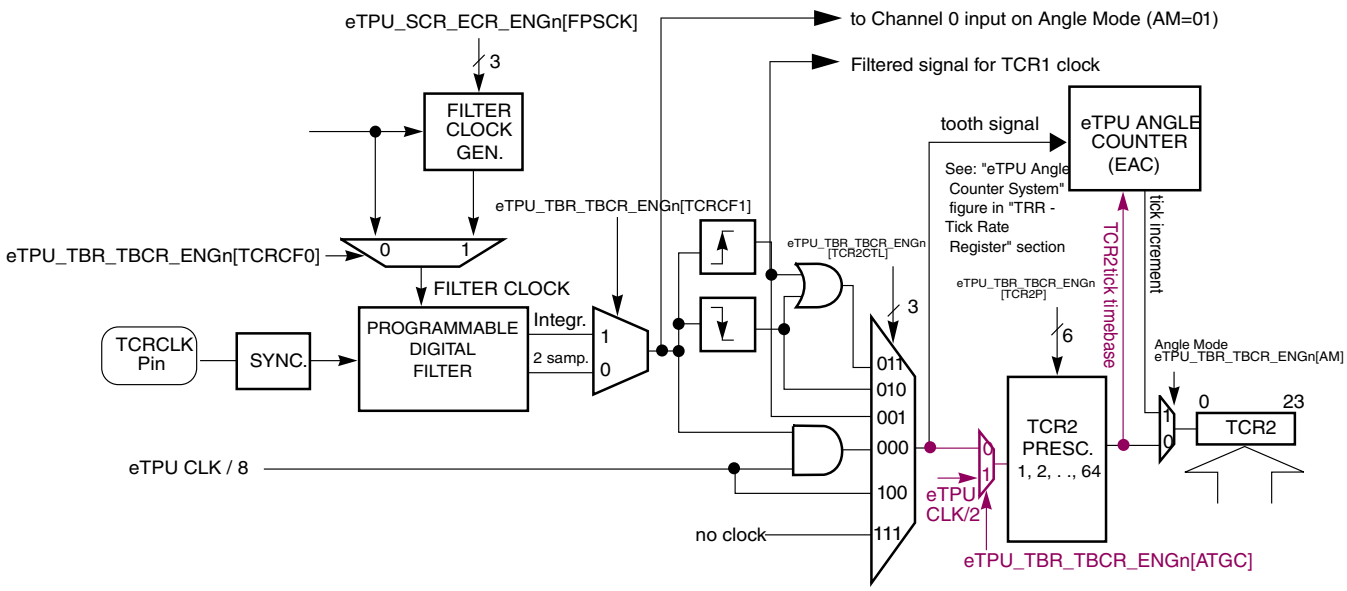


Figure 2-29. TCR2 clock control

The TCRCLK signal input is passed through a synchronizer and a programmable digital filter. In Angle Mode with AM=01, synchronizer and filter are also used in Channel 0, replacing its input synchronizer and filter, to get the same timing in the EAC and Channel 0. The TCRCLK synchronizer is an improved filter that provides best latency while maintaining proper noise filtering (see the field TCRCF in eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1))

The TCR2 bus runs through all the local engine channels. It transitions on clock T2 (see Microcycle and I/O timing). In channels which select TCR2 as MatchA and/or MatchB source, TCR2 value is compared against MatchA and/or MatchB registers. A recognized

match event sets its related Match Recognition Latch 1 or 2, and according to the Predefined Channel Modes (PDCM) it may generate a channel service request. For details on eTPU channels see Enhanced Channels.

The TCR2 counter is accessible by the microcode for read and write operations. Its current value is used for getting the current counter value (representing signal transitions, time or angle), and the captured values are used for channel relative count calculations of future events. The TCR2 value is readable to the host through eTPU Engine 1 Time Base 2 (TCR2) Visibility Register (eTPU_TBR_TB2R_ENG1) and eTPU Engine 2 Time Base 2 (TCR2) Visibility Register (eTPU_TBR_TB2R_ENG2). When the TCR2 bus value is imported from the STAC bus (STAC client mode), TCR2 is not writable by the microcode, and read access from the microcode or from the host reflect the imported TCR2 value.

2.14.6.2.1 TCR2 clock prescaling

Except in Angle Mode, any clock source selected by TCR2CTL is prescaled by a factor of 1 to 64, selected by eTPU_TBR_TBCR_ENG*n*[TCR2P]. For more information on prescaler configuration, see eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1) and eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2). The TCR2 Prescaler resets when the eTPU GTBE input is negated. After reset, it starts counting up to TCR2P when the eTPU GTBE input is asserted. When TCR2 increments (the eTPU GTBE input = 1), the prescaler starts a new count and the new TCR2P becomes effective. When TCR2 is written by microcode, the prescaler is reloaded with TCR2P and it becomes effective, if the eTPU GTBE input is asserted.

The counter that divides the eTPU clock by 8 before the prescaler also resets when the eTPU GTBE input is negated, or when TCR2 is written by microcode.

In Angle Mode the TCR2 prescaler is not used to directly increment TCR2, but can be used, selected by programming eTPU_TBR_TBCR_ENG*n*[ATGC]=1, as a clock source for the EAC generation of the angle ticks which increments TCR2. In this case, the TCR2 prescaler input is eTPU clock/2, independently of TCR2CTL. See Generating the angle ticks for details.

2.14.6.2.2 TCR2 Gated mode

TCR2 Gated mode is selected in the eTPU_TBR_TBCR_ENG*n*[TCR2CTL] field. In this mode the TCRCLK signal enables or disables transfer of the eTPU clock divided by 8 to the TCR2 prescaler. By programming the prescaler, TCR2 can run at rates from eTPU clock divided by eight down to eTPU clock divided by 512, in steps of eight eTPU clock

divisions. For more information, see eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1) and eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2).

2.14.6.2.3 TCR2 Signal Transition modes

These modes are selected when eTPU_TBR_TBCR_ENG*n*[TCR2CTL] is set to rise, fall or "rise-and-fall". In these modes the TCRCLK signal is the TCR2 clock source, and its maximum transition rate depends on the TCRCLK digital filter mode of operation. The TCRCLK digital filter can be programmed to use the eTPU clock divided by two, or use the same filter clock of the channels, controlled by eTPU_TBR_TBCR_ENG*n*[TCRCF]. It contains an up-down counter which operates as a digital integrator, optimizing signal latency in the selected mode and clock rate.

When eTPU clock divided by two is selected, the synchronizer and the digital filter are guaranteed to pass pulses that are wider than four eTPU clocks (two filter clocks). Otherwise the TCRCLK is filtered with the same filter clock as the channel input signals. For details on TCRCLK and channels digital filter control, see eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1), eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2), and Enhanced Digital Filter - EDF.

2.14.6.2.4 STAC Bus Client mode

In this mode the TCR2 register is continuously updated from the STAC bus, and the clock selection and prescaling logic becomes ineffective. It is not write accessible for the microcode, and when read, it reflects the STAC bus imported value. The use of EAC is not allowed in client mode. This mode is configured throughTPU Engine 1 STAC Configuration Register (eTPU_TBR_STAC_ENG1) and eTPU Engine 2 STAC Configuration Register (eTPU_TBR_STAC_ENG2).

2.14.6.2.5 STAC Bus Server mode

When TCR2 bus is exported to the STAC bus as a server, it can provide either time or angle bus to other peripherals, according to its operation mode. This mode is configured through eTPU Engine 1 STAC Configuration Register (eTPU_TBR_STAC_ENG1) and eTPU Engine 2 STAC Configuration Register (eTPU_TBR_STAC_ENG2). To provide sequential update of the STAC clients, the Angle tick rate must not be faster than the STAC programmed update rate. This requirement puts a limitation on the angle clock count rate on high rate mode. In this case the Angle and Angle Fraction accumulator (see Angle Tick Generator and Figure 2-35) are advanced at rate of eTPU clock divided by eight. Therefore, the STAC update rate for the Angle Bus must not be slower than eight eTPU clocks.

2.14.6.2.6 TCR2 bus in Angle Clock mode

In this mode the TCR2 counter operates as part of the eTPU Angle Counter (EAC). The TCR2 bus value reflects this angle representation in which it counts Angle Ticks. Angle Mode is selected when eTPU_TBR_TBCR_ENG*n*[AM]=1.

Note that when TCR2 works in Angle Mode, it does not count directly from the TCR2 clock input which indicates tooth signal transition. Its Angle counter is controlled by the Count Control and High Rate logic (see Count Control and High Rate Logic), which provides the interpolated pin position, and handle cases of missing tooth, acceleration, de-acceleration and mechanical corrections.

The EAC uses the TCRCLK signal to get the tooth transition indications. The eTPU_TBR_TBCR_ENG*n*[TCR2CTL] field has to be set for the appropriate tooth edge detection rise, fall, "rise-and-fall" or none. TCR2 count clock comes from the EAC control and not directly from the physical tooth. This way the EAC control processes the signal transitions and handles missing teeth and flywheel mechanical corrections. Note that when TCR2CTL selects "none" for tooth edge selection, the TCR2 is not necessarily frozen, but can still be incremented by the EAC logic.

In Angle Mode, eTPU channel 0, 1 or 2 operation is combined with the EAC operation. When channel 0 is selected for EAC operation, the TCRCLK digital filter is used both by the EAC and by channel 0 to get full synchronization between the two logics.

The eTPU Angle Counter (EAC) logic runs continuously and updates the TCR2 Angle counter, eliminating the microcode latency in updating the TCR2 value.

2.14.6.3 STAC interface

Both time bases TCR1 and TCR2 can be shared between the Engines and with other blocks in the same MCU. Each one of both eTPU engines can drive their time bases to the STAC (Shared Time and Count) bus, acting as a server, while any other block can capture the value into its resources and behave like a client. For further reference about the STAC bus operation see eTPU Engine 1 STAC Configuration Register (eTPU_TBR_STAC_ENG1) and eTPU Engine 2 STAC Configuration Register (eTPU_TBR_STAC_ENG2).

The eTPU can export to the STAC bus or import from the STAC bus the following internal resources:

- TCR1: Can be exported to or imported from the STAC bus. TCR1 can only be imported from STAC bus when the engine is not in Angle Mode. When TCR1 is imported from the STAC bus, it becomes read-only for the microcode and reflects the imported values. For details, see Timer Count Register 1 - TCR1.
- TCR2: Can be exported to or imported from the STAC bus. TCR2 can only be imported from the STAC bus when engine is not in Angle Mode. When TCR2 is imported from the STAC bus, it becomes read-only for the microcode, and reflects the imported values. When exported to the STAC bus, TCR2 can work in either Angle Mode or as a free running counter associated with the TCRCLK signal. For details, see EAC - eTPU Angle Counter.

Proper configuration of the following bits is necessary to determine what can drive the STAC bus: eTPU_TBR_TBCR_ENG*n*[AM] and eTPU_TBR_STAC_ENG*n*[REN2, RSC2], according to the following table.

Table 2-46. STAC bus and host read sources

AM (eTPU_TBR_TBCR_ENG*n*)	REN2, RSC2 (eTPU_TBR_STAC_ENG*n*)	TCR2 bus source (Host read of eTPU_TBR_TB2R_ENG*n*)	STAC bus driver
00	0× (disabled)	TCR2/Time	×
01, 10 or 11	0× (disabled)	TCR2/Angle	×
00	11 (Server)	TCR2/Time	TCR2/Time
01, 10 or 11	11 (Server)	TCR2/Angle	TCR2/Angle
01, 10 or 11	10 (Client)	not allowed[1]	

1. STAC client configuration in Angle Mode is also not allowed for TCR1.

Note that Angle Mode is not available for STAC bus clients: configuring both at the same time brings unspecified results and must be avoided. When TCR2 is a stand-alone counter or a STAC Bus server, the same value that is driven to the internal TCR2 bus is also exported to the STAC bus (either Time Count or Angle).

STAC bus configuration is provided by the eTPU_TBR_STAC_ENG*n* fields REN1/2 and RSC1/2. REN1/2 enable the STAC interface to interact with the resource (either TCR1 or TCR2 bus). RSC1/2 configure the resource (either TCR1 or TCR2 bus) as Server or Client.

Each time base / angle count resource from each engine receives a unique 4-bit hard-wired address that identifies it as a potential server. This address is used by the STAC Controller to coordinate which resource will drive the bus at a given STAC time-slot. For any time-slot there is a server driving the bus upon selection of the STAC Controller, and there may be a client linked to that server by the eTPU_TBR_STAC_ENG*n* fields bits SRV1/2 on each Engine. When the server address on the Red Line STAC bus matches

the value in SRV1/2, the client will load the STAC information into the appropriate resource. For information on eTPU STAC Bus configuration, see eTPU Engine 1 STAC Configuration Register (eTPU_TBR_STAC_ENG1) and eTPU Engine 2 STAC Configuration Register (eTPU_TBR_STAC_ENG2).

The eTPU does not include a STAC Controller module, which is instantiated once in the system integration.

Note

Setting a timebase as client of itself is not allowed.

2.14.6.4 GTBE - Global Time Base Enable

GTBE enables time bases in both engines, allowing them to be started synchronously. GTBE is divided in two block interface signals: the eTPU GTBE output and the eTPU GTBE input. GTBE bit sets the eTPU GTBE output , and the eTPU GTBE input enables time bases to start. The eTPU GTBE output signal can be used by MCU integration for synchronization between eTPU time bases and time bases from other modules. If GTBE must enable only the eTPU time bases, the eTPU GTBE output is simply connected to the eTPU GTBE input. These two cases are shown in the figure below. Synchronization logic can be as simple as an OR logic gate.

Once the eTPU GTBE input transitions to 1, the Engine 1 Time Bases start 1 eTPU clock earlier than Time Bases in Engine 2, except when TCRCLK is selected as clock source or TCR1 when eTPU_TBR_TBCR_ENG*n*[TCR1CS]=1 . This happens independently of prescaler values as long as they are the same for both engines, because the prescalers also freeze when the eTPU GTBE input = 0. Microcode can always write to TCR1/2 registers, with either value of the eTPU GTBE input.

NOTE

The timebase prescalers are reset when the GTBE input is negated.

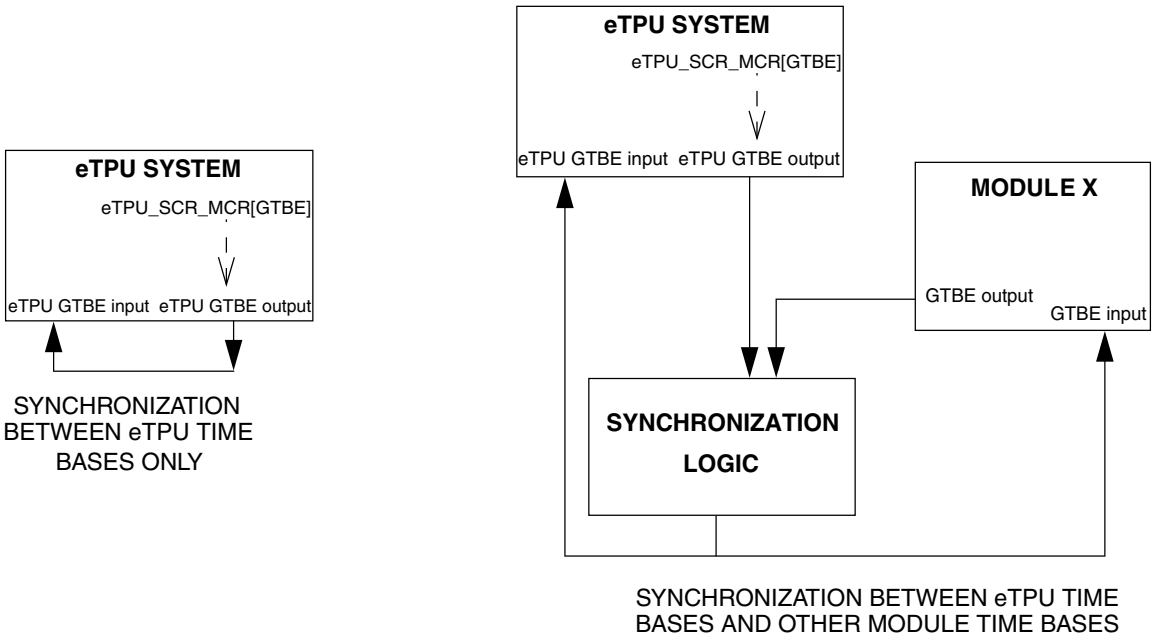


Figure 2-30. Time base synchronization

2.14.6.5 TCRCLK digital filter

The TCRCLK signal has an improved integrating digital filter with a 2-bit up-down counter. The counter counts up to 3 when a high signal level is detected, or down to 0 when a low level is detected. The signal state is updated to one when the counter stops at 3, or zero when the counter stops at 0. The eTPU_TBR_TBCR_ENG*n*[TCRCF] field (see eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1) and eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2)) determines whether the TCRCLK signal input (after a synchronizer) is filtered with the same filter clock as the channel input signals (see Enhanced Digital Filter - EDF) or uses the eTPU clock divided by 2, and also whether the TCRCLK digital filter works in integrator mode or the same two sample mode as the channel filters (see Table 2-30).

The TCRCLK filter delay and prescaling determines the minimum detectable TCRCLK pulse widths and, therefore, its maximum frequency, as shown in Filter clock prescaler and Table 2-45 . The TCRCLK signal delay from the module input to TCR1/TCR2 incrementing or detection in the EAC logic is explained in Input/Output Signal Delays.

2.14.7 EAC - eTPU Angle Counter

2.14.7.1 General

The EAC logic contains a mechanism which follows the flywheel angle, based on the tooth rate. This hardware works in combination with the TCRCLK signal, the TCR2 counter and Channel 0, 1 or 2 (depending on the eTPU_TBR_TBCR_ENG*n*[AM] field) to generate angle information on the TCR2 bus which is passed to all the local engine channels. The EAC helps to implement a digital angle PLL (see Figure 2-34), which combines hardware with microcode processing at channel 0, 1 or 2. The angle measurement is based on history knowledge of the tooth period, for predicting the period of the next tooth. The tooth period is partitioned into a programmable number of **Angle Ticks**. The eTPU application will use the divider in the MAC/Divide unit to calculate an integer and a fraction part of the angle tick such that the full tooth period gets the correct programmed number of angle ticks with no accumulated error.

Each single tooth can be divided in angle ticks, up to 1024. In a 60 teeth flywheel, 128 Angle Ticks per tooth provide resolution of ~0.05 degrees per tick, which meets the accuracy requirement of 0.1 degrees in current automotive applications.

The measurement of one tooth in angle ticks is independent on engine RPM; it is the tooth period itself (and the corresponding tick period) that is re-calculated for each new tooth, based on the difference between the estimated tooth and the actual detection.

For these applications, one of the eTPU channels 0, 1 or 2 is dedicated to service the physical tooth detection. Channel 0 shares the same filtered input as the TCRCLK signal to get the same timing as the EAC. The TCRCLK edge detection is selected by the eTPU_TBR_TBCR_ENG*n*[TCR2CTL] field for the EAC, and by IPACA/B on channel 0, which must be set to detect the same edge(s). When channels 1 or 2 are selected to work with the EAC, IPACA/B is used to select the tooth signal edge detection for both the channel and the EAC, and the tooth signal that feeds the EAC is the same filtered input which feeds the channel.

Channel 0, 1 or 2 generates the signal transition service request, and can also be used for generation of a window filter on this transition, to qualify TCR2 clocks. For this purpose, the selected channel should be configured with double match window filtering mode (refer to Channel Modes). Depending on the channel mode set for the channel, Match A recognition opens the window, and Match B recognition may close it or leave it open. See Angle Logic and Channel Modes for details. Match B also generates a time-out service request. Its input signal transition comes from the tooth. The window can be defined by microcode to open at a predefined point inside the tooth period, and stay open for a desired percentage to the tooth period. The window can be measured in angle or time This method improves the noise immunity by allowing transition detection only on an expected period, a feature which was software responsibility in previous TPU versions.

The EAC supports deceleration, acceleration, last tooth and missing tooth scenarios. The large range of angle ticks per tooth can be used to cover longer tick counts caused by one or more missing teeth, or to provide extra resolution for future application requirements. In case of a missing tooth, the EAC can be configured to insert a dummy tooth or to simply measure a longer tooth.

Figure 2-35 shows the block diagram of the Angle Counter system. TCR1 or TCR2 prescaler is used as a time base which measures the tooth period and is used for partitioning the period to angle ticks.

2.14.7.2 Angle Mode registers

In Angle Mode, the registers described below control eTPU angle operations. They are accessible only by microengine as source and destination registers in microinstructions. When eTPU is not in angle mode (eTPU_TBR_TBCR_ENG*n*[AM] field is negated), all Angle Mode registers can be used as general purpose registers.

2.14.7.2.1 TPR - Tooth Program Register

TPR provides configuration for the Angle Counter circuit. In this register, the microcode can properly adjust the tooth count (controlling last tooth, missing teeth, dummy tooth insertion, halt until tooth detection) and the number of angle ticks per tooth (field TICKS). Note that this register is sampled into a temporary register in the EAC logic when the High Rate Mode is detected (see High Rate Mode (Acceleration) and TPR Buffering), which means that changes to this register may take effect only for the next tooth.

See Count Control and High Rate Logic and to Special Cases of Missing Teeth and Last Tooth until Handling False Tooth Detection for a detailed explanation about the use of this register. The figure below provides a detailed description of TPR register.

Several conflict issues on TPR writes are explained in Special TPR write cases.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LAST	MISSCNT [1:0]		IPH	HOLD	MISS CNT [2]	TICKS									
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2-31. TPR

LAST —Last Tooth Indication

Asserted by microcode and negated when a tooth is detected or inserted via IPH.

Table continues on the next page...

1 = Last Tooth - reset TCR2 Counter at the end of the tooth tick count (after physical tooth or IPH=1) when MISSCNT=0.

0 = Not Last Tooth.

If the tooth is detected or inserted before the tooth tick count completes (going High Rate mode, see High Rate Mode (Acceleration)), LAST resets immediately, but TCR2 resets only when the tooth count completes and MISSCNT=0.

MISSCNT — Missing Tooth Counter

Decremented on each estimated tooth, stops at zero. Used for generation of up to 7 "Dummy Teeth" whenever it holds a non-zero value.

If the tooth is detected or inserted before the missing tooth tick count completes (going High Rate mode, see High Rate Mode (Acceleration)), MISSCNT resets immediately, but missing teeth count continues in High Rate mode (see TPR Buffering).

IPH — Insert Physical Tooth

This bit generates a dummy physical tooth which has the same effect as a real physical tooth, and resets itself subsequently. If EAC is in Halt mode, it switches back to Normal mode[1]. If EAC is in Normal Mode, it switches to High Rate Mode. If Angle Logic is frozen by HOLD=1 (see below), it returns to the state it was at the freezing moment.

1 = Insert dummy physical tooth.

0 = No Operation.

Note: IPH reads as 1 in the next microinstruction after it is asserted, negating subsequently. However, it can be set twice in two consecutive microinstructions to generate two teeth and make the EAC go from Halt to Normal to High Rate Mode.

HOLD — Force EAC Hold

This bit forces the EAC to halt its operation in a special EAC freeze mode until a new physical tooth (a real one or emulated with IPH=1) is detected. Assertion of this bit immediately freezes the EAC in the middle of the tooth period. When a new physical tooth is detected, the bit is automatically negated by the EAC. The HOLD bit can be used for synchronizing the EAC tooth count, in case that a false physical tooth is detected due to noise.

1 = Force EAC to halt until detection of a physical tooth.

0 = Normal Operation.

TICKS — Angle Ticks Number in the Current Tooth

This field defines the number of angle ticks in the current physical tooth. It partitions the tooth period to the required number of angle ticks. The actual number of angle ticks in a tooth is (TICKS+1).

Note: In High Rate mode (see High Rate Mode (Acceleration)), TPR writes are immediately effective only for bits IPH and HOLD. All other fields changes are "buffered" and become effective when EAC leaves High Rate mode. See also Special TPR write cases.

Note: Bits LAST, IPH and HOLD must not be asserted all at once

1. Missing a physical tooth naturally causes EAC to get into Halt mode.

2.14.7.2.2 TCR2 - Timer Counter 2

In Angle Mode, TCR2 (see the following figure) counts angle ticks instead of time.

	23	22	21	20	19	18	17	16
R W	Angle Tick Counter[23:16]							
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Angle Tick Counter[15:0]															
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2-32. TCR2 in Angle Mode

This 24-bit free-running counter is used to generate an accumulated Angle Fraction value. It is updated by the Angle Tick Generator - refer to Angle Tick Generator for more details. Refer to Count Control and High Rate Logic for a detailed explanation about the use of this register in Angle Mode.

TCR2 provides continuous count of the angle in units of angle ticks. The Angle Tick Counter in TCR2 can be reset due to "Last Tooth" microcode indication. TCR2 Prescaling is disregarded in Angle Mode: physical tooth detection is done by EAC regardless of the value set in TCR2P.

2.14.7.2.3 TRR - Tick Rate Register

The exact period of the Angle Tick is programmed in the Tick Rate Register (see the figure below) by microcode. The TRR time unit depends on eTPU_TBR_TBCR_ENG*n*[ATGC] (see Figure 2-35):

- if ATGC=0, the TRR time unit is the period of TCR1 clocks taken as eTPU clocks divided by 2*(TCR1P + 1), even if TCR1CS=1 (see eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1) and eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2)) .
- if ATGC=1, the TRR time unit is given by the eTPU clock divided by 2*(TCR2P+1), using the TCR2 prescaler fed by eTPU clock/2 (independently of TCR1CS).

See Calculating the Angle Tick Period Integer and Fraction for a complete description about the mechanism to calculate the value to be written into this register.

	23	22	21	20	19	18	17	16
R W	INTEGER[14:7]							
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INTEGER[6:0]							FRACTION								
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2-33. TRR

INTEGER — The integer part of TCR1/TCR2 clocks in one Angle Tick.

This number, decremented by one, works as a down-counter preload value. A value of INTEGER=0 represents an integer of 32768. A new value written is reloaded into the counter (becoming effective) when a new tick starts or a tooth is detected or inserted via IPH.

FRACTION — Nine-bit fractional part of TCR1/TCR2 clocks in one angle tick.

The FRACTION value is accumulated in the EAC Fraction Accumulator, and whenever the result overflows (i.e., the accumulated fraction added up to an integer), the Tick Prescaler is halted for one TRR time unit.

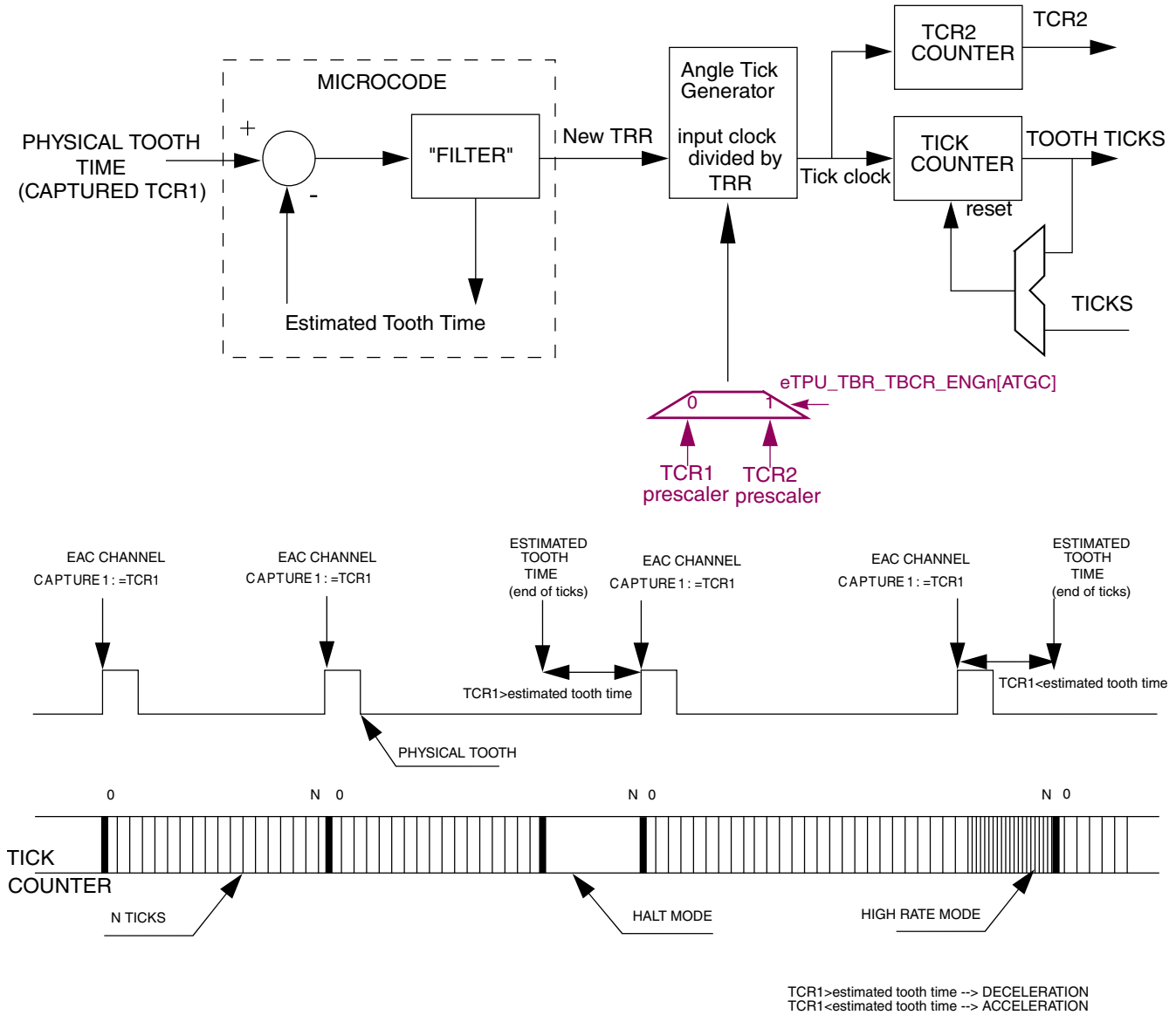


Figure 2-34. EAC "PLL"

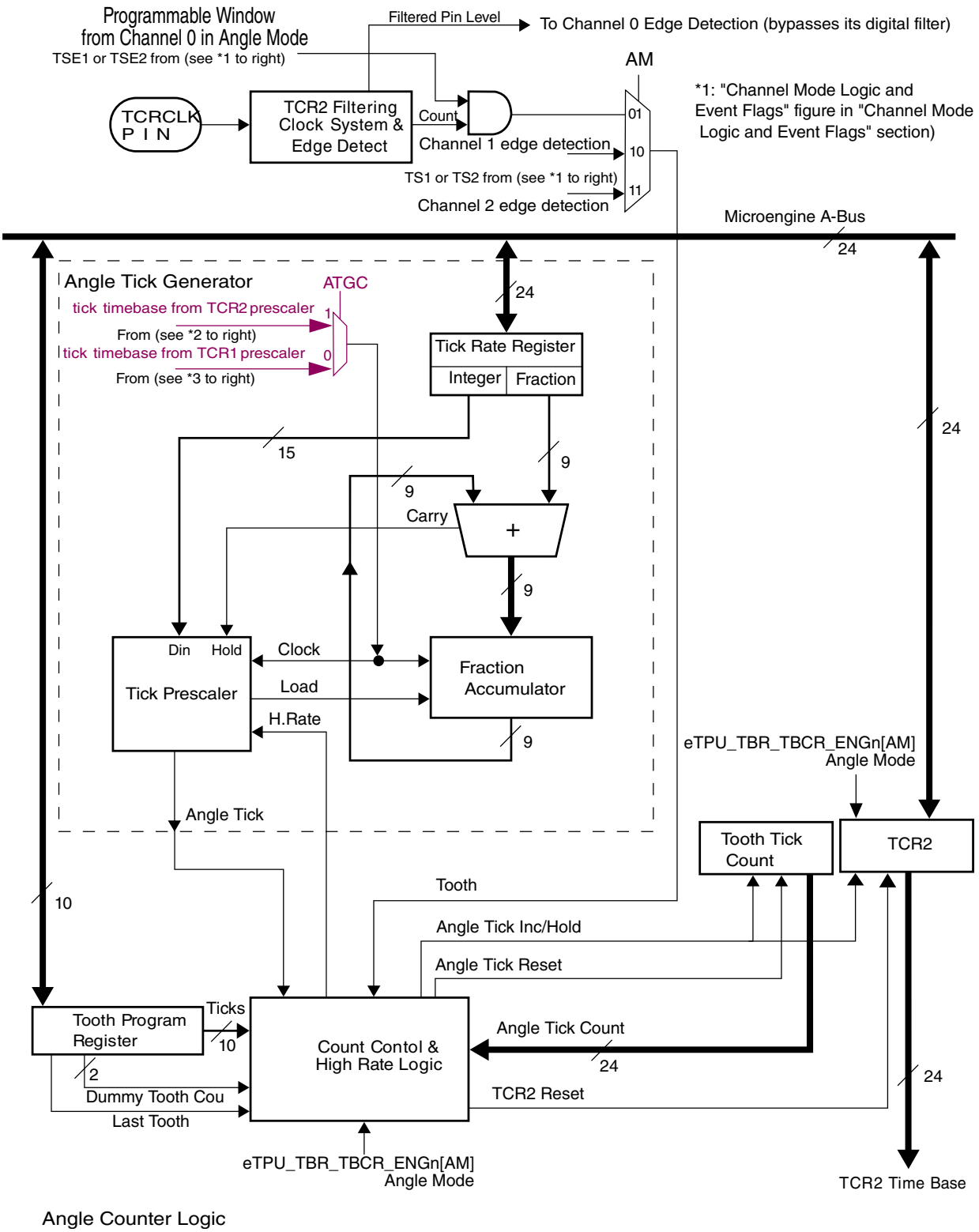


Figure 2-35. eTPU angle counter system

2.14.7.3 Acceleration and Deceleration

Acceleration and deceleration affect the new tooth period relative to the known period of the last tooth. Changes in tooth period may be extreme at very low engine RPM (such as cold start and warm start). The worst case of tooth period changes is caused during missing teeth, since there is more time for changes in angular velocity to be unnoticed by the EAC hardware. For example, on cold start (~20 RPM) there may be extreme acceleration: the ratio between a known tooth period before two missing teeth and the new tooth period after the missing teeth can be very high (up to a factor of 75). Acceleration and deceleration effects from tooth to tooth are less extreme as the engine climbs to high RPM.

In case of deceleration, the estimated tooth period ends before the actual tooth detection arrives. In this case, the EAC hardware waits at the end of the current tooth period, when it is said to be in **Halt mode**, until the real tooth indication is received, then continues with normal operation (**Normal mode**). See Figure 2-34.

In case of acceleration, the actual tooth period is shorter than the estimated tooth period. As a result, a new physical tooth indication arrives before the end of the estimated tooth period. In this case the EAC closes the gap on **High Rate mode** by counting on eTPU clock divided by eight to the end of the tooth, advances to the next tooth, and switches back to normal operation mode. See Figure 2-34.

The reason that the EAC does not jump directly to the next tooth is the need to provide sequential angle count throughout the whole tooth period, for channels or external STAC clients (if TCR2 is a STAC server) which compare angle in "equal" mode. These peripherals must get all the valid angle values in a sequential manner, to avoid missing angle matches.

TCR2 advancing from one tooth to another is a continuous count, and can be optionally reset at the end of the tooth. An estimated tooth is generated after the Tooth Tick Counter reaches the TICKS programmed value.

The EAC works continuously and switches automatically between Normal, Halt and High Rate modes. It relies on the microcode to calculate the estimated tooth period on every tooth, and to update the correct angle tick and tooth parameters in the EAC control registers. On high RPM, tooth period changes are reduced from tooth to tooth, and the EAC may follow the angle with good accuracy for several teeth without microcode intervention.

The EAC handles missing teeth by insertion of "dummy" teeth, or by enlarging the expected tooth period. It is a good practice to locate the flywheel missing teeth in non-critical angles, since a missing tooth may increase the angle measurement error (acceleration and deceleration is detected late).

2.14.7.4 Angle Tick Generator

The Angle Tick Generator is responsible for generating a programmed number of angle ticks in the tooth period. It generates the ticks in an average rate which ensures completion of the correct number of angle counts in the estimated period of the tooth, since the count of one tooth in angle ticks is independent on engine RPM. The main output of the Angle Tick Generator is the tick clock that feeds TCR2 in Angle Mode, as well as the internal Tooth Tick Counter (see Figure 2-35). The Tooth Tick Counter counts ticks within a tooth, from 0 up to TICKS, is controlled by the Angle Tick Generator logic and cannot be accessed by microcode. Refer to Figure 2-36 for a generic presentation of the angle tick count and the measurement of a single tooth period.

2.14.7.4.1 Calculating the Angle Tick Period Integer and Fraction

On each tooth the microcode has to update the exact period of a single angle tick used for counting the internal angle in the tooth. The period of an angle tick is determined by the EAC configuration, see Generating the angle ticks for details.

The microcode can use the eTPU MAC Divider unit (see MAC and Divide Unit (MDU)) to divide the tooth period by the number of angle ticks per tooth, which is stored in the TICKS field of TPR (refer to Angle Mode registers). This division yields the integer part of the angle tick period and the remainder. Dividing again the remainder shifted left nine positions, by the number of angle ticks per tooth translates the remainder to a 9-bit fraction. The microcode concatenates the 15-bit integer and the 9-bit fraction to a 24-bit value and writes it to TRR. The new rate is effective immediately after the next Angle Tick is generated by the Angle Tick Generator[28].

For high RPM, note that shifting the tooth period value nine positions to the left prior to the first divide operation would calculate, in one operation, the integer and the fraction. For example: On 60 teeth flywheel running at 1000 RPM, tooth period is 1 msec. If TCR1/TCR2 counts @25 MHz, it counts 25,000 times in a tooth, which can be represented by 15 bits. Therefore the tooth period can be shifted nine positions to the left prior to divide operation, and be represented with 24 bits.

Using shift left nine positions and one divide operation would get the result in MACL register (in MDU) which holds the integer and nine bits of the fraction:

```
Angle_Tick_Rate {Integer[14:0], Fraction[8:0]} = (ToothPeriod <<9) / Ticks
TRR = Angle_Tick_Rate {Integer[14:0], Fraction[8:0]}
```

28. In High-rate mode, the tick keeps being updated at the rate of eTPU clock/8 until it goes back to Normal mode, when the new TRR value is used.

NOTE

In the pseudocode above, the tooth period (ToothPeriod) is not, in general, the value of estimated tooth time. It is obtained by microcode by subtracting TCR1 values between two teeth detections, converted to TCR2 periods if ATGC=1. Its comparison with the estimated tooth time indicates acceleration (if minor) or deceleration (if greater) to the microcode.

On low RPM the initial tooth period, measured in TCR1/TCR2 counts, may be too big to be shifted nine positions to the left. For lower RPM (for example 500 RPM) the tooth period cannot be represented by 15 bits, and shifting it nine positions to the left would lose the MSB. In this case, two divide operations are required as follows: first divide the Tooth Period by the number of TICKS - the integer is stored in MACL and remainder in MACH. MACL is saved in another register. MACH is shifted 9 positions to the left and divided again by TICKS. In parallel with the second divide, the register which saved the original MACL is shifted left 9 positions. After the divide MACL contains the 9 bits fraction and the other register contains the 15-bit integer, shifted left nine times. The logical OR of the two registers is written to the TRR:

```
Angle_Tick_Rate {Integer[14:0]} = (ToothPeriod) / Ticks
Remainder[9:0]                  = ToothPeriod) mod Ticks
Angle_Tick_Rate {Fraction[8:0]} = (Remainder[9:0] << 9) / Ticks
TRR = Angle_Tick_Rate {Integer[14:0], Fraction[8:0]} = (Integer[14:0] << 9) | Fraction[8:0]
```

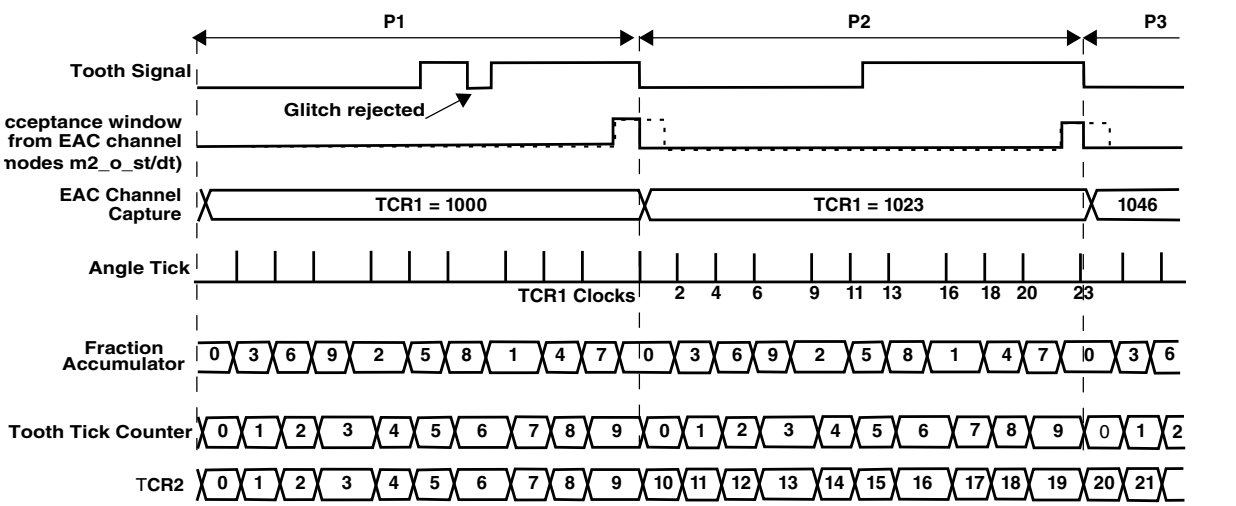


Figure 2-36. Angle Ticks Generation

2.14.7.4.2 Generating the angle ticks

The integer part of TRR is preloaded to a prescaler, which counts in time intervals as follows (see Figure 2-35):

- If eTPU_TBR_TBCR_ENG*n*[ATGC]=0, one unit of TCR1 clock period (if TCR1CS=0) or twice the TCR1 clock period (if TCR1CS=1)[29], that is, eTPU clock period*2*(TCR1P+1) (independently of TCR1CS).
- If ATGC=1, two units of the eTPU clock period multiplied by the TCR2 prescaler factor, that is, eTPU clock period*2*(TCR2P+1).

When the down counter reaches zero, it generates an angle tick pulse to the Angle Counter Logic and a Load pulse to the Fraction Accumulator. It is then preloaded with most updated TRR integer part. Due to the Load pulse, the 9-bit fraction is accumulated in a 9-bit Fraction Accumulator. If a fraction overflow condition occurs (the 9-bit adder asserts carry out), the accumulator saves the lower 9 bits of the addition result, which is the remaining fractional part. The carry out bit indicates an accumulated integer "one" which means that the angle tick is early by one input clock. It halts the prescaler operation for one input clock to compensate the accumulated error generated by the integer prescaler. As a result, the average angle tick period takes into account both the integer and the fraction parts. The accuracy depends on the bit count of the fraction. Using 9-bit fraction part while the width of the field TICKS in register TPR is 10 bits provides accuracy of two LSB on a full scale (TICKS=1023) or one LSB on lower scale (TICKS<=511).

When the Tick Prescaler gets High Rate mode indication from the Angle Counter Logic, it generates angle ticks at a rate of eTPU clock divided by eight. In this case it does not generate Load pulses to the Fraction Accumulator, ignores its "hold" input and preloads internally to a fixed period of eight eTPU clocks. When High Rate mode is entered, the prescaler is preloaded to a period of eight eTPU clocks before its first angle tick generation, ensuring separation of at least eight eTPU clocks between the last Normal mode angle tick and the first High Rate mode angle tick. The fraction accumulator resets when the tick count advances to the next tooth, or when TRR is written by the microcode.

2.14.7.5 Count Control and High Rate Logic

The Count Control and High Rate Logic controls TCR2 operation in Angle Mode, using the angle ticks generated by the Angle Tick Generator. Count Control logic is responsible for advancing, holding and resetting the TCR2 and Tooth Tick Counter in the proper timing, such that the TCR2 time base will reflect the correct estimated angle. This logic also includes the Tooth Program Register (TPR, see Angle Mode registers for more information).

29. The EAC logic overall runs at a rate of eTPU clock divided by 2, therefore the tick generator takes the halved TCR1 clock when TCR1CS=1 and ATGC=0.

The Count Control and High Rate Logic handles deceleration, acceleration, missing teeth and last tooth. On High Rate (acceleration) it ensures that the angle bus scans all valid angle values in a rate which can be traced by the STAC bus. This operation enables external STAC clients (if TCR2 is a STAC server) or channels working in "equal-only" comparator mode to match the TCR2 exported angle information in "equal" mode, in an exact match.

Because the eTPU channels are capable of capturing either TCR1 or TCR2 due to signal transition, the microcode can get either the angle or time of the physical pin transition. Since the EAC channel (0, 1 or 2) is connected to the physical tooth, the microcode can get the EAC error in angle domain (tooth appears at the wrong angle) or time domain (physical tooth captured time into the EAC channel, relative to the estimated tooth time). Note that in angle mode, the transition detect logic of the channel 0, if selected as the EAC channel, is fed from the digital filter of the TCRCLK signal, and not from the channel 0 internal digital filter. This ensures synchronous operation of channel 0 and the EAC hardware.

Another feature of the eTPU channel, when working in Single Match And Single Transition Enhanced Mode (refer to Single Match Enhanced Mode (sm_st_e)), is capturing a single time base due to signal transition before and after the digital filter. This option allows subtracting the digital filter delay to get accurate signal transition timing on the channel. This way, the TCRCLK signal may be programmed with a slow and reliable digital filter, and get accurate time measurement of the digital filter delay.

To assert the end of the estimated tooth period the Count Control and High Rate logic compares the TICKS field in TPR (refer to Angle Mode registers) with the current value of the tooth tick counter. When the Tooth Tick value is greater or equal to TICKS, it determines the end of the estimated tooth period. On acceleration this event occurs during High Rate mode operation, after the arrival of a physical tooth. In deceleration, this event occurs during Normal Mode, before the arrival of a physical tooth. On constant angular velocity, this event appears together with the arrival of a physical tooth.

The following sections describe the operation of the Counter Control and High Rate logic.

2.14.7.5.1 Normal Mode

In Normal mode the Counter Control logic advances TCR2 and the Tooth Tick counter as if the engine has a constant speed during the tooth period. It receives the angle ticks from the Angle Tick Generator in an average rate which is determined by the Tooth Rate Register (TRR). This is the reset mode.

When the Tooth Tick Counter is about to reach the last value effectively stored in TPR field TICKS plus one, the hardware detects the end of the estimated tooth period. If the physical tooth and the estimated tooth arrive at the same time the EAC stays in Normal mode, the Tooth Tick Counter is reset, and TCR2 is incremented (depending on TPR bits LAST and MISSCNT). If the physical tooth and the estimated tooth do not arrive at the same time, either acceleration or deceleration is detected, and the EAC switches to the proper mode. See the figure below for a detailed diagram of Normal Mode behavior.

The microcode which services the EAC channel physical tooth transition may update TRR according to various conditions to give the best estimation of the current tooth period, according to the previous tooth period and other engine parameters.

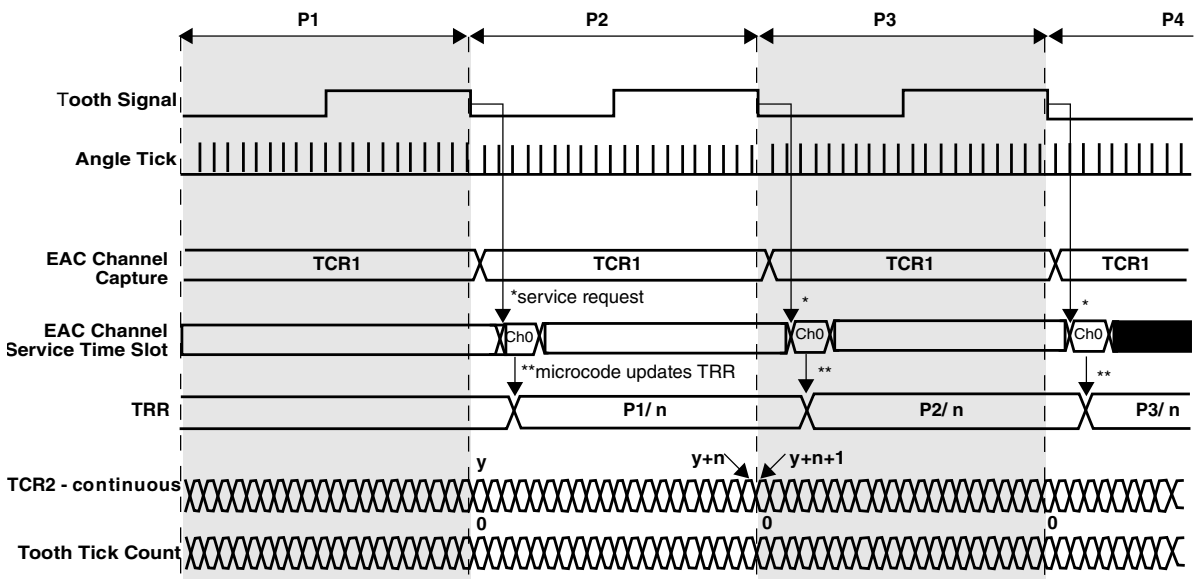


Figure 2-37. Normal Mode

2.14.7.5.2 Halt Mode (Deceleration)

In case of deceleration, the Tooth Tick Counter reaches the TICKS value before the arrival of the next tooth. The Count Control logic does not reset neither advances TCR2 and Tooth Tick Counters. The Count Control logic halts TCR2 and Tooth Tick Counters at the end of the tooth, waiting for the physical tooth to arrive.

When the physical tooth is detected the EAC switches back to Normal Mode and releases TCR2 to count the angle ticks of the new tooth, also resetting the Tooth Tick Counter. Only then TCR2 may wrap to 0, if TPR bit LAST is asserted. See the figure below for a detailed diagram of Halt Mode behavior.

The microcode service caused by the physical tooth determines the deceleration, calculates the new tooth period and Angle Tick period and updates TRR. This operation slows the angle tick rate generated by the Angle Tick Generator on-the-fly, to the rate required for the new tooth period.

Since the microcode service is initiated by the physical tooth edge, microcode latency may introduce a small angle error caused by using the TRR value of the previous tooth at the beginning of the current tooth. On high RPM, deceleration is relatively small but the microcode latency may take a significant percentage of the tooth period. On low RPM microcode service latency takes little percentage of the tooth period, but there may be cases of extreme acceleration and deceleration. The microcode latency can be calculated knowing TCR1 value during the service time, and TCR1 value captured in the EAC channel due to the physical tooth pin transition. The duration of the Halt mode is obtained using the estimated tooth time.

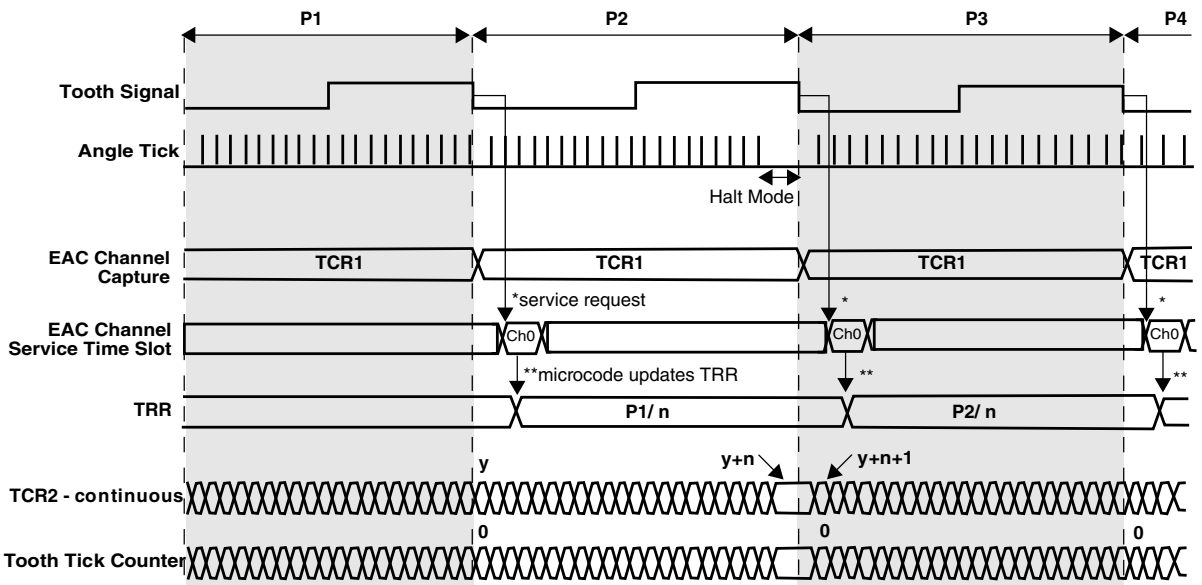


Figure 2-38. Halt Mode - Deceleration

2.14.7.5.3 High Rate Mode (Acceleration)

In case of acceleration, the next tooth arrives before the Tooth Tick Counter reaches the TICKS value. In this case the High Rate logic is responsible for closing the gap and advancing the tooth on the correct timing. The High Rate mode operates as follows:

- When the acceleration is detected (physical tooth arrives before the Tooth Tick Counter reaches the TICKS value), the Count Control and High Rate logic switches to High Rate mode in which both the Tooth Tick Counter and TCR2 count at rate of

eTPU clock divided by eight, until the Tooth Tick Counter reaches the current TICKS value. To ensure correct operation, the TICKS value is sampled in the logic at the beginning of the mode.

- At this point, which represents the estimated tooth edge, the logic resets the Tooth Tick Counter and advances TCR2 (or resets it if LAST is asserted and MISSCNT = 0).
- The control logic switches back to Normal Mode, using the most updated TRR value as input to the Angle Tick Generator. The logic samples the updated TICKS value for the tooth estimation, last tooth indication and number of missing teeth from TPR.

In High Rate mode the angle ticks are provided at high speed until the end of the current tooth. This operation is required to scan all the valid angle values of the current tooth, in a rate which is not too high for the STAC bus continuous update, but much higher than the rate dictated by TRR.

EAC channel microcode, which services the physical tooth transition detection, can start its service either before High Rate mode operation is complete (the Tooth Tick Counter has not reached the TICKS value) or after the EAC switched back to Normal mode. Any physical teeth received while the EAC is in High Rate Mode does not alter the immediate EAC state, but it is still detected by the EAC channel logic and can, therefore, alter future EAC behavior (for instance, closing the tooth detection window - see Angle Logic and Channel Modes).

At the beginning of High Rate mode operation, the TPR value is preloaded into a temporary register in the Counter Control Logic, used for scanning all the valid values to the end of the current tooth, with its appropriate LAST and MISSCNT attributes. While the EAC is in High Rate mode operation, the effect of microcode update of TPR fields LAST, MISSCNT and TICKS is delayed to the next estimated tooth, after the High Rate mode operation is complete (see Special TPR write cases). This is because the current physical tooth represents the next estimated tooth. If the microcode updates this field after High Rate mode operation is complete, the current physical tooth and estimated tooth are the same, and the effect is immediate. Either in High-Rate mode or not, the value read by microengine is the same written, even if not yet effective, until the EAC resets LAST and/or IPH, or decrements MISSCNT. Typically the microcode service may occur during the High Rate mode on extreme acceleration situation at low RPM. Therefore, the microcode operations are always related to the real physical tooth. From the above it can be seen that the microcode updates of the TICKS field in TPR affect the end time of the current physical tooth. For correct operation, this field should be updated before the Tooth Tick Counter has reached either the old or the new TICKS value.

During High Rate mode operation, TRR is ignored and the Angle Tick Generator uses eTPU clock divided by eight. Therefore, the TRR update by microcode will take effect only after the EAC switches back to Normal mode. If microcode service occurs after the Tooth Tick Counter has been reset, the EAC is already back in Normal mode, and some angle ticks may have been counted at the rate of the previous tooth. In this case the new TRR value will have immediate effect on the angle tick period, and the microcode should take into consideration the delay from the physical tooth to the estimated tooth in calculation of the next tooth period. See the figure below for a detailed diagram of High Rate Mode behavior.

An angle error may be introduced by the duration of the High Rate mode. Also, the scheduler latency may introduce a small accumulated error by using TRR value of the previous estimated tooth at the beginning of the current tooth. After the estimated tooth has advanced, the duration of the High Rate mode operation is the actual delay from the physical tooth edge to the estimated tooth edge. This delay can be obtained by comparing the estimated tooth time with the EAC channel capture register which captured TCR1 on the physical pin transition.

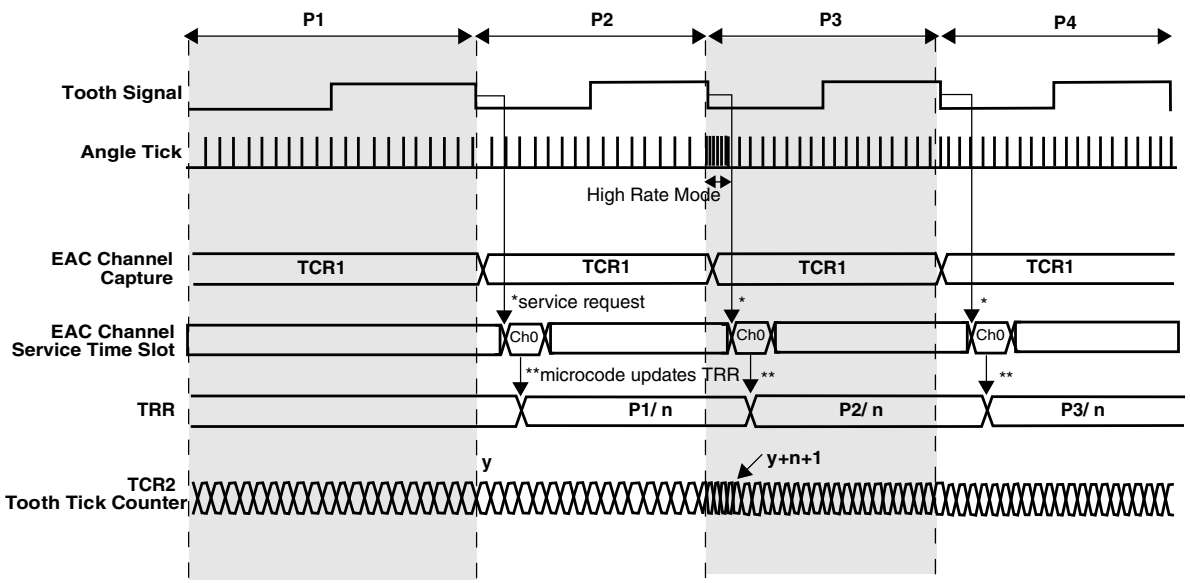


Figure 2-39. High Rate Mode - Acceleration

2.14.7.6 Special Cases of Missing Teeth and Last Tooth

The EAC handles cases of up to three missing teeth and the last tooth in the engine cycle. The following paragraphs describe these functions.

2.14.7.6.1 Handling the Last Tooth

The microcode can set the TCR2 counter to work in engine periods (wrap-around count) or continuous angle measurement.

For periodic operation, during the last engine cycle tooth the EAC microcode has to set the LAST flag in TPR. As a result, when the tooth period is ended, the Counter Control Logic generates a reset command to both the Tooth Tick Counter and TCR2 instead of an advance command. The operation resets the TCR2 based angle count, indicating a new period of the engine cycle. This implementation provides an engine cycle based periodic angle measurement.

2.14.7.6.2 Handling Missing Teeth

The EAC can handle up to three missing teeth in two ways:

- Count the angle ticks relative to the last physical tooth. The microcode should update the TPR TICKS field to the number of angle ticks included in two, three or four teeth, according to the flywheel type (one, two or three missing teeth). EAC hardware works in its regular manner.
- Insert a "dummy" tooth instead of the missing tooth, at the estimated point in time. After the "dummy" tooth, the Angle Tick Counter is incremented as if there was a physical tooth. A "dummy" tooth can be inserted only during Normal or High Rate operation modes. The microcode inserts "dummy" teeth by writing to the MISSCNT field in TPR.

In the first option the missing tooth is not counted on the angle measurement. For example, a flywheel with 59 physical teeth and one missing tooth can be considered as 58 identical teeth numbered (0-57) and tooth number 58 has a double number of angle TICKS. In this case a 720 degrees engine cycle has 118 teeth. TCR2 reflects the real angle, since it counts angle ticks continuously.

In the second option, the missing teeth are counted as "regular" teeth by automatic insertion of "dummy" teeth. The microcode has to write a non-zero value to the MISSCNT field in TPR. This field is a 3-bit down counter which affects the operation of the Counter Control logic.

For example, a flywheel with 59 physical teeth (0-58) and one missing tooth (59) can be considered as 60 teeth numbered (0-59), all having the same number of angle ticks. The microcode has to write "001" to the MISSCNT bits during the period of tooth number 58 to indicate that next tooth (59) is missing.

When the Tooth Tick Counter reaches the TICKS value, TCR2 is incremented as if a physical tooth has been detected. In addition, the MISSCNT value initializes a "dummy tooth counter" which is decremented to indicate the number of left "dummy teeth" which still need to be generated. Because a dummy tooth was counted, EAC does not enter Halt Mode and Tooth Tick Counter continues incrementing in the absence of a physical tooth detection.

In case of extreme acceleration on very low RPM (cold start) there can be a situation that the first physical tooth after one or two missing teeth appears even before the "dummy" tooth is generated. Due to the acceleration the EAC switches to High Rate mode in order to run through all the valid angle values, including the dummy teeth. When the Tooth Tick Counter reaches the TICKS value on High Rate mode, and the "dummy tooth" down counter is not zero, the generated "dummy tooth" advances to the next tooth and decrements the "dummy tooth" counter, but does not switch the EAC back to Normal mode. The last "dummy tooth" decrements the counter to zero, indicating that no more dummy teeth are to be inserted, and the next tooth is an estimated physical tooth. The EAC continues at High Rate mode until the Tooth Tick Counter reaches the TICKS value again, then advances to the next tooth while switching back to Normal mode. When in High Rate mode, the TPR register does not reflect the MISSCNT downcounting, see TPR Buffering for details.

MISSCNT can be rewritten before it reaches 0, allowing it to count more than 7 missing teeth, as long as no physical tooth arrives between the first MISSCNT write and the rewrite.

2.14.7.6.3 Combining Missing Teeth and Last Tooth

The Last Tooth indication takes effect when there are no more missing teeth to be generated, i.e the "dummy tooth" counter value is zero. If, for example, the microcode sets the missing teeth counter to "10" (two missing teeth) and sets the LAST flag, the first and the second dummy teeth will increment TCR2, and the third estimated tooth, which correlates with the physical tooth (the first of the next cycle), will reset TCR2, because LAST was set. This scheme enables the microcode to define one or more missing teeth to be replaced by "dummy tooth" insertion, and the end of the engine cycle in one service request. It is assumed that the two missing teeth must come together in the same engine cycle, and not split between two engine cycles (either the missing teeth are both last in an engine cycle or both not last, but not last in one engine cycle and first in the next).

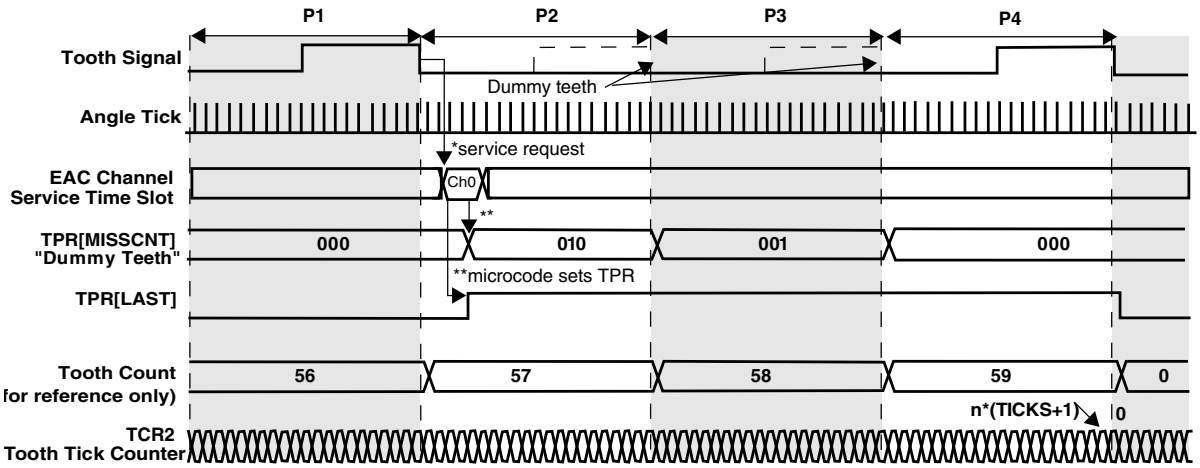


Figure 2-40. Missing Teeth and Last Tooth Combination

2.14.7.7 Handling Mechanical Tooth Correction

The EAC can handle tooth edge detection errors caused by flywheel mechanical errors. The eTPU application can hold a vector of tooth mechanical errors with one entry per tooth. This error can be measured in angle ticks which are independent of engine RPM. The TRR can be updated to the fixed period of any tooth, including its mechanical error.

Because TCR2 counts continuously, without being reset, the mechanical correction is transparent. Though the tooth has its own programmed TICKS value, TCR2 simply counts angle ticks, disregarding the boundary between two adjacent teeth.

2.14.7.8 Handling Mis-detected Tooth

When a physical tooth signal is missed by the engine sensor, the EAC may get into Halt mode at the end of the estimated tooth period, expecting the physical arrival. In this case, a Match timeout event of EAC channel will call service which detects extreme deceleration. The microcode can assert the IPH bit in TPR, to force the detection of the missed physical tooth. It can also calculate the accumulated angle bus error, and fix the next estimated tooth period, to close the gap.

2.14.7.9 Handling False Tooth Detection

Most of the false tooth detection, caused by noises on the engine tooth sensor, can be eliminated by the window blanking filtering, timed by EAC channel match recognitions. The EAC also provides means of fixing false detection of an additional tooth which passed the window filter. When such an event occurs, the EAC switched to High Rate

mode (advancing to the next tooth) and when the next physical tooth arrives, an extreme acceleration is detected: the EAC sees the remaining portion of the current tooth period as another tooth period. The microcode can detect the situation when the acceleration in not realistic, or when immediately after the detection of this extreme acceleration, the following tooth indicates extreme deceleration back to the original RPM.

When the microcode detects such a case, the Tooth Counter has been advanced by mistake to the next tooth. The microcode can set the HOLD bit in the TPR, forcing the EAC to freeze and wait for the next physical tooth to close the gap. When the next physical tooth arrives, HOLD is automatically negated and the EAC proceeds from that point to the remaining portion of the tooth period, in the same mode it was when HOLD bit was asserted.

2.14.7.10 Angle Logic and Channel Modes

The TCRCLK transition detection is qualified by a signal that comes from channel 0 (see Figure 2-35) and depends on the particular mode (PDCM) programmed for that channel. This configures a window for TCRCLK detection for the angle logic which is the same (except on High Rate mode, see High Rate Mode (Acceleration)) used to set the TDLA flag on single transition modes, and TDLB on double transition modes (see signals TSE1,TSE2 in Figure 2-18). The same applies when channels 1 or 2 are used to control EAC (see signals TS1, TS2 in Figure 2-18) . As a consequence, the window depends on the channel mode as follows:

- on all modes, the window closes upon a tooth edge detection: TDLA asserted on single transition modes, TDLB asserted on double transition modes.
- on mode m2_st: the window opens on Match A (which enables Transition A) and does not close with Match B. If Match B comes before Match A, it blocks Match A and, hence, Transition A.
- on mode m2_o_st: the window opens on Match A (which enables Transition A) and closes on Match B. Match B is enabled by Match A, so it cannot come before.
- on all other Single Transition modes, the window is "always open", independently of matches.
- on mode m2_dt: the window opens on Transition A, which is enabled by Match A. The window does not close with Match B, but if it comes before Match A the later gets blocked and, hence, blocks Transitions. Match A is also a condition for the window, so the microcode closes it by clearing MRLA.

- on mode m2_o_dt: the window opens on Transition A, which is enabled by Match A. The window closes on Match B, which is enabled by Match A. Match A is also a condition for the window, so the microcode closes it by clearing MRLA.
- on all other Double Transition modes, the window opens on Transition A.

2.14.7.11 Restarting angle logic

It is not recommended to change the eTPU_TBR_TBCR_ENG*n*[AM] field while GTBE=1. However, if the angle logic must be restarted without interfering with the timebase count running on TCR1, the procedure below must be followed:

1. Program eTPU_TBR_TBCR_ENG*n*[AM]=00 and TCR2CTL=111 at once. That prevents TCR2 from incrementing while the angle logic is disabled. The angle logic state machine resets to Normal mode and the tick prescaler to the initial count with AM=00, but not the microengine registers TPR and TRR.
2. Start a thread to reconfigure the EAC. The thread must set the EAC controlling channel (0, 1 or 2) flags in a state, depending on the channel mode, that lets the channel tooth detection window open (see Angle Logic and Channel Modes). It can optionally write TCR2 with an angle preset value equivalent to the first tooth expected after restart. The thread must also set TPR bit HOLD=1. The TPR bit IPH must be 0.
3. After the thread is finished, program eTPU_TBR_TBCR_ENG*n*[AM]=01, 10 or 11, and TCR2CTL according to the desired tooth edge selection if AM=01.

The first tooth detected after this procedure restarts the TCR2 counting, unfreezing the Angle Mode logic into normal mode.

2.14.7.12 Special TPR write cases

This section describes how simultaneous modification of TPR fields are resolved, and how the effect of TPR writes depend on the EAC mode.

2.14.7.12.1 TPR Buffering

In High Rate mode (see High Rate Mode (Acceleration)), TPR writes are immediately effective only for bits IPH and HOLD. Writes to all other fields are "buffered" and become effective when EAC leaves High Rate mode. However, if TPR is written a

second time right after IPH is asserted in Normal mode, this second write behaves as if EAC is still in Normal mode. Only in the next microcycle (after execution of a nop, for instance) the TPR writes are buffered, acknowledging High Rate mode.

MISSCNT and LAST can be written any value during High Rate mode, and the value that prevails for the next tooth is the one sampled when EAC goes back to Normal mode (or the value written in Normal or Halt mode thereafter). If MISSCNT and/or LAST are not zero when High Rate mode begins, they are sampled into the internal EAC logic and are effective while High Rate lasts (missing teeth count continues, and TCR2 is reset at the end of High Rate if LAST=1). However, their values in TPR are reset when High Rate mode starts. After that and until the end of High Rate mode, the value read by microcode is the same written. This behavior prevents read-modify-writes to TPR from unwillingly rewriting LAST or MISSCNT.

2.14.7.12.2 IPH and LAST

If both IPH and LAST are asserted in the same microinstruction with HOLD=0, the EAC acts as if LAST was set first and then IPH right after, so that:

- in Normal mode, it goes to High Rate with LAST=1;
- in Halt mode, it goes to Normal Mode resetting LAST (and TCR2);
- in High Rate mode, IPH is ignored and LAST becomes effective in the next tooth (physical or inserted) after it goes back to Normal mode.

2.14.7.12.3 IPH and TICKS

Because of different results depending on the EAC mode at the time of TPR write, it is not advisable to write 1 to IPH and change TICKS at the same microinstruction. A consistent behavior is obtained if IPH is written first and TICKS on the second microinstruction after (for instance with a nop between them), making the new TICKS value valid for the next tooth regardless of the mode.

The mode-dependent behavior is (assuming HOLD=0):

- in Normal mode, the new TICKS value becomes valid before EAC goes High Rate due to the IPH;
- in Halt mode, the EAC goes to Normal mode, and new TICKS is valid for the next tooth;
- in High Rate mode, the new TICKS value is effective when EAC leaves High Rate mode, and IPH is ignored;

2.14.7.12.4 IPH and MISSCNT

If both IPH and MISSCNT are written non-zero values with HOLD=0:

- in Normal mode, at the next microcycle the EAC goes to High Rate mode, the MISSCNT field in TPR goes to 0, and the missing teeth are counted in High Rate mode.
- in Halt mode, the EAC goes to Normal mode for one microcycle and then, yet another microcycle later, goes to High Rate mode, counting the missing teeth. The TPR fields IPH and MISSCNT are zeroed on the transition from Normal to High Rate mode.
- in High Rate mode, IPH is ignored (resetting at the next microcycle) and MISSCNT is buffered (see TPR Buffering).

2.14.7.12.5 IPH and HOLD

If IPH and HOLD are asserted at once, IPH cancels the HOLD and both reset. The EAC is not frozen, regardless of the EAC mode (High Rate, Normal or Halt).

2.14.7.12.6 LAST and HOLD

If LAST and HOLD are written 1 at once, LAST asserts and EAC is frozen. Afterwards, when a physical tooth is detected or IPH is asserted, the EAC is unfrozen in the same state it was before, and LAST is kept asserted.

2.14.8 Microengine

Each eTPU engine has a microengine that fetches, decodes and executes microinstructions. The Microengine only works when there are service requests to be attended, otherwise it turns to idle state, controlled by Hardware Scheduler (see Scheduler).

Microcode is stored in Shared Code Memory (SCM) that is 32-bit wide. Microengine can access SDM using a different bus from the one used to accesses code memory, so that code and data can be accessed at the same time (Harvard Architecture).

Some of eTPU functionality can only be made through the microengine, like configuring channels and interrupting host. Microengine gives eTPU a high degree of flexibility, since any desired treatment for channel's events can be implemented; however, that

flexibility comes at the cost of channel service's latency. Latency is worsened when channels from a same eTPU engine contend for microengine service. In the figure below, a block diagram of microengine architecture is shown.

Follows a summary of Microengine features:

- P, DIOB, A, B, C, D, SR, RAR, LINK, CHAN, MACL, MACH, ERTA, ERTB, TCR1, TCR2, TPR, TRR registers are accessible by microcode.
- 24-bit ALU and Post-ALU shifter performs basic arithmetic and logical operations described in ALU and Post-ALU Shifter.
- MDU (MAC/Divide Unit) performs integer MAC, multiply and divide operations.
- Fixed Microinstruction Size of 32bits.
- Fixed-length instruction execution (2 eTPU clocks)
- Static superscalar operation

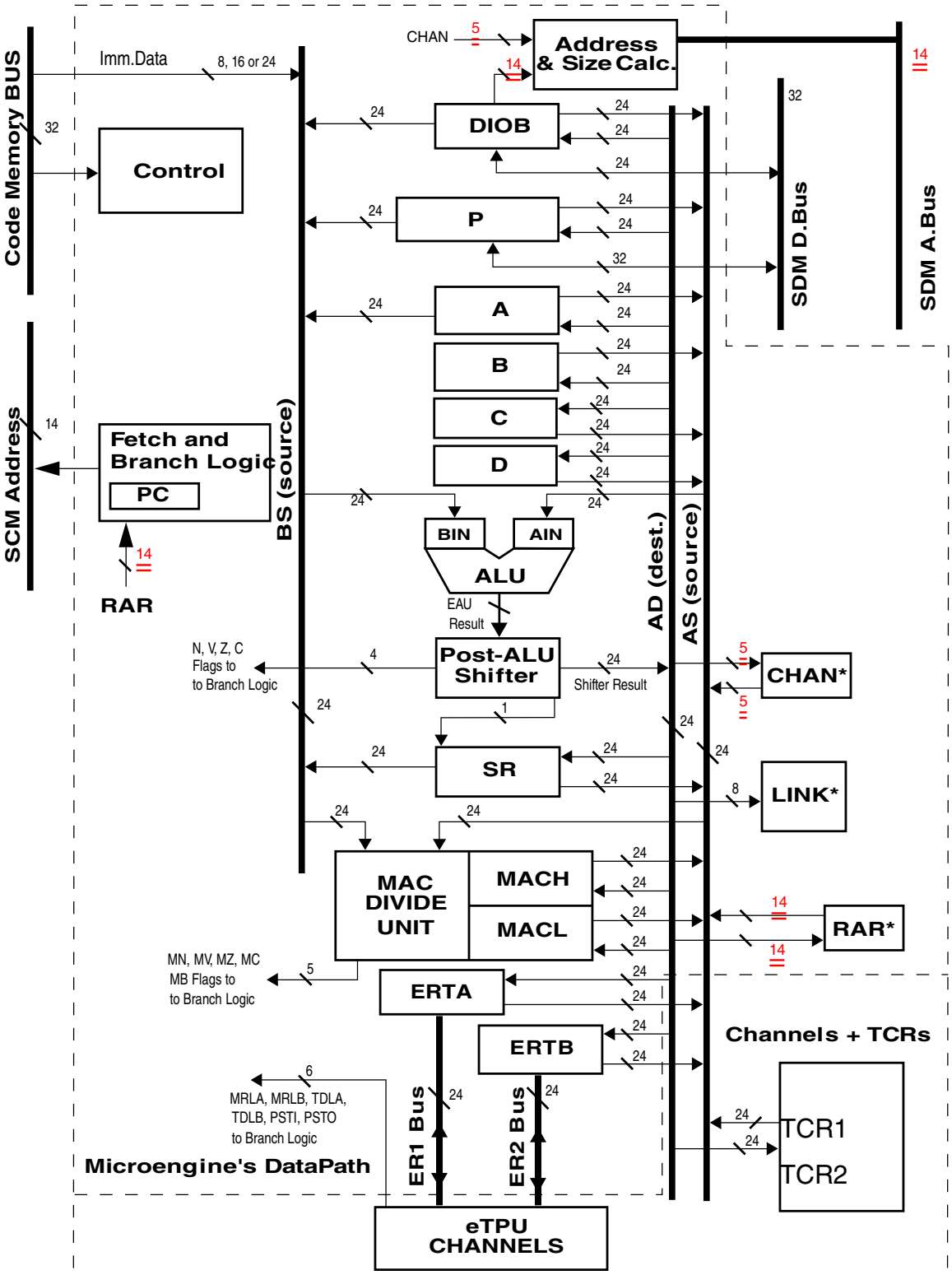


Figure 2-41. Microengine Block Diagram

2.14.8.1 Registers

eTPU microengine accesses a total of 18 registers. Fourteen of them are special purpose (registers A, B, C and D are for general use). Special purpose registers except CHAN and LINK can also be used as general use if the operation that use their contents are not performed. Register description is intended to just introduce their functionalities and not to provide detailed explanation of it since it will be described in Microinstruction Set. Registers less than 24 bits in size are right-justified.

None of the registers have guaranteed reset values. However, some are initialized just before the thread starts (see Time Slot Transition).

2.14.8.1.1 P Register

P register is the only one that is 32-bit wide. It can be used as source and destination for arithmetic/logical operation, and as source and destination for SDM read/write operations.

For P source/destination possibilities in ALU/MDU microoperations, see Selecting sources and destination.

When P is used as SDM read/write operations source or destination there are only 3 possibilities of access: all 32 bits, lower 24 bits and upper 8 bits. SDM operations are explained in detail in SDM Microoperations.

P is automatically loaded with one parameter before the thread starts (parameter preload). For more information see Entry Point format and Time Slot Transition.

Upper 8 bits of P register can be used as application state, since these bits can be tested as branch conditions. P[31:24] is also used in dispatch microoperation (see Dispatch Microoperation), and bit pairs P[29:28], P[27:26], P[25:24] can be directly copied into Channel flags 1 and 0 using field FLC. Together with Entry Table Condition Encoding, it provides fast state resolution without code execution.

2.14.8.1.2 DIOB Register

DIOB is an abbreviation for Data Input/Output Buffer. DIOB is 24-bit wide and can be used as source and destination for arithmetic/logical operations as well as SDM data source and destination. DIOB only can be accessed as 24 bits, both in arithmetic/logical and SDM read/write operations. When using DIOB to perform an SDM access, only lower 24 bits of SDM will be accessible (SDM upper 8 bits always remains unchanged).

DIOB can also be used as SDM addressing register, when DIOB contents is used as absolute SDM address (14-bit wide). In this case DIOB can also be pre-decremented or post-incremented (see Indirect Addressing Mode).

DIOB is automatically loaded with one parameter before the thread starts (parameter preload). For more information see Entry Point format and Time Slot Transition.

2.14.8.1.3 ERTA and ERTB Registers

ERTA/B registers are 24-bit wide and can be used as source or destination in arithmetic/logical operations. ERTA/B are the only source for channel's match registers write (see Write Channel Match and UDCM Registers). ERTA can also be the source for UDCM write.

When a thread starts to be executed, ERTA and ERTB are loaded with a copy of CaptureA and CaptureB registers respectively. ERTA/B can be used to receive a copy of MatchA and MatchB registers. ERTA/B are the only destination of MatchA/B read operation (see Special T4ABS Source Operation: Read Match Registers).

ERTA and ERTB also receive a copy of CaptureA and CaptureB registers when CHAN register is written (see CHAN Register). For more information about Capture and Match registers see MatchA and MatchB Registers and CaptureA and CaptureB Registers.

2.14.8.1.4 SR register - Shift Register

SR is a 24-bit wide register that can be used as source and destination register for arithmetic/logical operations. SR can shift right its contents by 1 bit at time and, at the same time, receive in its bit 23 the lost bit of a shift-right operation in post-ALU shifter (ALU and Post-ALU Shifter), allowing SR to be used to perform 48-bit shift right (see Shift Operations).

2.14.8.1.5 MACH and MACL Registers

Both MACH and MACL are 24-bit registers, part of MAC/Divide unit (see MAC and Divide Unit (MDU)). They can be used as source and destination in most arithmetic/logic operations. When multiply or divide operations are used (multiply-accumulate included), MACH and MACL have special purpose and some restrictions apply, see MAC and Divide Unit (MDU) for more information.

2.14.8.1.6 LINK Register

Link Register is an 8-bit wide register and can be used only as destination in arithmetic operations. LINK is a write-only command register, which precludes its use as a source register for ALU operations. When LINK register is written, it issues a service request for the channel number and eTPU engine equal to the number written in LINK register (see Functions and threads and Channel Link for information about Link Service Request).

2.14.8.1.7 RAR Register

RAR is a 14-bit register and can be used as source and destination in arithmetic operations. RAR also receives the contents of PC register when a subroutine call is executed. The contents of RAR register is loaded into PC when a return from subroutine is executed. RAR is loaded with value 0x3FFF during TST. For more information about subroutine call and return see Branch Operations and Return from subroutine respectively.

2.14.8.1.8 CHAN Register

CHAN is a 5-bit register that can be used as source and destination in arithmetic operations. The contents of CHAN register affects the execution of many channel-related microinstructions, because its number indicates the **selected channel** . CHAN register must not be used to store temporary values in arithmetic operations. For more details, refer to Channel Selection Register - CHAN.

2.14.8.1.9 Counter Registers: TCR1, TCR2, TPR and TRR

All these registers are 24-bit wide except TPR, which is a 16-bit register. They can be read or written in arithmetic/logical operations, and have special-purpose uses for time base and angle mode operations. For more information about those registers see Time bases and EAC - eTPU Angle Counter.

2.14.8.1.10 General Purpose Registers: A, B, C and D

A, B, C and D are 24-bit general purpose registers, which can be used to store intermediate values and do not have other specific uses with any eTPU feature.

2.14.8.2 ALU and Post-ALU Shifter

The ALU executes 24-bit arithmetic and logical operations. ALU's output goes directly to a 1-bit shifter, called post-ALU shifter, so it is possible, for example, to add and shift using only one microinstruction.

In some microinstruction formats, it is not possible to specify the operation executed by ALU. In these cases ALU will always perform addition.

In formats which have the field ALUOP for ALU operation selection, all of them can be performed, including add/subtract using C (carry) flag as ALU's carry-in, bitwise AND/OR/NOT/XOR, and shift/rotate of 2, 4, 8 and 16 bits. See ALU/MDU Operation Selection.

Subtraction, inversion, increment and decrement can be performed by combinations of source inversion and setting ALU's carry-in to 1.

ALU always performs 24-bit operations on its inputs, called **A-source** and **B-source**, and outputs a 24-bit result. 8-, and 16-bit inputs are zero padded to 24 bits. Likewise, ALU 24-bit output is always truncated to the destination register size.

2.14.8.2.1 ALU flags

Four flags - Carry, Negative, Overflow, Zero - described below, are related to ALU and post-ALU shift operations. Operation size and shifting affect flags generation logic. Operation size determines the result boundary to be used for flags generation. Operation size is determined by size of sources and destination (see Flags Sampling Control). For more information about flag generation, see Flags Sampling Control. ALU flags can be used as branch condition (see Conditional/unconditional branch) or conditional ALU/MDU operation (see Conditional ALU/MDU Operation Execution).

Field CCS/CCSV in microinstructions can force **no update** of all flags. Not all flags are updated in all ALU operations: Overflow is updated only on addition and absolute value operations, Carry flag is updated in most ALU operations, and only Zero and Negative are updated in all ALU operations.

ALU flags are never updated when microinstruction starts an MDU operation, regardless of CCS/CCSV, but are updated normally afterwards, on ALU operations that are executed in parallel with an ongoing MDU operation (MDU has its own flags).

Note

Operation size can be smaller than destination register. For example: 0xFFFF + 0x0001 (both 16-bit sources) stores 0x10000 in a 24-bit register and sets Zero and Carry flags because operation size is 16 bits.

2.14.8.2.1.1 Carry flag (C)

In an unsigned addition without shifting, Carry flag is the ALU carry from bit 7 to 8, 15 to 16, or 23 to 24 on 8-, 16- and 24-bit operation sizes respectively. In an unsigned subtraction without shifting, Carry Flag represent the sign of ALU's result considering operation size (Carry Flag equal to 1 means a negative result).

Carry flag definition is operation-dependent. The Carry flag in add/subtraction with Post-ALU shift is defined in Table 2-51 . Find the definitions for other operations in the following sections.

2.14.8.2.1.2 Negative flag (N)

Negative flag indicates the sign of result based on the operation size, regardless of the operation performed, as shown in the following table.

Table 2-47. Negative (N) flag behavior

Operation size	Value
8 bits	N = result[7]
16 bits	N = result[15]
24 bits	N = result[23]

Note

The N flag may not reflect the sign of the value actually written into the destination register, if it does not have the same size of the operation (see Flags Sampling Control). This is always the case for registers RAR (14 bits) and CHAN (5 bits).

2.14.8.2.1.3 Overflow (V)

Overflow is updated only on addition (with or without carry) and absolute value operations. In signed operations, overflow flag indicates that the result of arithmetic operation (add or subtraction) can not be represented by a word of the size of the operation. Overflow flag behavior for addition is defined in the following table. Overflow flag for Absolute operation is explained in Absolute Value operation . V Flag is calculated using ALU adder output (i.e., it is not affected by 1-bit shift/rotate operations).

Table 2-48. Overflow flag on addition - V

Op. size	Value[1]
8 bits	(AS[7] & BS[7] & !alu_adder_output[7]) | (!AS[7] & !BS[7] & alu_adder_output[7])
16 bits	(AS[15] & BS[15] & !alu_adder_output[15]) | (!AS[15] & !BS[15] & alu_adder_output[15])
24 bits	(AS[23] & BS[23] & !alu_adder_output[23]) | (!AS[23] & !BS[23] & alu_adder_output[23])

1. BS is taken after any inversion by the BINV field, but not added to the carry bit (CIN field)

NOTE

For V-flag definition on the absolute operation, see “Absolute Value Operation".

2.14.8.2.1.4 Zero flag (Z)

Zero flag equal to 1 indicates that the result from the ALU, limited to the operation size, is zero, regardless of the operation performed, whether the result is written, or where it is written. It depends on the operation size, as shown in the following table.

Table 2-49. Zero flag - Z

Operation size	Value
8 bits	Z = (result[7:0] == 0x00)
16 bits	Z = (result[15:0] == 0x0000)
24 bits	Z = (result[23:0] == 0x000000)

2.14.8.2.2 ALU ADD operation with and without shifting

ADD operation is selected by ALUOP or ALUOPI fields and when none of them is available in a microinstruction format[30]. Optionally, result can be shifted or rotated right by 1 bit, which is selected by SHF, ALUOP or ALUOPI fields. See Microinstruction Set for more details. The following table describes how CIN and BINV fields change ADD operation behavior.

Table 2-50. Types of ADD operations

BINV	CIN	Operation (adder output)
1	1	AS + BS
1	0	AS + BS + 1
0	0	AS - BS
0	1	AS - BS - 1

ALU adder output can be 1-bit shifted or 1-bit rotated right as follows:

Shift right:

```
if BINV==1
  result [23:0] = adder_output[24:1]
else
  result [23:0] = adder_output [24:1] xor 0x800000
endif
```

Shift left:

30. ALU operations only occur on formats where a destination field is found (T2ABD/T2D).

```
result[23:1] = adder_output[22:0]
result[0]    = 0
```

Rotate right:

```
case(opsize/CCSV)
8-bit:
  result[6:0]   = adder_output[7:1]
  result[7]     = adder_output[0]
  result[23:8]  = adder_output[23:8]
16-bit:
  result[14:0]  = adder_output[15:1]
  result[15]    = adder_output[0]
  result[23:16] = adder_output[23:16]
24-bit:
  result[22:0]  = adder_output[23:1]
  result[23]    = adder_output[0]
```

Note

Only for the Post-ALU rotate right, the operation size is determined by the field CCSV (see Flags Sampling Control). For example: if CCSV=00, T4ABS=P (24-bits), T4BBS=A (24 bits), T2ABD=B (24 bits), and ALUOP="Add ROR", then B gets A+P with bits 7:0 rotated, even though the operation size is 24 bits.

The following table describes Carry flag behavior.

Table 2-51. Carry flag update on ADD operation

BINV[1]	Op. size	shift/rotate	Value
1	8 bits	none	adder carry from bit 7 to bit 8
1	16 bits	none	adder carry from bit 15 to bit 16
1	24 bits	none	alu_adder_output[24]
0	8 bits	none	!adder carry from bit 7 to bit 8
0	16 bits	none	!adder carry from bit 15 to bit 16
0	24 bits	none	!alu_adder_output[24]
0 or 1	8 bits	shift left	alu_adder_output[7]
0 or 1	16 bits	shift left	alu_adder_output[15]
0 or 1	24 bits	shift left	alu_adder_output[23]
0 or 1	x	shift right	alu_adder_output[0]
1	8 bits	rotate right	adder carry from bit 7 to bit 8
1	16 bits	rotate right	adder carry from bit 15 to bit 16
1	24 bits	rotate right	alu_adder_output[24]
0	8 bits	rotate right	!adder carry from bit 7 to bit 8
0	16 bits	rotate right	!adder carry from bit 15 to bit 16
0	24 bits	rotate right	!alu_adder_output[24]
			except on max-constant generation (see Max Const Generation With T4BBS=111)

1. BINV has no effect on carry-out when used to code max constant (see Max Const Generation With T4BBS=111).

Flags N and Z on shift are updated according to the result after shift. Flag V with shift is updated according to the ADD operation only, the same way as without shift.

2.14.8.2.3 ADC Operation

ADC operation is selected by the ALUOP field. CIN field is ignored when this operation is selected. The following table describes how BINV change ADC operation behavior.

Table 2-52. Types of ADC operations

BINV	CIN	Operation
1	x	AS + BS + C flag
0	x	AS - BS - C flag

Flags behave exactly the same way as for ADD operation without shift/rotate.

2.14.8.2.4 Bitwise Operations

Bitwise AND, OR and XOR are selected by ALUOP field. On these operations CIN field is ignored and BINV field inverts (bitwise NOT) BS. C and V Flags are never updated on these operations. The following table describes AND, OR and XOR bitwise operations.

Table 2-53. Types of Bitwise Operations

ALUOP	BINV	Operation
10000	1	AS | BS
10000	0	AS | (~BS)
10001	1	AS ^ BS
10001	0	AS ^ (~BS)
10010	1	AS & BS
10010	0	AS & (~BS)

2.14.8.2.5 Set Bit / Clear Bit Operations

These operations set or clear the AS bit determined by BS[4:0]. If the bit number resolves to a value greater than 23, no bit is set or cleared (i.e., result is equal to AS). On these operations CIN field is ignored and BINV field inverts (bitwise NOT) BS. C and V flags are never updated for set/clear bit operations. These operations **override** B-Source size to 8 bits.

set bit (BINV = 1):

```
result = AS | (1 << BS)
```

clear bit (BINV = 1):

```
result = AS & ~(1 << BS)
```

set bit (BINV = 0):

```
result = AS | (1 << (31 - BS))
```

clear bit (BINV = 0):

```
result = AS & ~(1 << (31 - BS))
```

2.14.8.2.6 Exchange Bit

Exchange the AS bit determined by BS with C flag. If the bit number resolves to a value greater than 23, no exchange is performed (i.e., result is equal to AS and C flag is not updated). This operation **overrides** BS size to 8 bits. On this operation, CIN field is ignored and BINV field inverts (bitwise NOT) BS. V flag is never updated on exchange bit operation. C flag is always updated, regardless of CCSV, unless BS > 23.

Exchange Bit (BINV=1):

```
if BS <= 23
begin
  temp_C_flag = AS[BS]
  if C_flag == 1
    result = AS | (1 << BS)
  else
    result = AS & ~(1 << BS)
  C_flag = temp_C_flag
end
```

Exchange Bit (BINV = 0):

```
if (31 - BS) <= 23
begin
  temp_C_flag = AS[31 - BS]
  if C_flag == 1
    result = AS | (1 << (31 - BS))
  else
    result = AS & ~(1 << (31 - BS))
  C_flag = temp_C_flag
end
```

2.14.8.2.7 Multibit shift/rotate operations

These operations shift or rotate AS by 2, 4, 8 or 16 bits. Size of shift/rotate is determined by BS[1:0]. The following table describes the number of shifted/rotated bits depending on BS[1:0] value.

Table 2-54. Number of shifted/rotated bits for each BS[1:0] value

BS[1:0]	Bits shifted/rotated
0	2
1	4
2	8
3	16

Shift right is a logical operation (i.e., zeros are inserted on left). Multibit shift and rotate operations **overrides** BS size to 8 bits. The shifts and rotate operate on 24 bits, independently of the operation size.

V flag is never updated for multibit shift or rotate operations. Carry flag behavior is described in the following table. CIN is ignored in these operations, but BINV is effective.

Table 2-55. Carry flag value on multibit shift/rotate operations

ALUOP	BS[1:0]	C flag value[1]
11001 (shift left)	0	AS[22]
11001 (shift left)	1	AS[20]
11001 (shift left)	2	AS[16]
11001 (shift left)	3	AS[8]
11010 (shift right)	0	AS[1]
11010 (shift right)	1	AS[3]
11010 (shift right)	2	AS[7]
11010 (shift right)	3	AS[15]
11011 (rotate right)	0	AS[2]
11011 (rotate right)	1	AS[4]
11011 (rotate right)	2	AS[8]
11011 (rotate right)	3	AS[16]

1. CCS/CCSV can disable flag update on multibit shift/rotate, but the specified flag size in CCSV is ignored for the C flag.

2.14.8.2.8 Absolute Value operation

Absolute Value operation is selected by ALUOP field. On this operation, AS is interpreted as a signed number and its absolute value is the result. V and N flags are updated with the result signal determined by the operation size. AS bit 23 after size override and sign extension (if any, see A-Source Size Override), regardless of A-source register size, is used to check the operand signal and is copied to C-flag. Note that if AS is 8-bit or 16-bit, its sign is taken into account and copied to C only if sign-extension is performed. The following table summarizes flag updating for Absolute Value operation.

This operation is independent of B-source. Instruction fields T4BBS, BINV and CINV are ignored in this operation. The Absolute Value operation size is the minor between A-source size and Destination size.

Table 2-56. ALU flags in Absolute Value operation

operation size	V, N[1]	C	Z
8	alu_output[[7]	AS[23]	alu_output[7:0] == 0
16	alu_output[15]		alu_output[15:0] == 0
23	alu_output[23]		alu_output[23:0] == 0

1. V, N can be 1 on 8- and 16-bit Absolute Value, because the operand sign is always taken from bit 23. V, N can also be 1 in 23-bit Absolute Value (or 8-bit and 16-bit with sign extension), if the operand is 0×800000 (0×80, 0×8000).

2.14.8.3 MAC and Divide Unit (MDU)

MDU is an autonomous resource in the microengine which can carry out sequential multiply, multiply-accumulate, fractional multiplication and divide operations, selected through the microinstruction fields ALUOP or ALUOPI. The unit supports signed and unsigned multiply and fractional multiplication of any combination of 8, 16 or 24 bit operands [31], and also signed and unsigned 24-bit multiply-accumulate. Divide operation is unsigned, and both operands are always 24-bit wide.

Depending on the size of operands and the type of operation, MDU can take more than one microcycle to execute the operation, but microengine continues to execute microinstructions in parallel. When the microcode issues an END command, any MDU executing operation terminate immediately and is left incomplete. When selecting an operation that uses MDU, the result is always placed in MACH and MACL registers, and the register selected as destination does not have its value changed (Selecting sources and destination). During calculations, MACH and MACL holds temporary values and should not be written, otherwise the result is unspecified. One must not start an MDU operation while MDU is already busy: the result is unspecified for both the ongoing operation and the started one.

MDU Operations update its own set of 5 flags, described in MDU Flags. MDU operations never update C, N, V and Z flags. CIN and BINV microinstruction fields affect MDU operations according to the follwoing table.

Table 2-57. CIN and BINV with MDU Operations

B-source operand	BINV	CIN	Op. Performed
signed	1	1	AS mdu_op BS

Table continues on the next page...

31. There is no distinct selection of 24-bit fractional multiplication, for it works exactly as a 24-bit ordinary multiplication.

Table 2-57. CIN and BINV with MDU Operations (continued)

B-source operand	BINV	CIN	Op. Performed
	0	0	AS mdu_op (-BS)
	1	0	reserved
	0	1	reserved
unsigned[1]	1	1	AS mdu_op BS
	1	0	AS mdu_op (BS+1)
	0	x	reserved

1. includes the B-source (unsigned) in fmults (signed) operations.

2.14.8.3.1 Multiply and Multiply-Accumulate Operation Length

MDU needs 2 sources, A source and B source, to perform an operation. The time needed to perform a multiply or multiply-accumulate is:

- on 24-bit x 8 bit multiplies: 2 microcycles (one start-MDU plus one execution microcycle)
- on 24-bit x 16 bit multiplies: 3 microcycles (one start-MDU plus two execution microcycles)
- on 24 bit x 24 bit multiplies/macs: 4 microcycles (one start-MDU plus three execution microcycles)

An internal pipeline in MDU allows multiply-accumulate (or even non-fractional multiply) operations to start one microinstruction before a multiplication or multiply-accumulate (signed or unsigned) has been completed (e.g., one can start one multiply or multiply-accumulate once every three microinstructions). However, by doing that it is not possible to read the result in MACH and MACL (although the MDU flags can be tested), so this is intended to be used in a multiply-accumulate sequence. It is also allowed to mix different sizes in multiply/mac sequences.

Multiply-accumulate operations are similar to multiply operations, except that the contents of MACH and MACL registers are added to the multiplication result.

When multiply or multiply accumulate operations finish, MACL and MACH hold the least and the most significant 24-bit words, respectively.

2.14.8.3.2 Divide Operation Length

The divide operation is always unsigned. The division completes in 13 microcycles, meaning that after the start divide microinstruction, one has to wait for 12 microcycles and then read the result and the remainder in MACH and MACL registers. During the 12 execution microcycles, microengine can execute microinstructions unrelated to the MDU.

2.14.8.3.3 Signed Multiplication (mults)

MDU signed multiplication is defined as follows:

```
(signed) MACH,MACL = (signed) AS * (signed) BS
```

MC and MV flags are reset. MZ is set if result is 0, resets otherwise. MN is set if result is negative.

2.14.8.3.4 Unsigned Multiplication (multu)

MDU unsigned multiplication is defined as follows:

```
(unsigned) MACH|MACL = (unsigned) AS * (unsigned) BS
```

MC and MV flags are reset. MZ is set if result is 0, resets otherwise. MN is a copy of the most significant bit of result.

2.14.8.3.5 Signed Multiply-Accumulate (macs)

MDU signed multiply-accumulate is defined as follows:

```
(signed/unsigned) {MACH,MACL} += (signed) AS * (signed) BS
```

MC is not altered.

MV is set if result can not be represented by a 48-bit signed number. MACS never resets MV flag: it is left as is if no overflow occurs, or set it otherwise. This allows checking the overflow flag only once at the end of a series of multiply-accumulate operations in a scalar product calculation.

```
if (({MACH,MACL} += AS * BS < -2^47) || ({MACH,MACL} += AS * BS > 2^47 - 1))
  MV = 1
```

MZ is set if result is 0, resets otherwise. MN is a copy of the most significant bit of result.

Note that only 24-bit multiply-accumulate is available.

2.14.8.3.6 Unsigned Multiply-Accumulate (macu)

MDU Unsigned Multiply-Accumulate is defined as follows:

```
(signed/unsigned) {MACH,MACL} += (unsigned) AS * (unsigned) BS
```

MC is set if result can not be represented by a 48-bit unsigned non-negative number. MACU never resets MC flag: MC flag is left as is if no carry occurs, or set otherwise. This allows checking the carry flag only once at the end of a series of multiply-accumulate operations in a scalar product calculation.

```
if (({MACH,MACL} += AS * BS < 0) || ({MACH,MACL} += AS * BS > 2^48 - 1))
  MC = 1
```

MV is not altered.

MZ is set if result is 0, resets otherwise. MN is a copy of the most significant bit of result.

Note that only 24-bit multiply-accumulate is available.

2.14.8.3.7 Signed Fractional Multiplication (fmults)

MDU Signed Fractional Multiplication takes the B-Source as an **unsigned** 8- or 16-bit[32] fraction between 0 and $(2^8 - 1)/2^8$ (inclusive) for the 8-bit operation, or between 0 and $(2^{16}-1)/2^{16}$ (inclusive) for the 16-bit operation. Only A-Source is taken as a signed number. The value of B-Source is considered the unsigned numerator of a fraction with denominator 2^8 or 2^{16} for the 8- and 16-bit operations, respectively.

The integer part of the result is stored in MACH, and the fractional part in MACL. The result is signed, so that the concatenation of MACH and MACL form a 48-bit fixed point number with a 24-bit integer part, both for 8- and 16-bit operations. To calculate the unsigned numerator of the fractional part (with denominator 2^{24}) of the result, one must take the absolute value of MACL considering the signal of the result (not MACL alone), i.e.: if flag MN=1, invert MACL and add 1.

MDU flags are updated in the same way as in the Signed Multiplication.

2.14.8.3.8 Unsigned Fractional Multiplication (fmultu)

MDU Unsigned Fractional Multiplication takes both A-Source and B-Source as unsigned operands. B-Source is taken as an 8- or 16-bit[1] fraction between 0 and $(2^8 - 1)/2^8$ (inclusive) for the 8-bit operation, or between 0 and $(2^{16}-1)/2^{16}$ (inclusive) for the 16-bit operation. The value of B-Source is considered the numerator of a fraction with denominator 2^8 or 2^{16} for the 8- and 16-bit operations, respectively.

32. 24-bit fractional signed/unsigned multiplication is equivalent to mults/multu.

The integer part of the result is stored in MACH, and the fractional part in MACL. The fractional part in MACL is the numerator of a fraction with denominator 2^{24} . The concatenation of MACH and MACL form a 48-bit fixed point number with a 24-bit integer part, both for 8- and 16-bit operations.

MDU flags are updated in the same way as in the Unsigned Multiplication.

2.14.8.3.9 Unsigned Divide (div)

At the end of a divide operation MACL holds the result of the division, taking A-source as numerator and B-source as denominator, while MACH holds the remainder. If a divide by 0 is executed, MACL holds the maximum unsigned number (0xFFFFFF) as result and flag MV is set to indicate division by 0 (otherwise reset). The contents of MACH become indetermined.

MC flag is always reset.

MZ flag is set if MACL equals 0, and reset otherwise.

MN receives a copy of MACH bit 23 (msb from the remainder).

Note that signed division is not available.

2.14.8.3.10 MDU Flags

MDU has its own flags to indicate the result and status of an MDU operation. They are: MC, MZ, MV, MN and MB. All MDU flags are updated with the final result at the end of the operation, and do not change until the next operation finishes. Therefore it is possible to start a new MDU operation and test the flags of the previous one in parallel, except for mult/mac with 8-bit operand (takes only 1 microcycle).

2.14.8.3.10.1 MDU Negative Flag - MN

MN flag is always a copy of MACH bit 23 at the end of the operation, either in signed or unsigned ones. Note that MACH holds the rest of a division operation, which is always unsigned.

2.14.8.3.10.2 MDU Carry Flag - MC

MDU carry flag indicates if the result cannot be represented by a 48-bit number, in Signed and Unsigned Multiply Accumulates. It is reset in the other operations.

2.14.8.3.10.3 MDU Zero Flag - MZ

In multiply and multiply-accumulate operations, MDU zero flag is asserted if MACH and MACL are equal to zero at the end of an operation. In divide operations, zero flag is asserted if MACL (result) is equal to 0.

2.14.8.3.10.4 MDU Overflow Flag - MV

In multiply operations, MV flag is negated and keeps negated in the end, because the result of a multiplication can always fit in a 48-bit result (MACH and MACL concatenated). In a multiply-accumulate operation, MV is asserted if the result size is wider than 48 bits. MV flag work in both signed and unsigned operations.

In divide operations it is only asserted if a divide-by-zero operation was executed.

2.14.8.3.10.5 MDU Busy Flag (MB)

MB tests as true at the next microinstruction after the MDU start operation, and as false at the last microcycle of any MDU operation execution.

2.14.8.4 Branch conditions

Microengine allows conditional branch. There are five sets of flags that can be tested in a conditional branch: ALU flags, MDU flags, P flags, Channel flags, and Semaphore flag (flag SMLCK).

When a thread starts to be executed, the values in MDU and ALU flags are not initialized. ALU flags are described in ALU flags, MDU flags are described in MDU Flags. MDU and ALU flags are updated during execution of microinstructions.

P flags are actually the upper byte of P register, which optionally can work as user defined flags (see P Register).

Channel flags Flags0, Flag1, MRLA, MRLB, TDLA, TDLB, PSS, PRSS and PSTO are obtained from the selected channel (value in CHAN register), while channel flags, LSR, FM[0] and FM[1] are selected by the serviced channel, regardless of the CHAN value[33] .

Flags TDLA/B, MRLA/B, LSR, FM and PSS, are sampled at the beginning of a thread. Flag PSS does not change during its execution while CHAN register is not written. When a write in CHAN register is performed, all flags except LSR and FM are updated according to the channel specified by CHAN value. Flags MRLA/B and TDLA/B are reset when their respective latches in channel are cleared by microcode.

33. Serviced channel does not change during execution of a thread, and it is the channel that requested a service (initial value of CHAN register when a thread starts).

Table 2-58. Channel flags as branch condition

Flag	Description	Service or selected channel
Flag0	State Resolution flag	reflects the selected channel (CHAN)
Flag1	State Resolution flag	reflects the selected channel (CHAN)
MRLA	Match A Recognition Latch	these flags reflect the selected channel (CHAN)see MRLA/B - Match Recognition Latches and TDLA/B - Transition Detect Latches for more information.
MRLB	Match B Recognition Latch	
TDLA	Transition A Detection Latch	
TDLB	Transition B Detection Latch	
LSR	Link Service Request	reflects the serviced channel.
PRSS	Input Pin state sampled at service request	reflects the selected channel (CHAN). Does not change if CHAN is not changed.
PSS	Sampled Input Pin State	reflects the selected channel (CHAN). Does not change if CHAN is not changed (see Pin Control Registers).
PSTI	Current Input Pin State.	reflects the selected channel (CHAN). Changes any time.
PSTO	Current Output Pin State	reflects the selected channel (CHAN). Changes any time.
FM	Function Mode Bits	reflects the Function Mode for serviced channel (see eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C*n*SCR_ENG1) and eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C*n*SCR_ENG2))

Semaphore condition SMLCK always indicates if a semaphore is locked for the Engine, resolving as false before any lock attempt. For each trial, the SMLCK flag is updated. The SMLCK value set in one thread is not meaningful to the other. After a free, the SMLCK condition tests as false until a new lock attempt on the same thread.

Branch conditions are selected through instruction fields BCC and BCF (see Conditional/unconditional branch).

2.14.9 Microinstruction Set

Each microinstruction can execute up to 3 microoperations in parallel. Microinstructions are grouped into formats, and there are four types of microoperations:

- ALU/MDU Operations
- SDM Operations
- Channel Configuration/Control Operations
- Flow Control Operations

Each microinstruction format is defined by a set of microinstruction fields, which determine the operations, each belonging to one of the groups above (there may be several in one group). Complete microinstruction formats are shown in Microinstruction formats.

Parallelism conflicts may arise when two operations are executed in the same microinstruction. These situations are explained in Microinstruction Parallelism Issues.

2.14.9.1 SDM Microoperations

The access to SDM is made by providing an address and a register to perform a data transfer, except semaphore operations, which are also classified in the SDM group. Only P and DIOB registers can exchange data with SDM. Microengine always addresses SDM in 32-bit boundaries, for 8, 24, or 32 bit wide data.

Direction is determined by the field RW in all addressing modes: RW=0 selects read and RW=1 selects write.

2.14.9.1.1 SDM Addressing Modes

There are 4 addressing modes in eTPU:

- Absolute
- Selected Channel Relative
- Indirect
- Engine Relative

The addressing modes Absolute and Selected Channel Relative use immediate bits to form the physical address of SDM, which is identified in microinstruction as a field called AID. AID field can be 3-, 7-, or 8-bit wide depending on the addressing mode.

2.14.9.1.1.1 Absolute Addressing Mode

In Absolute addressing mode, the address range is 256 parameters, addressed by field AID, which in this mode is 8-bit wide. These parameters are located in SDM addresses from 0 to 255.

```
physical_address = AID[7:0]
```

2.14.9.1.1.2 Selected Channel Relative Addressing Mode

In Selected Channel Relative addressing mode, only the first 8 (with 3-bit AID) or 128 (with 7-bit AID) parameters of the selected channel are accessible, depending on the microinstruction format. Physical address is calculated using the channel parameter base address that is specified in the CPBA field of eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2). AID field is added to channel parameter base address to compose the physical address. The address equation is:

```
physical_address = selected_channel_parameter_base_address + AID[6:0], or
physical_address = selected_channel_parameter_base_address + AID[2:0]
```

2.14.9.1.1.3 Indirect Addressing Mode

In Indirect Addressing mode the physical address is taken from DIOB register. Only DIOB bits 14 to 2 are relevant. Since the SDM word address is shifted two bits up in DIOB, its contents hold the same parameter address value used by Host. The address equation for SDM access is:

```
physical_address = DIOB[13:2], or
physical_address = (truncated to 12 bits) (DIOB / 4)
```

Indirect addressing mode is used to access the Shared Subsystem Area when eTPU_CCCR_C*x*CR_ENG*n*[SSMAE]=1 and DIOB addresses the 16 Kbytes just above the SDM range (DIOB bit 14 = 1). See Shared Subsystems support for more details. If SSMAE=0, accesses with DIOB bit 14 = 1 hit the same SDM addresses as with bit 14 = 0, providing full backwards compatibility with respect to DIOB contents.

Indirect addressing mode can have post-increment or pre-decrement on DIOB, allowing stack operations. See DIOB stack operation for more information.

2.14.9.1.1.4 Engine Relative Addressing Mode

In Engine Relative Addressing mode the physical address is the concatenation of the eTPU_SCR_ECR_ENG*n*[ERBA] field (see eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2)) with the 7-bit AID instruction field. This allows the same function microcode, when running on distinct engines, to access different address spaces, global to the engine only.

The SDM physical word address in Engine Relative Addressing mode is calculated as follows:

```
physical address = (eTPU_SCR_ECR_ENGn[ERBA]<< 7) + AID[6:0]
```

2.14.9.1.2 SDM source/destination registers

When performing an SDM operation, only DIOB or P can be used as data source or destination. P is 32-bit wide, and DIOB is 24-bit wide. Microinstruction field P/D (1 bit) is used to choose between P and DIOB as data the source or destination. When the P/D field is not available in microinstructions that support SDM access, the source/destination is P.

Table 2-59. SDM source/destination Register Selection

P/D	Meaning
0	P access
1	DIOB access

2.14.9.1.3 SDM Operation Size

When using DIOB register to perform SDM data transfers, the operation size is always 24-bit wide (lower 24 bits of SDM). When using P register, the operation size can be 8-, 24- or 32-bit wide, which is controlled by microcode RSIZ field (2 bits). RSIZ meaning is shown in the following table.

Table 2-60. SDM P access Size

RSIZ	P access	DIOB access
00	full 32-bit access (i.e. P[31:0]=SDM[addr] [31:0])	RESERVED
01	only upper 8 bits are transferred (i.e. P[31:24] = SDM[addr] [31:24])	RESERVED
10	only lower 24 bits are transferred (i.e. P[23:0] = SDM[addr] [23:0])	DIOB = SDM[addr] [23:0
11	RESERVED	RESERVED

RSIZ is not available in all microinstructions that support SDM access. In microinstructions where RSIZ field is not available, SDM access will be 24 bits by default.

When performing a Zero SDM write operation (see Zero SDM Operation), RSIZ defines the size of operation regardless of the P/D field (SDM source/destination registers).

2.14.9.1.4 SDM access direction

RW field defines the direction of the access in the SDM. The access direction is summarized in the following table.

Table 2-61. SDM access direction

R/W	Meaning
0	read SDM parameter into P or DIOB registers
1	write SDM parameter from P or DIOB registers

2.14.9.1.5 Zero SDM Operation

Zero SDM operation is controlled by microcode field ZRO (1bit). When ZRO field is 0, the data portion written in SDM or in P/DIOB (SDM read) registers will always be 0x0. When performing a Zero SDM write operation, the RSIZ is relevant regardless of the P/D field (usually RSIZ is meaningful only for P/D = 0), which means that zero SDM write operation can be performed with 32, 24 or 8 bits according to SDM operation size. These conditions are summarized in the following table.

Table 2-62. Zero SDM Operation

ZRO	RW	P/D	Meaning
0	0	0	Clear P register. Size is determined by RSIZ field. See SDM Operation Size
0	0	1	Clear DIOB (all 24 bits), independently of RSIZ
0	1	x	Clear SDM parameter. Size is determined by RSIZ field. See SDM Operation Size
1	RW	P/D	Regular SDM operation

Note

When field STC is present, STC=11 also disables Zero SDM operation (see Table 2-63). The conflicts with DIOB operations (see DIOB stack operation) and ALU operations are resolved like a normal SDM operation (see Table 2-104).

Zero SDM read operations do check memory errors on the read data, however discarded. Zero SDM write operations normally update the ECC of the written data.

2.14.9.1.6 DIOB stack operation

SDM Indirect Addressing Mode (see Indirect Addressing Mode) is used if STC field (2 bits) exists in the microinstruction, controlling automatic increment/decrement of DIOB register, as shown in the following table, thus allowing stack operations. DIOB is incremented and decremented in word addresses, only from bits 15 downto 2, i.e.: the bits 23 to 16 and 1 to 0 are left untouched by STC pre-decrement and post-increment.

Table 2-63. DIOB Post-Increment / Pre-Decrement - STC

STC	Meaning
00	Post-Increment of DIOB
01	Pre-Decrement of DIOB
10	No Increment/Decrement (normal access)
11	No SDM Access[1]

1. Also disables Zero SDM operation.

2.14.9.1.7 Semaphore Operations

Semaphore lock and free operations are available through eTPU microcode. For more information about semaphores see Hardware Semaphores. There are 2 microinstruction fields that control semaphore operations: FL (1 bit) and SMPR (2 bits). Serviced channel sees 4 semaphores, selected by field SMPR.

Table 2-64. Semaphore Operations Fields

Field	Meaning
FL	0 = free semaphore, 1 = lock semaphore
SMPR	semaphore number selector

When freeing a semaphore, the field SMPR has no meaning. This is because only one semaphore can be locked at a time by each engine, so when freeing a semaphore it is not necessary to specify its number.

Note

If microcode tries to lock a semaphore already locked for the same Engine, the semaphore continues locked for the Engine and the SMLCK branch condition resolves as true.

2.14.9.2 ALU/MDU Operations

ALU/MDU microoperations mostly comprises two sources, one destination and one operation. The operation is generally selected through fields ALUOP, ALUOPI or SHF. In formats where there is no operation selection field (ALUOP, ALUOPI or SHF), the operation performed is always addition; however, it is possible to perform subtraction, increment or decrement using fields BINV (see B-Source Inversion) and CIN (see Carry-in Control).

2.14.9.2.1 Source and Destination Register Set Selection

Microcode field T4ABS allows selection of a source from either one of two register sets, shown in Table 2-68. The same applies to T2ABD, used for ALU destination selection with other two register sets, as shown in Table 2-69. Microinstruction fields ABSE and ABDE control the register set selection for source and destination, respectively, when available at the format. In formats without ABSE/ABDE, the field T4BBS determines the register sets used by T2ABD and T4ABS, as shown in Microinstructions without fields ABSE and ABDE.

2.14.9.2.1.1 Microinstructions With Fields ABSE and ABDE

In microinstructions where ABSE and ABDE fields are available (1 bit each), ABSE controls register set selection for T4ABS and ABDE controls register set selection for T2ABD. The following table shows the meaning of values for ABSE and ABDE fields.

Table 2-65. Register Set Selection by ABSE or ABDE

ABSE or ABDE	Register Set Selected
0	second
1	first

2.14.9.2.1.2 Microinstructions without fields ABSE and ABDE

When ABSE and ABDE are not available in a microinstruction format, register set is selected by T4BBS field. The following table explains how to select a register set either for T4ABS and T2ABD.

Table 2-66. Register set selection by T4BBS without ABSE and ABDE

T4BBS	Register set for T2ABD	Register set for T4ABS
0xx	first	first
100	second	second
101	second	first
110	first	second
111	first	first
none[1]	first	first

1. Refers to operations with immediate data as B-source, without ABSE and ABDE.

2.14.9.2.2 Selecting sources and destination

All ALU/MDU operations need 2 sources (called AS and BS) and 1 destination (called AD), except for some of those that use immediate data (see Operations With Immediate Data). Fields T4ABS (4 bits), ABSE (1 bit), T4BBS (3 bits) select sources, while T2ABD (4 bits) and ABDE (1 bit) select the destination. When MDU is used (multiply/divide), T2ABD destination selection is ignored and results are stored in MACH and MACL (see MAC and Divide Unit (MDU)). ABSE and ABDE are not available in some microinstruction formats that support ALU/MDU operations. However, in all formats where ABSE is available, ABDE is also available and vice-versa. The existence of ABSE/ABDE fields changes the meaning of T4BBS field, as shown in the table below. On instructions with immediate data, it is used as B-source (see Operations With Immediate Data).

All sources and destinations have a size associated to it, and these sizes are used to select flag sample position (see Flags Sampling Control). Sizes can be 8, 16 or 24 bits. Registers that are not exactly of one of these sizes are treated as the immediately upper size (e.g., CHAN[4:0] is an 8-bit source). See Flags Sampling Control for more information.

Some parallelism issues arise when selecting P, DIOB, ERTA or ERTB as destination registers, since they can be modified by other microoperations in the same microinstruction (see Microinstruction Parallelism Issues for details).

Table 2-67. B source selection - T4BBS

T4BBS	Meaning in microinstruction formats with ABSE/ABDE	Meaning in microinstruction formats without ABSE/ABDE[1]
000	BS[23:0] = P[23:0]	
001	BS[23:0] = A[23:0]	
010	BS[23:0] = SR[23:0]	
011	BS[23:0] = DIOB[23:0]	
100	reserved	BS = 0
101	reserved	BS = 0
110	reserved	BS = 0
111	BS=0, or Max const., if CIN=0 and BINV=0 (see Generating "max" constant).	

1. T4BBS also selects A-source and destination register set in this case, according to Table 1.

T4ABS selects one source from 2 register sets, shown in the following table. ABSE and T4BBS control which set T4ABS field uses to select the source. For more information about how to select a register set for T4ABS and T2ABD see Source and Destination Register Set Selection . All sources are zero-filled to 24 bits, unless sign-extension is specified (see A-Source Size Override).

Table 2-68. A source selection - T4ABS

T4ABS	First register set		Second register set	
	Selected register	Size	Selected Register	Size
0000	AS[7:0]=P[7:0]	8	AS[7:0]=0	8
0001	AS[7:0]=P[15:8]	8	AS[23:0]=C[23:0]	24
0010	AS[7:0]=P[31:24]	8	AS[15:0] = TPR[15:0]	16
0011	AS[23:0] = ERTB[23:0]	24	AS[23:0] = B[23:0]	24
0100	AS[23:0] = D[23:0]	24	AS[23:0] = TRR[23:0]	24
0101	AS[15:0] = P[15:0]	16	AS[7:0] = 0, read_match[1]	8
0110	AS[15:0] = P[31:16]	16	AS[13:0] = RAR[13:0]	16
0111	AS[7:0] = P[23:16]	8	AS[23:0] = MACH[23:0]	24
1000	AS[23:0] = P[23:0]	24	AS[23:0] = MACL[23:0]	24
1001	AS[23:0] = A[23:0]	24	AS[4:0]=CHAN[4:0]	8
1010	AS[23:0] = SR[23:0]	24	AS[14 :2] = CHAN_BASE[2]	16
1011	AS[23:0] = DIOB[23:0]	24	AS[13:0] = ENGINE_BASE[3]	16
1100	AS[23:0] = TCR1[23:0]	24	Reserved	-
1101	AS[23:0] = TCR2[23:0]	24	Reserved	-
1110	AS[23:0] = ERTA[23:0]	24	Reserved	-
1111	AS[23:0] = 0	24	Reserved	-

1. T4ABS=0101 with second register set also reads MatchA/B registers into ERTA/B (see Special T4ABS Source Operation: Read Match Registers).
2. CHAN_BASE is the selected channel's base SDM address in channel relative address mode (see CHAN_BASE as a source).
3. ENGINE_BASE is the eTPU_SCR_ECR_ENG*n*[ERBA] field shifted left 9 positions.

T2ABD selects the destination from 1 of 2 register sets, shown in the following table. ABDE and T4BBS control which set T2ABD field uses to select the destination.

Table 2-69. Destination selection - T2ABD

T2ABD	First register set		Second register set	
	Selected register	Size	Selected Register	Size
0000	A[23:0] = AD[23:0]	24	C[23:0] = AD[23:0]	24
0001	SR[23:0] = AD[23:0]	24	LINK[7:0] = AD[7:0]	8
0010	ERTA[23:0] = AD[23:0][1]	24	TPR[15:0] = AD[15:0]	16
0011	ERTB[23:0] = AD[23:0][2]	24	B[23:0] = AD[23:0]	24
0100	DIOB[23:0] = AD[23:0]	24	CHAN[4:0] = AD[4:0]	8
0101	P[15:0] = AD[15:0]	16	D[23:0] = AD[23:0]	24
0110	P[31:16] = AD[15:0]	16	RAR[12:0] = AD[12:0]	16
0111	P[23:0] = AD[23:0]	24	MACH[23:0] = AD[23:0]	24
1000	TCR1[23:0] = AD[23:0]	24	MACL[23:0] = AD[23:0]	24
1001	TCR2[23:0] = AD[23:0]	24	Reserved	-
1010	P[31:24] = AD[7:0]	8	Reserved	-

Table continues on the next page...

Table 2-69. Destination selection - T2ABD (continued)

T2ABD	First register set		Second register set	
	Selected register	Size	Selected Register	Size
1011	P[23:16] = AD[7:0]	8	Reserved	-
1100	P[15:8] = AD[7:0]	8	Reserved	-
1101	P[7:0] = AD[7:0]	8	Reserved	-
1110	TRR[23:0] = AD[23:0]	24	Reserved	-
1111	no destination selected[3]	24	Reserved	-

1. T2ABD=0010 with first register set also writes to MatchA or UDCM registers of the selected channel if field ERWA=0 (see Write Channel Match and UDCM Registers).
2. T2ABD=0011 with first register set also writes to MatchB register of the selected channel if field ERWB=0.
3. if no destination is selected, ALU flags are updated, although the result is lost.

2.14.9.2.2.1 Max Const Generation With T4BBS=111

When T4BBS = 111, BINV = 0, and CIN = 0, the value assigned to BS will be 0x800000, and not 0x0 as expected. See Generating "max" constant for detailed explanation about this.

2.14.9.2.2.2 Special T4ABS Source Operation: Read Match Registers

When T4ABS = 0101 and the source for T4ABS is selected from the second register set, the constant 0x00 is used as AS (8-bit size) and the following register transfer is performed in parallel as well: match registers of the selected channel (value in CHAN register) are copied to ERTA/ERTB registers, where ERTA receives the value of MatchA register and ERTB receives the value of MatchB register (see ER - Event Registers). Note that ALU destination can still be chosen by T2ABD in parallel. When ERTA or ERTB is selected by T2ABD, a parallelism issue arises (see ALU Operations and Read Match Registers).

2.14.9.2.2.3 CHAN_BASE as a source

Each channel has a parameter base address in SDM, which is configured in the CPBA field of eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2). CHAN_BASE, which represents a parameter address (CPBA*2), can be used as A-source using T4ABS=1010 when T4ABS selects a source from the second register set. In this case, CHAN_BASE is loaded into AS[13:2] to form the byte address (AS[23:14]=0, AS[1:0]=0). For example, in Indirect addressing mode, where the destination register is DIOB, CHAN_BASE is loaded into DIOB[13:2], which is the parameter address, and DIOB[13:0] represents the byte address. CHAN_BASE is the base address of the selected channel (given by CHAN register).

2.14.9.2.3 Flags Sampling Control

This section explains how the flags Z (zero), C (carry), N (negative) and V (overflow) are updated in an ALU operation. When there are post-ALU shift operations, the ALU Carry Out is not directly sampled in Carry flag, but passed to the post-ALU shifter (see Shift Operations). Since the size of source operands in ALU operations is variable, flags can be sampled as an operation of 8, 16 or 24 bits wide. The operation size selection is automatic, based on defined sizes of sources and destination, using the equation:

```
operation_size = minor(size_of(destination), greater(size_of(A-Source),
size_of(B-Source)))
```

Operation size can also be shown with the following table:

Table 2-70. Operation Size Determination

A Source	B Source	Destination	Operation Size
x	x	8 bits	8 bits
8 bits	8 bits	x	8 bits
16 or 24 bits	x	16 bits	16 bits
16 bits	8 or 16 bits	24 bits	16 bits
8 or 16 bits	16 bits	24 bits	16 bits
24 bits	x	24 bits	24 bits
x	24 bits	24 bits	24 bits

Note

Whenever BS = (constant) 0, its size is considered 8 bits, and all 24 bits in B-bus are set to 0. Therefore, all operations with BS = (constant) 0 have their size determined by AS and Destination only.

CCS field (1 bit) controls whether flags will be updated or not (Table 2-72). When CCS bit exists in a microinstruction, the operation size will be used to sample flags. In some microinstructions CCS field is replaced by CCSV (2 bits, Table 2-71). Flag sampling according to CCSV can be set as defined by the operation size, or fixed as 8 or 16 bit operations.

Note

For the post-ALU rotate right operation, CCSV also determines the rotate size: whether 8-bit, 16-bit, or determined by the operation size.

Table 2-71. Flag Sampling Using CCSV field

CCSV	Meaning
00	sample flags as an 8 bit operation
01	sample flags as a 16 bit operation
10	sample flags as defined by operation size
11	do not sample flags

Table 2-72. Flag Sampling Using CCS field

CCS	Meaning
0	sample flags as defined by operation size
1	do not sample flags

When neither CCS nor CCSV are present in the microinstruction, flags are not sampled. CCS and CCSV do not affect the Carry update on Exchange Bit operation (see Exchange Bit), but does control the N and Z flags.

2.14.9.2.4 B-Source Inversion

The data selected as second source (T4BBS) can be inverted (bitwise boolean NOT) before operation. This is controlled by microinstruction field BINV (1 bit, in the following table). A zero value for BINV activates B-source inversion.

Table 2-73. B-Source Inversion - BINV

BINV	Meaning
0	invert B-source[1]
1	keep B-source bus unchanged

1. Except on max-constant selection, see Generating "max" constant .

BINV also selects between *adc* or *sbc* enhanced ALU operation, using inverted C flag as carry-in besides BS inversion for *sbc* . Note that BINV does not invert carry in fixed-carry operations (see Table 2-74).

When BINV = 0, T4BBS = 111 and CIN = 0, the value assigned to BS is 0x800000, instead of 0x0. See Generating "max" constant for more details.

2.14.9.2.5 Carry-in Control

CIN field (1 bit, Table 2-74) controls the carry-in for addition/subtraction operations. Functionality of CIN field depends on the arithmetic operation selected by ALUOP. When ALUOP is not available in microinstruction, the operation selected is *add* . For carry-in control in MDU operations, see the following table.

Table 2-74. ALU Carry-In Control

operation	CIN=0	CIN=1
add (addition)	carry-in used is 1[1]	carry-in used is 0
adc (addition with carry)[2]	carry-in used is C flag	
sbc (subtraction with borrow)[3]	carry-in used is inverted C flag	

1. Except on max-constant selection, see Generating "max" constant .
2. Selected by ALUOP=11000 and BINV=1
3. Selected by ALUOP=11000 and BINV=0

2.14.9.2.5.1 Generating "max" constant

When T4BBS = 111, CIN = 0 and BINV = 0, BS is assigned to 0x800000 (called "max constant") instead of 0x000000. This is an exception for CIN and BINV fields: when "max constant" is selected, the carry in is 0 and B-source ("max constant" itself) is not inverted, neither the carry out.

"Max constant" is the value which, added to a time base value minus 1, gives the farthest wrapped time base value that satisfies a channel greater-equal comparison. See Enhanced Channels for more info.

2.14.9.2.6 Shift Operations

There are three types of shift operations: ALU, post-ALU and Shift Register. ALU shift operations are covered in ALU/MDU Operation Selection . Post-ALU and Shift Register are covered in the following sections.

2.14.9.2.6.1 Shift Register Operations

SR can be used as a general purpose register and it can easily shift-right its contents, combined or not to post-ALU shift operations. If field SRC (1 bit) in microcode is 0, SR will shift its contents 1 bit to the right according to the algorithmic description below. SR shifting operation depends also on SHF or ALUOP fields. ALUOP and SHF never exist both on the same microinstruction format.

Table 2-75. Shift Register Control - SRC

SRC	Meaning
0	shift right SR 1 bit
1	no shift

SR Operation:

```
SR[22:0] = SR[23:1];
if SHF == "01" or ALUOP == "10110" then
 SR[23] = ALU_OUT[0];
else
  SR[23] = 0;
endif;
```

2.14.9.2.6.2 Post-ALU shift operations

Post-ALU shift can be selected by SHF field (2 bits) or by some specific ALUOP field values. SHF and ALUOP fields are never both available in the same microinstruction format. When selecting post-ALU shift operation using ALUOP field, ALU will always add the sources before shifting the result.

Table 2-76. Post-ALU shift operation

Post ALU operation	SHF[1]	ALUOP
shift left (1 bit)	00	10101
shift right (1 bit)	01	10110
rotate right (1 bit)	10	10111
no shift/rotate	11	any other[2]

1. ALU performs AS+BS before shift/rotate for all SHF values.
2. Some ALUOP combinations perform shift/rotate, but not using the Post-ALU Shifter (see Table 1)

Carry flag is only updated when CCS or CCSV fields allow it (see Flags Sampling Control). Algorithmic descriptions of post-ALU shift operations are presented in ALU ADD operation with and without shifting .

2.14.9.2.7 Conditional ALU/MDU Operation Execution

The 3-bit field AS/CE allows conditional execution of arithmetic operation, as shown in the following table . The same field can also be used for overriding the size of A-Source (see A-Source Size Override).

Table 2-77. ALU/MDU Conditional Execution

AS/CE	Meaning
000	used for A-Source size override (see A-Source Size Override)

Table continues on the next page...

Table 2-77. ALU/MDU Conditional Execution (continued)

AS/CE	Meaning
001	
010	execute if C = 1
011	execute if C = 0
100	execute if Z = 1
101	execute if Z = 0
110	execute if N = 1
111	execute unconditionally/no size override

Other operations not related to ALU/MDU in the same microinstruction are not affected by the AS/CE field.

If a conditional operation is selected, there is no A-Source size override; similarly, when size override for A-Source is selected, the ALU/MDU operation executes unconditionally.

When a conditional ALU/MDU operation is not executed:

- the destination register is not updated. If the destination is CHAN, no actions associated with CHAN assignment occur (see Channel Selection Register - CHAN).
- the ALU and MDU flags are not updated.
- MDU does not start any operation, i.e., MACH and MACL are not updated.
- SR does not shift.
- T4ABS-selected read-match does not occur.

2.14.9.2.8 A-Source Size Override

Some values if the AS/CE field are used for A-Source Size Override, as shown in the following table.

Table 2-78. A-Source Size Override

AS/CE	Meaning
000	A-source size override to 8 bits
001	A-source size override to 16 bits
010	used for conditional execution (see Conditional ALU/MDU Operation Execution)
011	
100	
101	

Table continues on the next page...

Table 2-78. A-Source Size Override (continued)

AS/CE	Meaning
110	
111	execute unconditionally/no size override

Register size override zero-pads an overridden source to 24-bits (if no sign extension is performed, see A-source Sign Extension) and affects operation size calculation. When register source is wider than size override, most significant bits of selected register are not used as A source (zeros are used instead). When size override is wider than selected register, register value is padded to zeros.

When size override is used with MDU operations, it affects only the operand values, but not the operation size: MDU operation size is fully determined by the operation definition (fields ALUOP, ALUOPI).

Table 2-79. AS/CE field A source size override funcionality

Size Override	Size of Selected Register	AS Value[1]
8 bits	8 bits	reg[7:0]
8-bits	16 bits	reg[7:0]
8bits	24 bits	reg[7:0]
16 bits	8 bits	reg[15:0]
16 bits	16 bits	reg[15:0]
16 bits	24 bits	reg[15:0]

1. All values are zero-padded to 24 bits

2.14.9.2.9 A-source Sign Extension

The SEXT microinstruction field forces sign extension of A source according to the size of A operand, either overridden or not by AS/CE field, according to the following table. The sign is taken from the size-overridden value, not the original one.

Table 2-80. A source Sign Extension

SEXT	Meaning
0	extends sign of A source from its size to 24-bits
1	does not extend sign of A source

A-source sign is not extended in microinstructions without SEXT field, even if AS/CE field is present.

2.14.9.2.10 ALU/MDU Operation Selection

When field ALUOP is available in microinstruction, enhanced ALU operations shown in the following table can be performed, otherwise addition is performed. The ALU operations are defined in ALU and Post-ALU Shifter. The MDU operations are defined in MAC and Divide Unit (MDU).

Table 2-81. ALU Operation Selection - ALUOP

ALUOP	Operation	Comment
00000	AS mults BS[7:0]	signed multiplication
00001	AS multu BS[7:0]	unsigned multiplication
00010	AS fmults BS[7:0]	signed fractional multiplication
00011	AS fmultu BS[7:0]	unsigned fractional multiplication
00100	AS mults BS[15:0]	signed multiplication
00101	AS multu BS[15:0]	unsigned multiplication
00110	AS fmults BS[15:0]	signed fractional multiplication
00111	AS fmultu BS[15:0]	unsigned fractional multiplication
01000	AS mults BS[23:0]	signed multiplication
01001	AS multu BS[23:0]	unsigned multiplication
01010	AS macs BS[23:0]	signed multiply-accumulate
01011	AS macu BS[23:0]	unsigned multiply-accumulate
01100	AS div BS[7:0]	unsigned division by 8-bit value
01101	AS div BS[15:0]	unsigned division by 16-bit value
01110	AS div BS [23:0]	unsigned division by 24-bit value
01111	n.a.	RESERVED
10000	AS[23:0] | BS[23:0]	24 bit bitwise OR
10001	AS[23:0] ^ BS[23:0]	24 bit bitwise XOR
10010	AS[23:0] & BS[23:0]	24 bit bitwise AND
10011	abs(AS)	absolute value of AS
10100	AS + BS	arithmetic addition
10101	(AS + BS) shl 1	arithmetic addition with 1-bit post-ALU shift left. (Post-ALU shift operations)
10110	(AS + BS) shr 1	arithmetic addition with 1-bit post-ALU shift right (Post-ALU shift operations)
10111	(AS + BS) ror 1	arithmetic addition with 1-bit post-ALU rotate right (Post-ALU shift operations)
11000	AS adc/sbc BS[1]	addition/subtraction with C flag (Carry-in Control)
11001	AS shl (2^(BS[1:0]+1))	AS is shifted left: 2 bits for BS=0; 4 for BS=1; 8 for BS=2; 16 for BS=3
11010	AS shr (2^(BS[1:0]+1))	AS is shifted right: 2 bits for BS=0; 4 for BS=1; 8 for BS=2; 16 for BS=3
11011	AS ror (2^(BS[1:0]+1))	AS is rotated right: 2 bits for BS=0; 4 for BS=1; 8 for BS=2; 16 for BS=3
11100	AS exch BS[4:0]	exchange C flag and AS bit determined by BS (Exchange Bit)
11101	AS setb BS[4:0]	set bit in AS determined by BS [2]
11110	AS clrb BS[4:0]	clear bit in AS determined by BS[3]
11111	n.a.	RESERVED

1. Addition/Subtraction is selected by field BINV (see B-Source Inversion)

2. In setb and clrb operations, the register that drives A source is not changed, unless selected as destination of the operation.
3. In setb and clrb operations, the register that drives A source is not changed, unless selected as destination of the operation.

2.14.9.2.11 Operations With Immediate Data

Immediate data can be used with some specific microinstruction formats. eTPU microcode allows 8-, 16- or 24-bit immediate data. Immediate data is loaded as B-source, so T4BBS field is not available. When using 24-bit immediate, an add is performed with A-source = 0 and flags are not updated. ALU operations are available with 8-bit immediate, although the field that selects ALU operation in this case is ALUOPI.

2.14.9.2.11.1 24-bit Immediate Destination

When using 24-bit immediate data, the destination register is selected by T2D field (2 bits), according to the following table.

Table 2-82. 24-bit Immediate Destination - T2D

T2D	Target Register
00	P[23:0]
01	A
10	SR
11	DIOB

2.14.9.2.11.2 Enhanced ALU Operations With Immediate Data

Enhanced operations with immediate data, selected by ALUOPI (5 bits) are allowed only with an 8 bit immediate operand (see the following table).

Table 2-83. ALU Operation Selection With Immediate Data - ALUOPI

ALUOPI	Operation	Comment
00000	AS mults #imm8	signed multiplication
00001	AS multu #imm8	unsigned multiplication
00010	AS fmults #imm8	signed fractional multiplication
00011	AS fmultu #imm8	unsigned fractional multiplication
00100	AS div #imm8	unsigned division
00101	n.a.	reserved
00110	n.a.	reserved
00111	n.a.	reserved
01000	AD[7:0] = AS[7:0] | #imm8, AD[23:8] = AS[23:8]	bitwise OR

Table continues on the next page...

Table 2-83. ALU Operation Selection With Immediate Data - ALUOPI (continued)

ALUOPI	Operation	Comment
01001	AD[7:0] = AS[7:0] ^ #imm8, AD[23:8] = AS[23:8]	bitwise XOR
01010	AD[7:0] = AS[7:0] & #imm8, AD[23:8] = AS[23:8]	bitwise AND
01011	AD[7:0] = AS[7:0] & #imm8, AD[23:8] = 0×0	bitwise AND with clear
01100	AD[15:8] = AS[15:8] | #imm8, AD[23:16] = AS[23:16], AD[7:0] = AS[7:0]	bitwise OR
01101	AD[15:8] = AS[15:8] ^ #imm8, AD[23:16] = AS[23:16], AD[7:0] = AS[7:0]	bitwise XOR
01110	AD[15:8] = AS[15:8] & #imm8, AD[23:16] = AS[23:16], AD[7:0] = AS[7:0]	bitwise AND
01111	AD[15:8] = AS[15:8] & #imm8, AD[23:16] = 0×0, AD[7:0] = 0×0	bitwise AND with clear
10000	AD[23:16] = AS[23:16] | #imm8, AD[15:0] = AS[15:0]	bitwise OR
10001	AD[23:16] = AS[23:16] ^ #imm8, AD[15:0] = AS[15:0]	bitwise XOR
10010	AD[23:16] = AS[23:16] & #imm8, AD[15:0] = AS[15:0]	bitwise AND
10011	AD[23:16] = AS[23:16] & #imm8, AD[15:0] = 0x0	bitwise AND with clear
10100	AS + #imm8	arithmetic addition
10101	(AS + #imm8) shl 1	arithmetic addition with 1-bit shift left.
10110	(AS + #imm8) shr 1	arithmetic addition with 1-bit shift right
10111	(AS + #imm8) ror 1	arithmetic addition with 1-bit rotate right
11000	n.a.	reserved
11001	AS shl (2^(#imm8[1:0]+1))	AS is shifted left: 2 bits for #imm8=0; 4 for #imm8=1; 8 for #imm8=2; 16 for #imm8=3
11010	AS shr (2^(#imm8[1:0]+1))	AS is shifted right: 2 bits for #imm8=0; 4 for #imm8=1; 8 for #imm8=2; 16 for #imm8=3
11011	AS ror (2^(#imm8[1:0]+1))	AS is rotated right: 2 bits for #imm8=0; 4 for #imm8=1; 8 for #imm8=2; 16 for #imm8=3
11100	AS exch #imm8[4:0]	exchange C flag and AS bit determined by #imm8[4:0] (see Exchange Bit)
11101	n.a.	reserved
11110	n.a.	reserved
11111	n.a.	reserved

2.14.9.3 Channel Control and Configuration Microoperations

Channel Control and Configuration fields set configuration values in the channel logic of the channel selected by the CHAN register, except fields LSR and CIRC.

2.14.9.3.1 Channel flags operations

Each channel has two associated hardware flags, called Channel Flag 0 and Channel Flag 1. Microcode field FLC (3 bits) allows them to be set or cleared, as shown in the following table. These flags can be tested by microcode as a branch condition, and may also influence in the Entry Point taken, allowing fast state decoding. For more details, see Entry Points.

Table 2-84. P flags operation - FLC

FLC	Meaning
000	Clear flag0
001	Set flag0
010	Clear flag1
011	Set flag1
100	Copy flag1:flag0 from P[25:24]
101	Copy flag1:flag0 from P[27:26]
110	Copy flag1:flag0 from P[29:28]
111	No operation

2.14.9.3.2 Comparator and Time Base Selection

TBSA and TBSB fields (4-bit wide each) are used to configure the type of the comparator and the time bases used for match or capture (see the following tables). TBSA can also be used to control the Output Buffer Enable signal (See Pin Control Registers).

Table 2-85. Time Base Selection A - TBSA

TBSA[3] = 0	**TBSA bit**	**2**	**1**	**0**
	bitfield	**Comparator selection**	**Capture selection**	**Match TB selection**
	0	greater or equal	TCR1	TCR1
	1	equal-only	TCR2	TCR2
TBSA[3] = 1	**action**	**2**	**1**	**0**
	set OBE=1	0	0	0

Table continues on the next page...

Table 2-85. Time Base Selection A - TBSA (continued)

	set OBE=0	0	0	1
	do nothing	1	1	1
	reserved	all other values		

Table 2-86. Time Base Selection B - TBSB

TBSB[3] = 0	**TBSB bit**	**2**	**1**	**0**
	bitfield	**Comparator selection**	**Capture selection**	**Match TB selection**
	0	greater or equal	TCR1	TCR1
	1	equal-only	TCR2	TCR2
TBSB[3] = 1	action	**2**	**1**	**0**
	do nothing	1	1	1
	reserved	all other values		

2.14.9.3.3 Transition Detection and Pin Action Control

IPACA/B and OPACA/B fields are used to configure transition detection sensitivity (for the channel input signal) or output pin action control (for the channel output signal), as defined in the table below. IPACA and IPACB have the same format, where IPACA is related to Match A and first transition detection, and IPACB to Match B and second transition detection. The same applies in analogue way to OPACA and OPACB.

For the output signal, configuring OPAC registers does not change the current signal state, but defines the action to be done when a match or transition detection occurs. See Match Recognition and Channel Modes on Output Signal Generation for more information.

IPACA/B=1xx also enables assertion of MRLA/B during Time Slot Transition. See Match Recognition.

Table 2-87. Input and Output Pin Action Control - IPACA/B and OPACA/B

value	IPAC meaning	OPAC meaning
000	Do not detect transitions	Do not change output signal
001	Detect rising edge only	Match[1] sets output signal high
010	Detect falling edge only	Match[1] sets output signal low
011	Detect both edges	Match[1] toggles output signal
100	Detect input signal = 0 on Match[1]	Transition detection sets output signal low
101	Detect input signal = 1 on Match[1]	Transition detection sets output signal high
110	reserved	Transition detection toggles output signal
111	do not change IPAC	do not change OPAC

1. Match A is used for IPACA/OPACA, and Match B for IPACB/OPACB.

2.14.9.3.4 Immediate Pin State Control

It is possible to change output signal state immediately by using PSC (2 bits) and PSCS (1 bit) fields.

Table 2-88. Immediate Pin State Control - PSC and PSCS

PSC	PSCS	Meaning
00	0	set signal as specified by OPACA (see Transition Detection and Pin Action Control)
00	1	set signal as specified by OPACB (see Transition Detection and Pin Action Control)
01	x	set signal high
10	x	set signal low
11	x	don't change signal state

2.14.9.3.5 Write Channel Match and UDCM Registers

Match registers can have their values changed using ERWA and ERWB fields (1 bit each). They also set their respective MRLE register (see Match Recognition).

ERWA can also be used to program the UDCM register (see UDCM - User Defined Channel Mode). The field CMW selects where the contents of ERTA is copied when ERWA is active (see the following table).

Table 2-89. Write MatchA/B - ERWA/B

Field	CMW	Value	Action
ERWA	1	0	write ERTA value in MatchA. Enable matches for MatchA register (MRLEA=1)
	0	0	write ERTA value into UDCM
	1	1	don't change UDCM, MatchA and MRLEA
	0	1	reserved
ERWB	1	0	write ERTB value in MatchB. Enable matches for MatchB register (set MRLEB=1)
	1	1	don't change MatchB and MRLEB
	0		
	0	0	reserved

If ERTA or ERTB is a destination of an ALU operation and, at the same time, the respective ERWA/B field is active, the new ERTA value is the one written into the MatchA/B register or the UDCM register.

2.14.9.3.6 Clear Transition/Match Event Registers

Flags MRLA, MRLB, TDLA and TDLB (see ER - Event Registers) indicate the state of matches and transitions detected in the selected channel, and it is possible to clear those flags using the microcode fields MRLA, MRLB (1 bit each) and TDL (1 or 2 bits, depending on the format). The flags cleared by these microcode fields are the actual channel flags, and also the ones sampled into the branch logic.

TDL can be one or two bits wide, depending on the microinstruction format (see Microinstruction formats). Two-bit TDL allows independent clearing of TDLA and/or TDLB . Table 2-91 defines the two-bit TDL field.

Table 2-90. Clear Transition/Match Event Registers - MRLA/B, TDL

Field	Meaning
MRLA	0 = clear MRLA event register, 1 = don't change
MRLB	0 = clear MRLB event register, 1 = don't change
TDL (1 bit)	0 = clear TDLA and TDLB flags, 1 = don't change

Table 2-91. Independent TDLA/B clear - two-bit TDL

value	Meaning
0 0	clear TDLA
0 1	clear TDLB
1 0	clear both TDLA and TDLB
1 1	do not clear TDLA or TDLB

2.14.9.3.7 Disable Matches

Microcode field MRLE (1 bit) allows disabling matches on channel selected by CHAN register, for both MatchA and MatchB registers, by clearing their respective MRLE bits. Matches can be enabled for each Match register using ERWA and ERWB fields (see Write Channel Match and UDCM Registers).

Table 2-92. Disable Matches - MRLE

MRLE	Meaning
0	Disable matches for Match A and Match B
1	don't change match enabling

Some instruction formats have a two-bit MRLE field (see Microinstruction formats) which allows independent disabling of Matches 1 and 2, as shown in the following table.

Table 2-93. Two-bit MRLE

MRLE	Meaning
0 0	Disable Match A (clear MRLEA)
0 1	Disable both Matches (clear MRLEA and MRLEB)
1 0	Disable Match B (clear MRLEB)
1 1	nop

2.14.9.3.8 Disable Match and Transition Service Requests

Microcode field MTD (2 bits) disables match and transition service requests for the selected channel. MTD does not disable Link Service Request and Host Service Request. MTD sets or resets register SRI (for more details see SRI - Match/Transition Service Request Inhibit Latch) and TCCEA (see TCCEA - Transition Continuous Capture Enable) .

Table 2-94. Disable Match and Transition Service Request - MTD

MTD	Action on SRI	Action on TCCEA
00	SRI = 0: enable service requests for match and transition	TCCEA = 0: disable transition captures[1] when TDLA = 1
01	SRI = 1: disable service requests for match and transition	
10	SRI = 1: disable service requests for match and transition	TCCEA = 1: enable transition captures[2] when TDLA = 1
11	don't change	

1. Disables only captures on transition events specified by IPACA.
2. Enables only captures into CaptureA register, on transition events specified by IPACA.

2.14.9.3.9 Predefined Channel Modes

Microcode PDCM field (4 bits) defines the channel mode (see Channel Modes).

PDCM coding is shown in the table below. Note that PDCM bit 0 selects between Single Transition (PDCM[0]=0) and Double Transition (PDCM[0]=1) predefined modes.

PDCM is also used to select the user-defined channel mode, defined by the channel register UDCM (see UDCM - User Defined Channel Mode).

Table 2-95. Predefined Channel Modes

PDCM	Channel mode
0000	em_b_st
0001	em_b_dt
0010	em_nb_st
0011	em_nb_dt

Table continues on the next page...

Table 2-95. Predefined Channel Modes (continued)

PDCM	Channel mode
0100	m2_st
0101	m2_dt
0110	bm_st
0111	bm_dt
1000	m2_o_st
1001	m2_o_dt
1010	**user-defined channel mode**
1011	reserved
1100	sm_st
1101	sm_dt
1110	sm_st_e
1111	keep current channel mode

2.14.9.3.10 Channel Interrupt and Data Transfer Requests

Microcode can issue Interrupt Requests, Data Transfer Requests and Global Exception through CIRC field. For more information see Interrupts and data transfer requests.

Table 2-96. Channel and Data Transfer Requests - CIRC

CIRC	Meaning
000	Channel Interrupt Request from selected channel
001	Data Transfer Request from selected channel
010	Channel Interrupt and Data Transfer requests from selected channel
011	Channel Interrupt and Data Transfer requests from serviced channel
100	Channel Interrupt Request from service channel
101	Data Transfer Request from service channel
110	Global Exception
111	don't request interrupt

2.14.9.3.11 Clear Link Service Request

Microcode LSR field (1 bit) is used to clear the Link Service Request flag of the serviced channel (may not be the one selected by CHAN). The LSR branch condition is always cleared, but not the Link Service Request, if another channel link was received by the serviced channel during the executing thread. See Channel Link for more information.

Table 2-97. Link Service Request Negation Control - LSR

LSR	Meaning
0	clear Link Service Request (flag LSR)
1	don't change

2.14.9.4 Flow Control Microoperations

eTPU has *jump* and *call* microoperations to change microcode flow. Besides, eTPU has *dispatch jump* and *dispatch call* that can be used to implement a jump table. In *call* (or *dispatch call*) microoperation, the return address is saved in RAR register. If nested sub-routine calls are necessary, return address values have to be saved in a stack, usually implemented with DIOB register.

Flow Control microoperations are also provided to repeat a given microinstruction, to finish the current thread execution, and to halt the microengine.

2.14.9.4.1 Ending Current Thread - END

Microcode END field (1 bit) finishes current thread and allows other channels to be serviced. If END field is 0, the current instruction is completed and the thread is finished. END = 1 has no effect, and the next microinstruction is executed Any MDU operation (see MAC and Divide Unit (MDU)) that could be still pending when the thread is finished is left incomplete. END also releases any semaphore locked by the Engine.

2.14.9.4.2 Branch Operations

Branch operations can be **jump** or **call** . The target address of *jump* or *call* microoperations is always immediate and absolute. Branch microoperation is affected by FLS field (refer to Flush Pipeline).

2.14.9.4.2.1 Selecting Jump or Call Microoperations

The only difference between *jump* and *call* microoperations is that when a *call* is executed the value of PC or PC+1 (depending on flush, see Flush Pipeline) is saved in RAR register. The microcode field J/C (1 bit) selects whether *jump* or a *call* is executed, according to the following table.

Table 2-98. Jump / Call Selection - J/C

J/C	Meaning
0	jump
1	call

2.14.9.4.2.2 Branch Target Address

Microcode BAF field (14 bits) indicates the absolute address of a *jump/call target*.

2.14.9.4.2.3 Conditional/unconditional branch

Jump and *call* can be conditional or unconditional, depending on the BCC (6 bits) and BCF (1 bit) fields, as shown in Table 2-99 and Table 2-100 . BCF determines whether branch is taken when condition specified by BCC is true or false. When a branch condition uses the channel flags, the channel context is related to the channel number written in CHAN register.

Table 2-99. Branch Condition Inversion - BCF

BCF	Meaning
0	branch if condition determined by BCC is false
1	branch if condition determined by BCC is true

Table 2-100. Branch Condition Selection - BCC

BCC	Meaning	BCC	Meaning
001110	Flag 0	001111	Flag 1
100000	V ALU flag	110000	PSS channel flag
100001	N ALU flag	110001	PRSS channel flag
100010	C ALU flag	110010	"Less Than" ALU flag combination (signed) [1]
100011	Z ALU flag	110011	"Lower or Equal" ALU flag combination (unsigned)[2]
100100	MV MDU flag	110100	P[24]
100101	MN MDU flag	110101	P[25]
100110	MC MDU flag	110110	P[26]
100111	MZ MDU flag	110111	P[27]
101000	TDLA channel flag	111000	P[28]
101001	TDLB channel flag	111001	P[29]
101010	MRLA channel flag	111010	P[30]
101011	MRLB channel flag	111011	P[31]
101100	LSR channel flag	111100	PSTO channel flag
101101	MB flag MDU flag	111101	PSTI channel flag

Table continues on the next page...

Table 2-100. Branch Condition Selection - BCC (continued)

BCC	Meaning	BCC	Meaning
101110	FM[1] channel flag	111110	SMLCK semaphore flag
101111	FM[0] channel flag	111111	false
all other values reserved			

1. "less than" is a signed comparison, equal to the xor between ALU flags V and N; e.g., 0 < 0xFFFFFF tests as false (0 < -1).
2. "lower equal" is an unsigned comparison, equal to Z or C; e.g., 0 < 0xFFFFFF tests as true.

2.14.9.4.3 Dispatch Microoperation

Dispatch microoperation is an unconditional branch where the target address is always PC+P[31:24] (unsigned). Dispatch is affected by FLS field (refer to Flush Pipeline). Dispatch microoperation is defined by R/D field (2 bits, see the following table). Field R/D can also be used to define return from sub-routine (see Return from subroutine).

Table 2-101. Return and Dispatch - R/D

R/D	Meaning
00	return from subroutine (see Return from subroutine)
01	dispatch jump
10	dispatch call
11	don't change microinstruction flow

2.14.9.4.4 Return from subroutine

When a subroutine *call* or a *dispatch call* microoperation is executed, the return address is saved in RAR register. To return from a subroutine a microoperation is available to load the contents of RAR register back to the PC. Fields R/D (2 bits) or RTN (1 bit, see the table below) can be used to return from subroutine. For R/D field, see Table 2-101.

Return from subroutine microoperation is affected by FLS (see Flush Pipeline) when field R/D is used. Return execution through RTN always flushes the pipeline.

Table 2-102. Return from subroutine - RTN

RTN	Meaning
0	Return with pipeline flush
1	Do not return

2.14.9.4.5 Flush Pipeline

When a branch, dispatch or subroutine return microoperation is executed, the next microinstruction can be executed unconditionally before the flow change takes effect, since microengine has a three-stage pipeline. Executing the next microinstruction after a branch maximizes execution performance. This feature is controlled by field FLS (1 bit, see the table below). When FLS=0 the pipeline is flushed, so the next microinstruction placed after a branch is decoded as NOP if the branch is taken. If FLS=1, the microinstruction placed after the branch is executed, either if the branch is taken or not, as shown in the figure below.

Flush also controls which value is stored in RAR in a call: in case of no flush, it is the address of the branch/dispatch instruction + 2, even if RAR is the ALU destination of the instruction after the call; in case of a flush, it is the address of the instruction following branch/dispatch.

If a branch with no flush is followed by another branch with no flush, the instructions are executed in the following order:

1. first branch
2. second branch
3. first branch's destination instruction
4. second branch's destination instruction, and the flow proceeds normally from then on

The destination of the first branch must not be another flow changing instruction (branch, return or dispatch). Similar flows apply when returns or dispatches are used instead of branches. This scheme can be used to implement quick table look-ups with a dispatch replacing the first branch, for instance.

Table 2-103. Flush Pipeline - FLS

FLS	Meaning
0	flush pipeline when jump / call / dispatch jump / dispatch call / return is executed
1	do not flush pipeline when jump / call / dispatch jump / dispatch call / return is executed

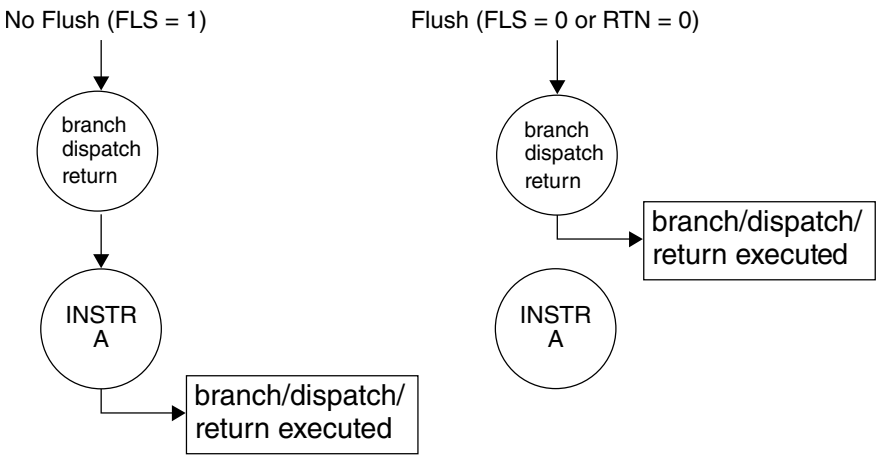


Figure 2-42. Flush Pipeline

2.14.9.4.6 HALT microinstruction

HALT is a microinstruction provided to implement software breakpoints (see Software breakpoints). Note that HALT is coded as a microinstruction format, not a field (see Microinstruction formats). The execution of this instruction puts the microengine in halt state. For more information about the implications of microengine halt state, see Microengine Halt State . HALT is valid only if software breakpoints are enabled at the Debug interface . If software breakpoints are not enabled, HALT executes as a NOP and is treated as an Illegal Instruction (see Illegal instructions).

2.14.9.4.7 NOP microinstruction

There is not a unique microinstruction with an assigned opcode to do No Operation. NOP microinstruction is achieved through any of the formats shown on Microinstruction formats where the user can assign to each individual field the corresponding value for "No Operation". However, to prevent future impacts of instruction changes on object code compatibility, the instruction value 0x4FFFFFFF should always be used for NOP.

2.14.9.5 Illegal instructions

An instruction is considered illegal if any reserved field value is used, including when the fields marked rsv in the instruction formats (see Microinstruction formats are assigned value 0. A HALT instruction is exceptionally considered illegal when executed with software breakpoints disabled (see HALT microinstruction). Global Exception may be issued up to two microcycles after instruction fetch. The execution results of an illegal instruction on the microengine, channel logic or host interface are unspecified, except for the HALT case.

If the microengine decodes an illegal instruction, the following actions are taken:

- A Global Exception is issued.
- The ILF1/2 flags in eTPU Module Configuration Register (eTPU_SCR_MCR) are set to indicate this occurrence to the Host.
- A breakpoint occurs, if NDEDI is present and configured to do so (see the NDEDI details for more information).

2.14.9.6 Microinstruction Parallelism Issues

This section clarifies parallelism issues that arise when two non-commutative microoperations appear in the same microinstruction.

2.14.9.6.1 ALU Operations and Read Match Registers

ALU operations have only one destination register, but there is one case where source selection determines destination: read Match register in ERTA and ERTB registers. In this case if ALU destination is ERTA or ERTB a conflict arises. The ALU destination value overwrites the value read from the match registers.

2.14.9.6.2 ALU and SDM Operations

P and DIOB registers can be selected as destination by both ALU and SDM (read) microoperations in the same microinstruction. Since P and DIOB update from SDM data happens after P and DIOB update for ALU/MDU microoperations, the data read from SDM remains in P or DIOB after an operation when one of them is specified as destination for both ALU and SDM microoperations. In this case, the value loaded into P or DIOB is the one read from SDM. However, the ALU operation is executed and its flags are updated accordingly.

When P or DIOB is destination of an SDM read and also an ALU source at the same microinstruction, the value before the read is used for the ALU operation.

If DIOB is the ALU destination and P is loaded from SDM or vice-versa, no conflict occurs, and the result is the same as if operations occurred separately.

All the above also applies to Zero SDM operations.

When using P or DIOB as destination for ALU operations and also as source for a SDM write operation, the data written in SDM is the one calculated by ALU, which means it is possible to calculate a value and write it in an SDM address using only one microinstruction.

The old value of DIOB or the old value minus 4 (pre-decrement) is always used when DIOB is selected as address (indirect address mode), no matter if DIOB is selected as destination of either the SDM or ALU. For the value loaded into DIOB, refer to the following table.

Table 2-104. DIOB load from SDM and ALU

DIOB selected as SDM read destination?	DIOB selected as ALU destination?	DIOB load value
no	no	DIOB, --DIOB (pre-decrement), or DIOB++ (post-increment)
yes	no	SDM read data (post-inc and pre-dec ignored)
yes	yes	SDM read data (post-inc, pre-dec and ALU result ignored)
no	yes	ALU result (post-inc an pre-dec ignored)

When CHAN is the destination of an ALU operation in parallel with a channel-relative SDM access, the CHAN value used to calculate the address is the one before the ALU operation.

2.14.9.6.3 ERTA/B as ALU destination and ERWA/B

The value in ERTA and ERTB registers can be written in match registers of the selected channel by using fields ERWA and ERWB (Write Channel Match and UDCM Registers). If, at the same microinstruction, ERTA or ERTB is destination of an ALU/MDU microoperation, the value written in Match registers is the ALU/MDU result.

The same applies to UDCM when ERTA is the destination of an ALU operation and instruction fields ERWA and CMW are active .

If an ALU operation occurs in parallel with ERWA/B but ERTA/B are not the destination of an ALU/MDU operation, then UDCM and MatchA/B receives the ERTA/B value.

2.14.9.6.4 ERWA/B and MRLE

ERWA/B automatically sets the MRLEA/B channel latch, respectively (see Write Channel Match and UDCM Registers). Microinstruction fields ERWA/B independently sets MRLEA/B channel flags, regardless of MRLE.

2.14.9.6.5 CHAN assignment, Read Match and ERWA/B

When CHAN is a destination of an ALU operation it causes a read of the CaptureA/B register values into ERTA/B. The Capture registers loaded into ERTA/B are selected by the new CHAN value. The value of the CaptureA/B registers overwrites any read-match commanded simultaneously.

If CHAN assignment happens with an ERWA/B operation in the same instruction, the updated Match register(s) belong to the new selected channel.

2.14.9.6.6 Read Match and ERWA/B

If a read match operation is executed with ERWA/B in the same microinstruction, the MatchA/B registers receive the old values of ERTA/B, and the ERTA/B registers receive the old MatchA/B values simultaneously, i.e.: ERTA/B and MatchA/B swap their values.

If ERTA/B is the destination of an ALU operation at the same instruction, MatchA/B gets the ALU result (see ERTA/B as ALU destination and ERWA/B), but the ERTA/B not being written still receives the old MatchA/B values.

Note

Read match, ERWA/B and CHAN assignment can be active at the same instruction. Combining rules CHAN assignment, Read Match and ERWA/B and Read Match and ERWA/B, the result is: ERTA/B receives the CaptureA/B values of the new CHAN value, and MatchA/B of the new channel receives the old ERTA/B value(s).

2.14.9.6.7 Stack accesses and ALU operations

Post-increment is ignored in a stack operation (field STC) if DIOB is loaded from SDM: DIOB keeps the value read from SDM. Pre-decrement is ignored in a stack operation (field STC) if DIOB is destination of an ALU operation, for DIOB load value, but not for DIOB as address. Post-increment/pre-decrement remains valid in all other situations. These rules can be summarized in the following equivalent C code, and in Table 2-104.

```
DIOB = *DIOB[15:2];       // read without posinc/predec
*DIOB[15:2] = DIOB;       // write without posinc/predec
```

```
DIOB = *(--DIOB[15:2]);  // read with predec
DIOB = *DIOB[15:2];      // read with posinc (ignored)
*(--DIOB[15:2]) = DIOB;  // write with predec
*DIOB[15:2]++ = DIOB;    // write with posinc (value written is before
increment)
```

2.14.9.6.8 SRC and ALU/MDU operations

If operation SRC is active (field SRC = 0) and register SR is selected as destination of an ALU operation, the value of the ALU operation prevails over the shifted value.

The value of SR used as source in the ALU/MDU operation is the one before the shift.

2.14.9.6.9 Semaphore lock/free and SMLCK branch condition

When the SMLCK branch condition is tested at the same microinstruction of a semaphore lock or free, the condition is evaluated after the semaphore action (either free or lock) is taken.

2.14.9.6.10 Dispatch and SDM Read

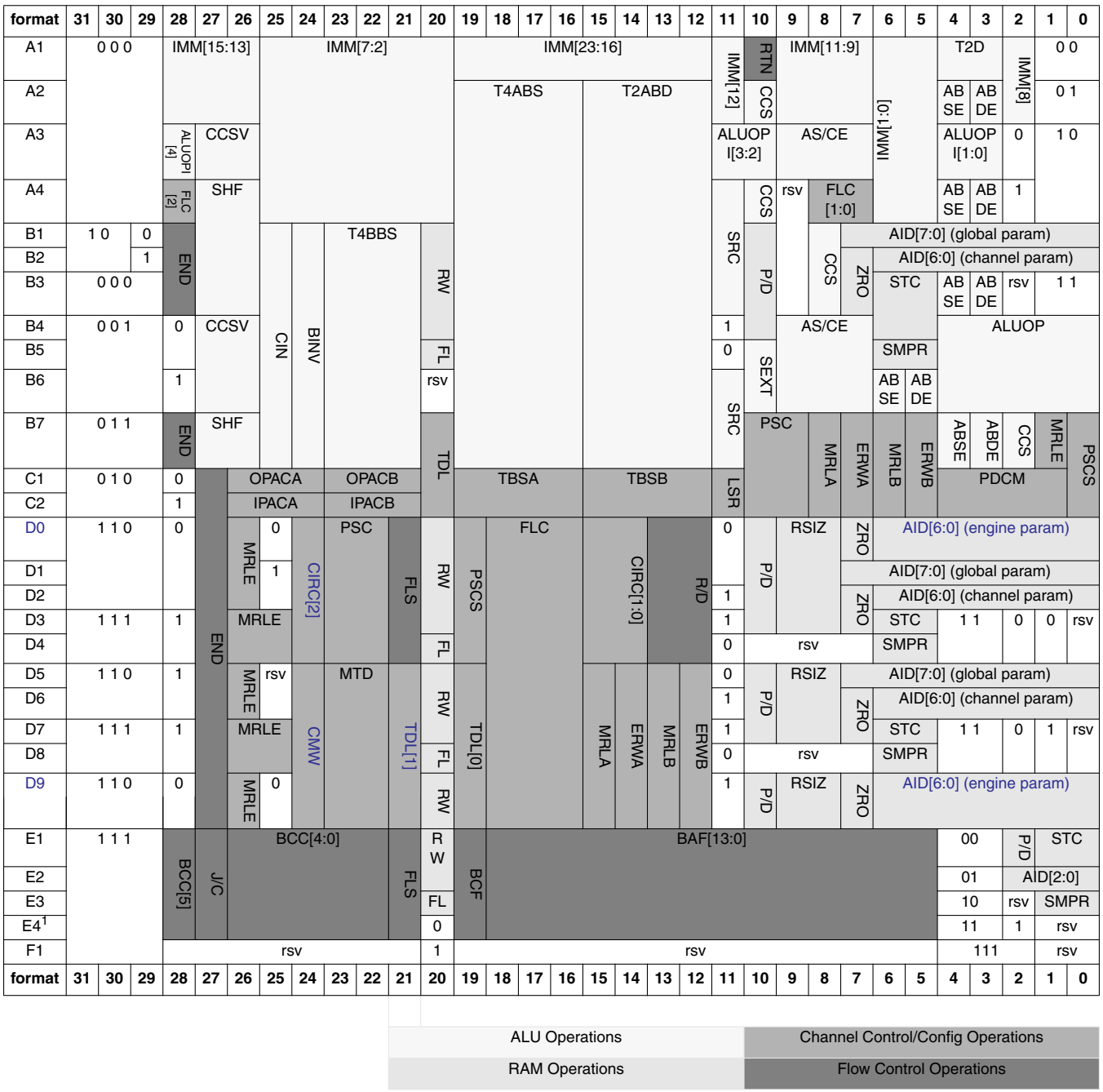
When the most significant byte of P is read from SDM (read 8 msb bits or 32 bits) and a dispatch instruction is executed simultaneously, the dispatch target address is calculated upon the P value before the read.

2.14.9.6.11 CHAN Assignment, PSC/PSCS, and clear MRLEA/B, MRLA/B, TDLA/B

When clear MRLEs, MRLA/B or TDLs is done and a CHAN assignment is done at the same time, the flag selected by the old CHAN value is cleared in the channel, but the branch conditions receive the state of the flags selected by the new CHAN.

When a pin action is commanded through PCS/PSCS and a CHAN assignment is done simultaneously, the output signal affected is selected with the old CHAN value.

2.14.9.7 Microinstruction formats



NOTES:
1 format E4 is deprecated; the same result can be obtained with format E1 with field STC=11

Figure 2-43. Microinstruction formats

All bits marked "rsvd" are reserved, and must be coded as 1.

2.14.10 Shared Subsystems support

Shared Subsystems support allows eTPU to control other modules, either independently or sharing control with the host. A notable example of shared subsystem is the Reaction Module (see the Reaction Module chapter for details), which is connected to the eTPU as shown in the figure below. The host CPU accesses the Reaction Module through the eTPU, which can also access the same registers.

When the SSMAE field in eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) or eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2) is 1, the eTPU channel's threads can control shared subsystems by accessing their registers, which are mapped onto the higher 16 Kbytes of the microcode data address space (byte addresses/DIOB values 0x4000 to 0x7FFF, equivalent to parameters 0x1000 to 0x1FFF), as shown in Figure 2-45. The shared subsystem registers are accessible by microcode only with the indirect address mode (see SDM Addressing Modes).

When the SSMAE field in eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) or eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2) is 0, the channel's threads can access only the SDM, mirrored throughout the whole DIOB address range.

The same Shared Subsystem register address space is accessible by the host through the Shared Subsystem Area within the eTPU memory map (see Figure 2-45 and Memory map), allowing shared control and/or supervision between the host and the eTPU. The arbitration mechanism between host and eTPU accesses is the same used for SDM accesses (see SDM arbitration). The CDC (see Coherent Dual-parameter Controller (CDC)) cannot be used for coherent transfers between SDM and the Shared Subsystem Area, or within the Shared Subsystem Area itself.

Note

The detailed register map within the Shared Subsystem Area depends on the subsystem module.

A Shared Subsystem may be divided into submodules or channels, hereafter called **Shared Subsystem Units** (or **Shared Units** for short), which can be individually controlled by an eTPU function (see Functions and threads)

A Shared Unit is logically linked with an eTPU channel through the eTPU_SSIR_ETPUSSSIR*n* configuration[34]. There is one eTPU_SSIR_ETPUSSSIR register for each Shared Unit to configure its interface with the eTPU (see eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR*n*)). The Shared Unit uses

34. However, in the case of Reaction Module, the eTPU_SSIR_ETPUSSSIR*n*configuration does not determine which eTPU channel output enables the Reaction Channel modulation; this is defined in the Reaction Module configuration.

the Link Service Request (LSR) mechanism to request service to the eTPU. The Link Service Request is targeted to the channel/engine configured in LENG and LCHAN. The function assigned to this channel must be written to service the Shared Unit. The Shared Unit link service request may have the same source as an interrupt to the host.

Note

Shared Unit link service requests are level-sensitive to the eTPU. If it comes from an interrupt source, the link service request stays asserted as long as the interrupt is asserted and enabled, so the interrupt request must be cleared before the clear of the LSR flag in the link service request thread.

LENG can also be used by the host to disable Link Service Requests to the eTPU. The eTPU microcode can only disable LSRs from the Shared Unit through configuration specific to the Shared Unit itself.

Note that more than one Shared Unit can generate LSRs to the same eTPU channel, in which case the LSR source can be determined by microcode through the Shared Unit status register(s). An eTPU channel configured as a Shared Unit LSR target can also receive Link Service Requests from other eTPU channels or Shared Units. The same LSR service and flag behavior found in links between eTPU channels applies to Shared Unit links (see Channel Link).

The table below summarizes the mapping of Reaction Module resources into Shared Subsystem Support.

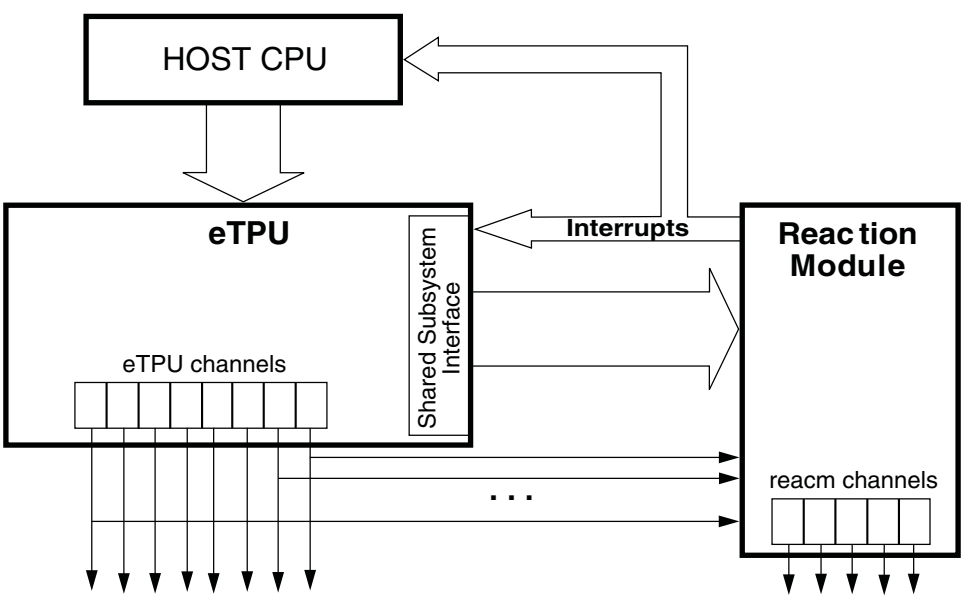


Figure 2-44. Shared Subsystem block diagram example: Reaction Module

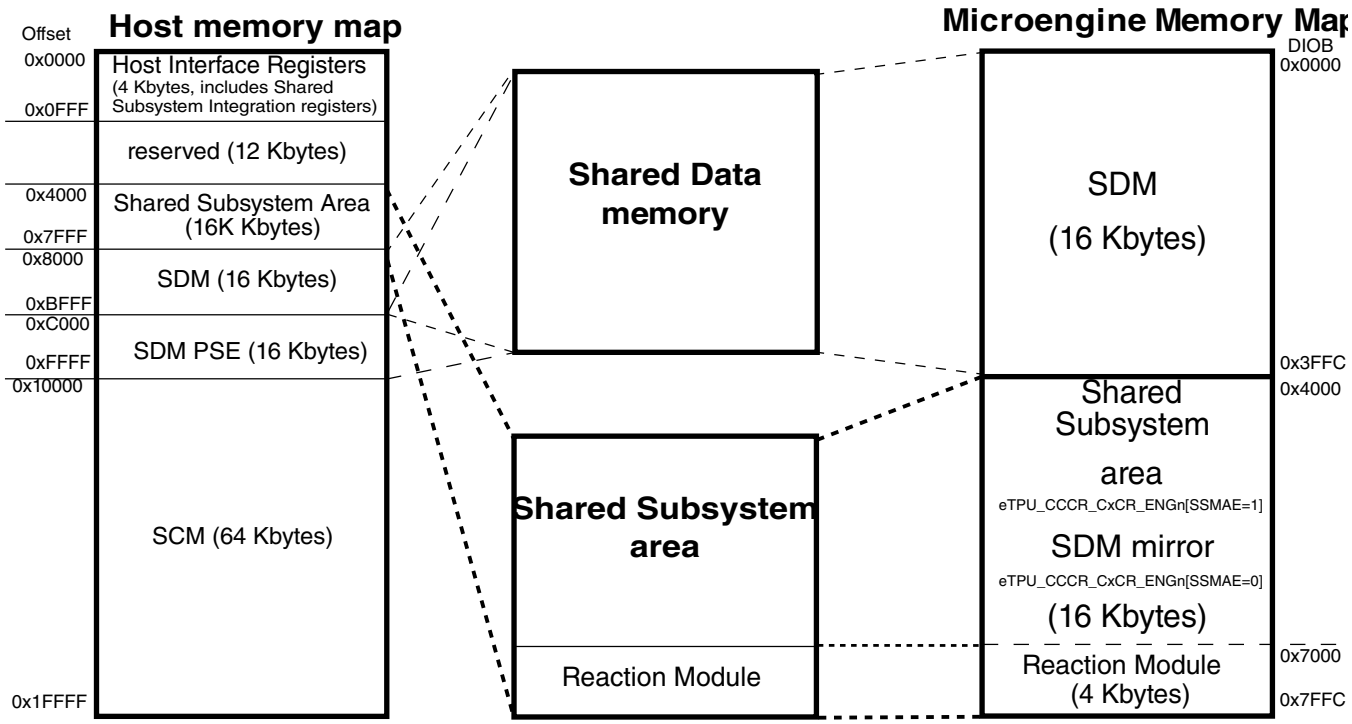


Figure 2-45. Shared Subsystem memory mapping

Table 2-105. Reaction Module Shared Subsystem support

Reaction Module resource	Host mapping	eTPU/microengine mapping
Register access	eTPU Base + 0×7000	DIOB address 0×7000 (indirect address mode)
Interrupts	See Interrupts chapter	Link Service Requests mapped through Subsystem Integration registers eTPU_SSIR_ETPUSSSIR*n*
DMA Requests	See eDMA chapter	Not applicable

2.14.11 Development support features

Following sections describe several features available to support development. Most debug features are accessible through a separate debug bus, and are not available through registers in the standard eTPU memory map. The details of the access to this interface are MCU-dependent (see Debug Interface chapter). The microengines can also be put in halt state by a device-level debug request.

NOTE

The behavior of the debug support logic is not specified in the event of a non-correctable memory error on either SCM or SDM (see Memory error support).

2.14.11.1 Nexus Class 3 Support

NDEDI is an IP block designed to support Nexus functionality for the eTPU. It provides a Nexus class 3 or class 1 debug interface, depending on the MCU (see Configuration section). Nexus is a development support external interface defined by the **IEEE standard ISTO 5001-1999**. More details about it can be found in the NDEDI details.

Some of the next subsections describe debug features provided by the eTPU microengine debug support combined with the NDEDI block.

2.14.11.2 Microengine Halt State

Halt is a microengine state where it suspends execution during a thread, or does not start executing a scheduled thread from idle state. While Idle State is entered from END execution without any other scheduled thread, microengine enters Halt State by any of the following events:

- execution of the HALT microinstruction (software breakpoint).
- external halt request through the Debug Interface (includes Nexus breakpoint request via EVTI input pin - see Nexus Class 3 Support).
- the other Engine enters halt state and they are configured to halt simultaneously (bit HTWIN is asserted via Nexus Interface).
- System level device debug request assertion and NDEDI register NDEDIETPUx_DC field CBI=1. If same register's field CBT=1, microengine halts at the next time-slot boundary, if CBT=0 it halts immediately. As a particular case, microengines come halted out of reset if device debug request is asserted, since CBI reset value is 1. Microengine does not execute out of reset, either in halt (device debug request asserted) or idle state (device debug request negated), but halt enables several other features (see below).

- occurrence of any of the hardware breakpoint conditions. See Hardware breakpoints for details.
- execution of a single-step microinstruction: microengine returns to halt state after executing a single microinstruction while in halt state. See Single-step Execution and Forced microinstruction execution for details.

When microengine enters halt state, it automatically triggers the following actions:

- suspends input signal sampling and filters (respective Engine channels only), if NDEDI_ENGINEx_DC[PINS] = 1 at the Debug Interface.
- releases the SDM arbitration for Host or CDC accesses, no matter if microengine was halted in the middle of a dual parameter (back-to-back) access.
- stops TCR1/2 clocks of the respective Engine, if signal ndedi_stop_tcr is asserted at the Debug Interface.
- if the other Engine is also in halt state or stopped, allows turning the eTPU Module Configuration Register (eTPU_SCR_MCR) VIS bit to 1.

If all halt conditions are cleared when VIS=1, microengine(s) keep on halt state until VIS=0, when it automatically exits halt state, except on single-step (see Single-step Execution), so that single-step execution is ignored while VIS=1.

MDU continues executing until it finishes any ongoing operation even if microengine is in halt state, except when the halted instruction is an END.

There are two kinds of halt state, depending on the previous microengine state when halted:

1. **halt_idle**, if the engine was not executing a thread when halted; the engine cannot leave halt_idle to fetch instructions, so one cannot single-step or follow a program flow; it can, however, execute forced instructions (see Forced microinstruction execution).
2. **halt_exec**, if the engine was executing a thread when halted. The engine can single-step and continue a program flow from halt_exec.

When microengine exits halt state, any dependable action is suspended and, if exiting halt_exec, the instruction pointed by the PC is fetched, while the instruction already fetched before halt is executed. Note that both the PC and the prefetched instructions can be modified during halt state, with a forced execution of a branch instruction (see Forced microinstruction execution).

2.14.11.3 Hardware breakpoints

Microengine can enter halt state through a command from the Debug Interface, configuring a hardware breakpoint. Hardware breakpoints can halt the microengine on specific conditions, listed below. These conditions depend on NDEDI configuration (see the NDEDI details for more information).

- CHAN register assignment (only by microcode, not by time slot transition).
- SDM read and/or write to a given address and/or write data. The breakpoint is always qualified by the SDM address, but the following variations are allowed:
  - break on write only, read only, or read-and-write.
  - break on higher-byte write data value, lower 24-bit write value, full word (32-bit) write value, or regardless of data. Break on read data is not supported.
- PC (program counter) value.
- Beginning of a thread with a Host Service Request pending.
- Beginning of a thread with a Link Service Request pending.
- Beginning of a thread with a Match Service Request pending.
- Beginning of a thread with a Transition Service Request pending.
- End of a thread.
- Illegal instruction execution.

All these conditions can also be qualified by the value of the CHAN register.

NOTE

Data breakpoints are not supported on Shared Subsystem Area accesses (see Shared Subsystems support).

Hardware breakpoints halt the microengine before the halted instruction is executed, except on data breakpoints. Data breakpoints allow the instruction that performs the data access to be executed before the microengine is halted.

Note

Data breakpoints in eTPU versions before eTPU2 halt the breakpoint instruction before its execution in all cases, including data breakpoints.

On any of these conditions, halt of one microengine does not depend on the halt of the other, unless the other engine is configured to do so, via Nexus Interface. Occurrence of any of these conditions halts the microengine, i.e., the conditions are logically ORed together, and they can be individually enabled.

While in halt state, the microengine can also execute any forced microinstruction not in the normal program flow (see Forced microinstruction execution) or, if in halt_exec, in single-step (see Single-step Execution).

There are situations when requests for stopping an engine, breakpoint and service can occur simultaneously. Breakpoint requests always prevails over a stop request (eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) or eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2) bit MDIS=1 or device debug request=1). When the eTPU is idle: stop request prevails over Service Request if there is not a hardware breakpoint request; a hardware breakpoint request leads to debug mode immediately if there is no Service Request, and after TST if there is Service Request (regardless of stop requests). The rules above are summarized in the the following table, showing the destination state of the microengine in each situation.

Table 2-106. Breakpoint, stop and service requests resolution from idle

Breakpoint request	MDIS	Service request	Final state
no	0	0	Idle
no	1	0	Stop
no	0	1	TST
no	1	1	Stop
Yes	0	0	Halt_idle
Yes	1	0	Halt_idle
Yes	0	1	TST[1]
Yes	1	1	TST[1]

1. Breaks after TST, if signal ndedi_sync_break is still asserted.

When a thread is ending, it goes to Idle or TST only if there is neither a hardware breakpoint request nor a request to stop (eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) or eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2) bit MDIS=1 or device debug request=1). When thread is ending and there are simultaneous hardware breakpoint and stop (MDIS=1 or device debug request=1) requests, hardware breakpoint prevails and the engine enters Debug mode (Halt_idle state).

If the engine entered Debug mode after a thread finished (Halt_idle state) and a "go" command comes from the debug interface, the engine state machine goes to Idle and the rules above apply. It means that if a "go" is issued in Halt_idle state with MDIS=1, the engine goes to Idle for one microcycle and then stops (if MDIS or device debug request keeps asserted and there is no other breakpoint request).

Note

Hardware breakpoint requests are ignored for the first microinstruction executed when microengine leaves halt.

2.14.11.4 Hardware Watchpoints

Debug Interface allows watchpoints on the same conditions available for hardware breakpoints (see Hardware breakpoints).

Note

Data breakpoints are not supported on Shared Subsystem Area accesses (see Shared Subsystems support).

2.14.11.5 Software breakpoints

A software breakpoint occurs when microengine executes a HALT microinstruction. Any number of software breakpoints can be set in code, usually replacing an active microinstruction.

Like any other microinstruction, HALT increments the PC and pre-fetches the next instruction. So, before the halt state is suspended, if the original program flow must be followed, the original instruction at the HALT address must be executed, regardless if the software breakpoint is removed (replacing HALT by the original microinstruction) or not. Follows the procedure to resume execution from a software breakpoint:

1. restore the original instruction in SCM (replace HALT).
2. force a jump with flush to the original instruction (see Forced microinstruction execution).
3. if the software breakpoint must be kept: single-step and replace the original instruction with a HALT.
4. let the flow continue, issuing a GO command (leaving halt state).

Special care must be taken if HALT is followed by another HALT, and the second HALT is removed when microengine was halted by the first one. In this case, replacing the second HALT with the original microinstruction is not enough to remove the second breakpoint, because the second HALT was already prefetched and would be executed anyway when halt was suspended. The debugger must also do a forced execution of unconditional branch with flush to the original microinstruction address. That will clear the pipeline, replacing the prefetched instruction with a NOP, and load PC with the address of the removed breakpoint. So, when halt state is suspended, the original microinstruction will be fetched while NOP is executed, and program flow continues normally from then on.

Note

A HALT instruction placed after a no-flushing branch, dispatch or return may be a problem from the debugger application standpoint: after the HALT is executed, the eTPU debug interface informs the address of the branch/dispatch/return destination, and the debugger application has no direct way to identify which HALT instruction was executed, if multiple HALTs lead to the same address. This can be solved if the debug support block (NDEDI) has a register holding the address of the last instruction executed, otherwise one should forbid non-flushed HALT instructions. See the NDEDI details.

Software breakpoint setting and removal is possible only with SCM RAM implementations or ROM implementations with SCM RAM emulation (see). There is only one way of inserting software breakpoints into SCM RAM: programming VIS=1 in eTPU Module Configuration Register (eTPU_SCR_MCR), and then accessing SCM as an ordinary RAM from the slave bus. This can be done only if both Engines are halted or stopped.

2.14.11.6 Single-step Execution

When microengine is already in halt_exec state, it can run the next microinstruction in the normal program flow and get back to halt state. PC is incremented, or assigned the BAF value in a branch with satisfied condition. Note that the executed instruction was already prefetched in the instruction pipeline, and a new microinstruction is fetched during its execution. The prefetched instruction may be cleared during halt state by the forced execution of a branch with flush (see Forced microinstruction execution), making single-step execute a NOP instead of the next instruction in the program flow.

Single-step execution is controlled by the debug interface, and is a feature available from Nexus if eTPU is connected to the NDEDI block (see NDEDI details).

The single-step execution of a NOP instruction can be useful to control input signal sampling and filtering, if NDEDI_ENGINEx_DC[PINS] = 1 at the Debug Interface (see NDEDI details). Single-step does not happen if VIS=1.

2.14.11.7 Forced microinstruction execution

When microengine is already in halt state (either halt_idle or halt_exec), it can run forced microinstructions through the debug interface. This feature is available from Nexus if eTPU is connected to the NDEDI block (see NDEDI details). The microinstruction, specified by the user, is not fetched from SCM and comes directly from the debug interface. MDU start commands issued by forced instructions are executed, and the MDU runs the operation until the end, independently of the halt state. The microinstruction field END is ignored.

During forced execution of any instruction except Branches, Returns and Dispatches, the PC does not change, and the prefetched instruction in the pipeline is bypassed, but not discarded. When halt state is suspended, the prefetched instruction is executed and the instruction pointed by the PC is prefetched in parallel (two-stage pipeline).

Forced execution of a Branch, Dispatch or Return loads the PC with the BAF field (if branch condition is satisfied), PC+P or RAR, respectively. If branch condition is not satisfied, PC value stays unaltered. The flush control (field FLS) also works, so that a successful forced branch with flush replaces the prefetched instruction with a NOP. So, to clear the instruction pipeline during halt, all one has to do is an unconditional branch to the desired address with flush. HALT instructions must not be executed as forced.

Forced operations that depend on the serviced channel are unspecified when executed in halt_idle and must be avoided.

Note

Forced instructions must not be commanded when eTPU Module Configuration Register (eTPU_SCR_MCR) field VIS=1.

2.14.11.8 Microengine Register Access

eTPU provides no direct access to microengine and channel registers from the slave bus or any other interface. However, these registers can be read and written in halt state by executing forced microinstructions (see Forced microinstruction execution). Immediate

data microinstructions may be used to set register values. Some registers are not selectable for immediate data destination, so intermediary register(s) - notably P - may have to be used to carry the desired new value to the target register in two or more microinstructions. Usually the previous values of intermediary register(s) must be previously saved and restored after the whole operation.

Similar procedures apply for register reads: their contents must be dumped to SDM, where they can be read from the slave bus.

2.14.11.9 Microengine Flag Access

Microengine halt state allows reading the branch conditions flags through forced microinstructions or, more easily, through the NDEDI register NDEDI_ENGINEx_CFSR (see the NDEDI details). Flag conditions set by the user are seen by microengine for the next microinstruction execution. The flag set options are limited by the possibilities of forced microinstruction execution.

If the eTPU runs (not single-stepping) after exiting the halted state, the conditions modified during halt may remain only for the first microcycle after the halted state. After the first microcycle, branch conditions are altered only according to their regular update scheme.

2.14.11.10 Microengine Stall

Microengine can get into a stall state, attending a request from a debug interface signal assertion. The reason for a Stall request from NDEDI (or from any other debug support block) should be a temporary lack of resources, for instance queue full. During stall the microengine suspends execution, but all the other Engine logic continues operating: time bases, angle logic, channel logic, input sampling and filters. Stall differs from Halt, not enabling any of the debug features that Halt enables (see Microengine Halt State). It also does not break an atomic microengine access, unlike halt.

The Microengine can be stalled when idle and from the moment TST ends, before executing the first thread microinstruction, until just before the last thread microinstruction is executed. Stall requests are ignored in any other occasions. Microengines in a dual engine system can be independently stalled. If a forced end is issued when microengine is in stall coming from execution, the END is executed only when the microengine resumes execution from stall.

2.14.12 Safety features

This section describes embedded safety features: MISC and memory error support.

The Multiple Input Signature Calculator - MISC - is an SCM test feature accessible through eTPU Module Configuration Register (eTPU_SCR_MCR) and eTPU MISC Compare Register (eTPU_SCR_MISCCMPR). MISC allows SCM test "on the fly", i.e., while eTPU is running, with no impact on eTPU functionality or performance.

Memory Error support features comprises SCM and SDM error detection, correction, report, and soft error.

2.14.12.1 SCM Test - Multiple Input Signature Calculator

The Multiple Input Signature Calculator (MISC) comprises special hardware that sequentially reads all SCM positions and calculates, in parallel, a 32-bit signature from a 32-input CRC signature calculator with the following polynomial:

$$1 + x^{1} + x^{2} + x^{22} + x^{31}$$

A complete description of the signature calculation procedure can be found in MISC Algorithm.

Once started by the Host the MISC runs continuously, restarting after the completion of each cycle, when it sets SCMMISC. The average time for a MISC calculation can be measured by checking SCMMISC state at regular intervals, incrementing a counter and clearing SCMMISC if it is set.

MISC accesses to the SCM array are executed if none of the engines is accessing the SCM, to avoid degradation of the microengine performance: it happens while no channel is being serviced. An ongoing MISC operation can be aborted by writing 0 to SCMMISEN.

The Host must load eTPU MISC Compare Register (eTPU_SCR_MISCCMPR) with the expected value to be found at the end of the MISC cycle, and then start the signature calculation by programming field SCMMISEN=1 in eTPU Module Configuration Register (eTPU_SCR_MCR). MISC zeroes the signature accumulator and starts reading SCM data and calculating the signature. After last SCM position is read, MISC compares the value in signature accumulator against the value in eTPU_SCR_MISCCMPR: if there is a mismatch MISC stops, a Global Exception is issued and SCMMISF assumes value 1. If no mismatch is found, MISC repeats the procedure automatically. When signature is being calculated, SCM address starts at the last SCM address and counts down to 0. The conditions for executing a MISC operation are (see also Table 2-26):

- Both microengines in idle state (no channel is being serviced) or stopped, in any combination (e.g., engine 1 idle with engine 2 stopped).
- VIS =0.
- SCMMISEN =1.

Note that MISC can run regardless of SCM implementation type (RAM or ROM).

If SCMMISEN=0 or VIS=1, the MISC logic stays at its initial state, with address counter pointing to the last SCM position and accumulator reset.

MISC can also be used to perform single-bit error fixes in the SCM. For more details, see Error fixing.

2.14.12.2 Memory error support

Error Correction Code (ECC) support for SDM and SCM provides guaranteed single bit error correction and double bit error detection. Error detection and correction is performed on read accesses and on write accesses less than 32-bit wide to the SCM.

SDM ECC is calculated per byte, 5 bits wide each, totalling 20 ECC bits for each 32-bit word. That allows the ECC to be updated without a read-modify-write operation on writes of any size.

Error is not checked on bytes not selected on SDM read accesses, either by the Host, CDC or microengine. For instance, the most significant byte is not checked for errors on a microengine 24-bit read access.

Error detection also works for host accesses to the PSE area (see Parameter Sign Extension area). In this case, error is not detected in the most significant byte (bits 31:24). Also, because of sign extension, error is always detected on the bits 23:16, regardless if they are selected, whenever the most significant byte is read.

SCM ECC is 7 bits wide, comprising the whole 32-bit word. Read-modify-write is automatically made when 16-bit or 8-bit wide writes are made into the SCM in a host access. SCM errors are also detected on Entry Table accesses during TST (see Time Slot Transition) and on MISC accesses, which are considered microengine accesses with regards to interrupt generation and error reporting.

2.14.12.2.1 Error detection disabling

The error detection logic in SDM and SCM can be disabled by setting the eTPU Memory Error Control Register (eTPU_MESR_MECR) bits DEDD and CEDD, respectively. The disabled state turns ineffective any action due to error detection, including error correction. Note that error detection is always enabled out of reset.

2.14.12.2.2 Memory error flags

SDM single (correctable) and non-correctable bit errors are flagged in eTPU Memory Error Status Register (eTPU_MESR_MESR) fields DCERR and DNCERR, respectively.

SCM correctable and non-correctable errors are flagged in the eTPU_MESR_MESR fields CCERR and CNCERR, respectively.

Double bit errors are guaranteed to be flagged, while three or more simultaneous bit errors in an SDM byte or SCM word are not. An error flagged as non-correctable can be in two or more bits.

Overrun flags DCOVR and CCOVR indicate, for SDM and SCM respectively, when a correctable error occurs when DCERR, CCERR are already asserted. Similarly, overrun flags DNCOVR/CNCOVR assert when a non-correctable error occurs when DNCERR/ CNCERR are asserted already.

2.14.12.2.3 Memory error reports

The address, syndrome and data of a non-correctable SDM access can be sampled into eTPU Data Error Report Address Register (eTPU_MESR_DERAR), eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR), and eTPU Data Error Report Data Register (eTPU_MESR_DERDR), respectively. eTPU_MESR_DERAR also captures the SDM read access originator, whether the host, CDC, or specific eTPU channel service.

Similarly, non-correctable SCM accesses address, syndrome and data are stored in eTPU Code Error Report Address Register (eTPU_MESR_CERAR), eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR), and eTPU Code Error Report Data Register (eTPU_MESR_CERDR), respectively. eTPU_MESR_CERAR also captures the SCM read access originator, whether the host, MISC, or specific eTPU channel service. The occurrence of a non-correctable error overwrites the report register values from a correctable error, either on SDM (DCERR=1) or SCM (CCERR=1).

The last address, syndrome and data of a correctable SDM access can also be sampled into eTPU_MESR_DERAR, eTPU_MESR_DERSR, and eTPU_MESR_DERDR, but only if DNCERR is 0. That prevents access address, syndrome and data from a correctable error to overwrite the same information from a non-correctable access.

Similarly, the last address, syndrome and data from a correctable SCM access are stored in eTPU_MESR_CERAR, eTPU_MESR_CERSR, and eTPU_MESR_CERDR only if CNCERR is 0.

For either SDM or SCM, a new correctable error overwrites the report registers' values from an earlier correctable error, but report values from a non-correctable error are not overwritten, as long as bit DNCERR/CNCERR is asserted, neither by a new correctable error nor by a non-correctable one.

The sampling of SDM and SCM error report registers can be disabled through the eTPU_MESR_MECR fields DRDIS and CRDIS, respectively. When DRDIS (CRDIS) is asserted, the update of the SDM (SCM) error report registers is disabled, but not of eTPU_MESR_MESR fields DCERR, DNCERR (CCERR, CNCERR). DRDIS, CRDIS must be set before reading the respective report registers to assure their coherent reading, preventing them to be updated between reads.

2.14.12.2.4 Error detection and wait-states

Microengine SDM accesses do not take any wait-state due to ECC support. Host SDM read accesses take two wait-states minimum[35] in MCUs with SDM ECC support, regardless if an error was detected or not, if the SDM error detection is enabled (eTPU Memory Error Control Register (eTPU_MESR_MECR) field DEDD=0). Host SDM read accesses take a minimum of one wait-state if the SDM error detection is disabled (eTPU_MESR_MECR[DEDD]=1). Host SDM write accesses take one wait-state minimum, regardless of ECC support presence or enabling.

Host SCM read accesses have two wait-states[35], regardless if an error was detected or not. Writes from the host to the SCM take one wait-state on 32-bit accesses, and one additional wait-state (two wait-states total[35] on 16-bit and 8-bit wide accesses. Those figures are not affected by the SCM error detection disabling (eTPU_MESR_MECR[CEDD]).

2.14.12.2.5 Memory error interrupts

For **microengine accesses**, the Global Exception interrupt asserts whenever a non-correctable error is detected, either on SCM or SDM (see Interrupts and data transfer requests). These errors are flagged by eTPU Module Configuration Register (eTPU_SCR_MCR) fields SCMERR and SDMERR, respectively. Single bit errors are corrected and activate the Memory Error Correction Interrupt if it is enabled (see Interrupt types and sources), but do not generate Global Exceptions.

35. One wait-state more, compared with eTPU versions before eTPU2 and previous eTPU2 without ECC support

For **host accesses** a bus error is issued whenever a non-correctable error is detected, either on SCM or SDM (but not through the CDC) and it also generates a Global Exception. Single bit errors are corrected (if the error correction feature is available) and do not cause bus transfer errors, but assert the Memory Error Correction Interrupt if it is enabled (see Interrupt types and sources).

2.14.12.2.6 Effect of errors on microcode execution

Correctable errors, either on SCM or SDM, do not affect microcode execution if detection/correction is enabled.

The executing thread is affected by a non-correctable SDM error as follows:

- If DTEND =1: on SDM errors due to a read operation, if the instruction is not also an END, the following instruction automatically executes an END, and its operations may be executed, totally or partially. On SDM errors during the TST preload, the thread automatically ends on the first instruction, if the error occurs on a preload parameter selected by the entry point bit PP (see Entry Point Format section.) The first instruction may be executed totally or partially. Data read into P or DIOB may be corrupted in either case.
- If DTEND =0: the TST or thread continues the execution flow normally. Data read into P or DIOB may be corrupted in either case.

The executing thread is affected by a non-correctable SCM error as follows:

- If CTEND =1: on SCM errors, the corrupted instruction is not executed, instead replaced by a simple END. Due to instruction prefetching, an error may be detected and yet not affect the thread execution. However, an error detected in a prefetched instruction may cause error report and global or memory error interrupts, even if not executed. If a non-correctable error occurs on the entry table read during TST (see Time Slot Transition section), the first instruction in the thread is replaced with a simple END, a global exception is issued and the reports and flags are normally updated.
- If CTEND =0: the execution flow continues normally, and the results depend on the particular instruction corruption.

Table 2-107. Error support summary if CEDD = DEDD = 0

Action		SCM		SDM	
		single bit error	double bit error	single bit error	double bit error
Memory Error Correction Interrupt		Asserts	Not asserted	Asserts	Not asserted
Global Exception		Not asserted	Asserts	Not asserted	Asserts
Bus Error (host reads)		Not issued	Issued	Not issued	Issued
Thread	CTEND = 0	Normal execution	Potential erroneous execution[1]	Normal execution	see DTEND

Table continues on the next page...

Table 2-107. Error support summary if CEDD = DEDD = 0 (continued)

Action		SCM		SDM	
		single bit error	double bit error	single bit error	double bit error
	CTEND = 1		Terminates before erroneous execution[2]		
	DTEND = 0		see CTEND		Potential erroneous execution
	DTEND = 1				Terminates after erroneous read[3]
Code Error Report Registers	CNCERR = 0	Updates eTPU_MESR_CERAR, eTPU_MESR_CERSR, eTPU_MESR_CERDR		Not related	
	CNCERR = 1	eTPU_MESR_CERAR, eTPU_MESR_CERSR, eTPU_MESR_CERDR not updated			
Data Error Report Registers	DNCERR = 0	Not related		Updates eTPU_MESR_DERAR, eTPU_MESR_DERSR, eTPU_MESR_DERDR	
	DNCERR = 1			eTPU_MESR_DERAR, eTPU_MESR_DERSR, eTPU_MESR_DERDR not updated	

1. The thread executes correctly if the error is detected on an instruction fetched but not executed.
2. The erroneous instruction is not executed at all, and replaced by a simple END.
3. The instruction following the erroneous read may be executed totally or partially.

2.14.12.2.7 Error Correction Code (ECC) and syndrome definition

SDM ECC is composed of 5 Hsiao code[36] bits for each data byte. **SCM ECC** is composed of 7 Hsiao code bits for each 32-bit word.

Table 2-108 shows the encoding of the SDM syndrome and global parity fields as they read on eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR). There is one syndrome and one global parity for each byte in an SDM word, numbered SYND0 to SYND3 from the least to the most significant.

Table 2-108. SDM syndrome definition

SYNDn (hex)	Result	SYNDn (hex)	Result
0E	error on data bit 8n+7	10	error on **code** bit 8n+4
15	error on data bit 8n+6	08	error on **code** bit 8n+3
07	error on data bit 8n+5	04	error on **code** bit 8n+2
1A	error on data bit 8n+4	02	error on **code** bit 8n+1

Table continues on the next page...

36. Hsiao, M.Y;.A Class of Optimal Minimum Odd-weight-column SEC-DED Codes; IBM Journal of Research and Development http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5391627

Table 2-108. SDM syndrome definition (continued)

SYNDn (hex)	Result	SYNDn (hex)	Result
19	error on data bit 8n+3	01	error on **code** bit 8n
13	error on data bit 8n+2		
16	error on data bit 8n+1	00	no error
1C	error on data bit 8n	any other	non-correctable error

The following table shows the definition of the SCM syndrome field as it reads on eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR).

Table 2-109. SCM syndrome definition

SYND (hex)	Result	SYND (hex)	Result	SYND (hex)	Result
40	error on **code** bit 6	58	error on data bit 25	46	error on data bit 11
20	error on **code** bit 5	0B	error on data bit 24	4A	error on data bit 10
10	error on **code** bit 4	54	error on data bit 23	52	error on data bit 9
08	error on **code** bit 3	15	error on data bit 22	62	error on data bit 8
04	error on **code** bit 2	16	error on data bit 21	13	error on data bit 7
02	error on **code** bit 1	34	error on data bit 20	29	error on data bit 6
01	error on **code** bit 0	25	error on data bit 19	31	error on data bit 5
		26	error on data bit 18	43	error on data bit 4
49	error on data bit 31	64	error on data bit 17	45	error on data bit 3
0D	error on data bit 30	2C	error on data bit 16	19	error on data bit 2
0E	error on data bit 29	1A	error on data bit 15	51	error on data bit 1
38	error on data bit 28	23	error on data bit 14	61	error on data bit 0
4C	error on data bit 27	2A	error on data bit 13	00	no error
1C	error on data bit 26	32	error on data bit 12	any other	non-correctable error

2.14.12.2.8 Error injection

Errors can be injected into the SDM read data path through eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR), eTPU Data Error Injection Data Pattern Register (eTPU_MESR_DEIDPR) and eTPU Data Error Injection Parity Pattern Register (eTPU_MESR_DEIPPR). They define bit flips on accesses at the specific address specified in eTPU_MESR_DEIAR.

SDM error injection is enabled independently for host and microengine accesses by the eTPU Module Configuration Register (eTPU_SCR_MCR) fields HDEIE and MDEIE, respectively. The user must program eTPU_MESR_DEIAR, eTPU_MESR_DEIDPR,

and eTPU_MESR_DEIPPR with HDEIE=MDEIE=0, and only then enable error injection by setting HDEIE=1 and/or MDEIE=1 to avoid races (see eTPU Memory Error Status Register (eTPU_MESR_MESR)).

Similarly, SCM errors can be injected into the SCM read data path through eTPU Code Error Injection Address Register (eTPU_MESR_CEIAR), eTPU Code Error Injection Data Pattern Register (eTPU_MESR_CEIDPR) and eTPU Code Error Injection Parity Pattern Register (eTPU_MESR_CEIPPR), enabled by the eTPU Module Configuration Register (eTPU_SCR_MCR) field CEIE. Errors are injected on host accesses to the address specified in eTPU_MESR_CEIAR when VIS =1, or on microengine and MISC accesses when VIS=0 .

2.14.12.2.9 Error fixing

Single bit **soft errors** can be fixed by rewriting the right value, as corrected by the ECC logic, into the same address of the erroneous data. eTPU provides means to perform such fixes safely, coherently and concurrently with the normal operation.

SDM fixes can be done using the CDC (see Coherent Dual-parameter Controller (CDC)), limited to the lower 8 Kbytes of SDM. The Host application simply has to program a transfer with both origin and destination positions pointing to the address of the data to be fixed. Because the PBBASE field has an address granularity of 8 bytes, a word adjacent to the one to be fixed must also be rewritten. The CDC transfer is atomic, therefore the fix is done coherently and does not interfere with the application code or data flows.

SCM fixes can be done by programming eTPU Code Error Fix Register (eTPU_MESR_CEFR) (see eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR)). The fixes are performed by the MISC logic, which must be enabled (see SCM Test - Multiple Input Signature Calculator). The modes available, selected by CFIXM, are:

- **Automatic Fix Mode (CFIXM=11)**: the MISC logic automatically rewrites the contents of an address whenever it finds a correctable error; the CFIXM field keeps its value if a fix is made.

2.14.13 Performance monitoring features

2.14.13.1 Idle Counter

The Idle Counters eTPU Idle Register for Engine 1 (eTPU_ER_IDLER_ENG1) and eTPU Idle Register for Engine 2 (eTPU_ER_IDLER_ENG2) continuously count microcycles in which the microengine is not busy with channel service . They can be used to measure the microengine utilization by rating the count measured during a period of time to the number of microcycles contained in the period. The Idle Counters do not count microcycles when the engine is stopped, or is in TST or halt states.

2.14.13.2 Latency Monitor

eTPU implements six Latency Monitors to measure the time taken to service a request. Each Latency Monitor provides:

- The measure of the greatest latency found during actual thread executions, accessed from eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM*n*MR_ENG1) and eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM*n*MR_ENG2).
- A comparison between the measured latency and a threshold, determined through eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM*n*MR_ENG2) and eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM*n*CR_ENG2).

Each Latency Monitor works independently in one of the following modes, selected with the LMM field in eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM*n*CR_ENG1) and eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM*n*CR_ENG2) :

- **Disabled mode**: no latency measures or actions are taken.
- **Channel mode**: monitors the latency of an individual channel.
- **Priority mode**: monitors the combined latency of all channels in a given priority level - High, Medium or Low (see Scheduler).

Channel and Priority modes are detailed in the subsections below. The following applies to both modes:

- After each latency measurement, the value measured is compared with the LATENCY value in eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM*n*MR_ENG1) and eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM*n*MR_ENG2). If the new measured value is greater, it replaces the old value in eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM*n*MR_ENG1) and eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM*n*MR_ENG2).
- After each latency measurement, the value measured is compared with the MAXLAT value in eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM*n*CR_ENG1) and eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM*n*CR_ENG2), and if the new measured value is greater:
  - A Global Exception is issued.
  - The eTPU Module Configuration Register (eTPU_SCR_MCR) flag LTO1 (engine 1) or LTO2 (engine 2) is set, independently of the LATENCY value in eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM*n*MR_ENG1) and eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM*n*MR_ENG2).

NOTE

The LATENCY value is updated even if MAXLAT=0.

NOTE

MAXLAT=0 just inhibits the assertion of the LTO1/2 flag and Global Exception due to the respective Latency Monitor. Other Latency Monitor instances with non-zero MAXLAT value may assert LTO1/2 and Global Exception.

2.14.13.2.1 Channel mode

When the Latency Monitor operates in Channel mode:

- Only the channel selected by the MONCHAN field in eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM*n*CR_ENG1) and eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM*n*CR_ENG2) is monitored.
- Any combination of Host Service Requests (HSR), Channel Service Requests and Link Service Requests can be selected for monitoring, based on the SRSEL field in eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM*n*CR_ENG1) and eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM*n*CR_ENG2).

- The latency is counted in microcycles, from the moment the logic OR of all the service requests selected by SRSEL asserts to the moment this logic OR negates.
- The latency count is interrupted when the channel is disabled, either by programming the CPR field in eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2) to 0, or by a watchdog timeout. Latency is counted until the channel is disabled and is not discarded; a new count can start from zero when the channel is re-enabled.

NOTE

If a selected service request type or a combination of them continues asserted during and after its service thread execution, its latency counter will keep on running; this can happen, for instance, if either matches can service requests (e.g. channel modes em_st/em_dt), one match happens before the other is serviced, and only one of them is cleared. The Latency Monitor can, in cases like this, yield pessimistic measurements.

NOTE

The Latency Monitor in channel mode must be disabled before the channel it monitors is disabled.

2.14.13.2.2 Priority mode

When the Latency Monitor operates in Priority mode:

- Only the combination of channels assigned with the priority selected by the SRSEL field in eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM*n*CR_ENG1) and eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM*n*CR_ENG2) is monitored.
- The latency is measured in microcycles, starting from either of these two events, whichever happens first:
  - The logic OR of service requests of the selected priority transitions from 0 to 1 at a moment when all their grants are at state negated.
  - The grants of the selected priority are cleared by the scheduler at the end of the thread, and any of the service requests at the same priority is still active soon after the thread ends.
- The latency is measured until the grants of the selected priority are cleared by the scheduler, after the measurement started.

Note

A Latency Monitor in priority mode must be disabled before any of the channels in the selected priority is disabled, or any disabled channel is assigned with the selected priority.

2.14.13.3 Watchdog Monitor Operation

The watchdog mechanism can also be used to monitor, without interfering with the application execution, whether the execution time of any individual thread or a sequence of threads reaches a given limit. This can be achieved by asserting the WMO field in eTPU Watchdog Timer Register for Engine 1 (eTPU_ER_WDTR_ENG1) and eTPU Watchdog Timer Register for Engine 2 (eTPU_ER_WDTR_ENG2).

2.15 Initialization/application information

2.15.1 Configuration sequence

After initial power-on reset the eTPU remains in an idle state[37], requiring initialization of several registers before any function can begin execution. Also, if the SCM is implemented in RAM, it should be initialized with the eTPU application code prior to configuring the eTPU. Configuration procedures are summarized as follows:

- If SCM is implemented as RAM, load the eTPU application code (see SCM access).
- Initialize SDM (optional if no ECC support available)
- Initialize the SCM MISC logic (optional, see SCM Test - Multiple Input Signature Calculator).
- Initialize Watchdog and Latency Monitors (optional).
- Initialize eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1) and eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2) to set up:
  - TCR1 and TCR2 prescalers and clock sources.
  - Select digital filtering mode.

37. Except when device debug request is asserted on power-on reset: in this case, the microengines wake-up in halt state.

  - TCRCLK signal filter control.
  - Angle mode operation (if necessary).
- Initialize eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2) to set up:
  - Entry table base.
  - Filter prescaler clock control.
- Initialize eTPU Engine 1 STAC Configuration Register (eTPU_TBR_STAC_ENG1) and eTPU Engine 2 STAC Configuration Register (eTPU_TBR_STAC_ENG2) if you need to set up TCR1/2 resource Client/Server operation.
- Write to eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2) to choose the Function to be performed by each channel, and its parameter base address.
- Write to eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C*n*SCR_ENG1) and eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C*n*SCR_ENG2) to choose among the possible variations within the function flow (FM bits).
- Write to SDM for parameter initialization of each configured channel.
- Write to eTPU Watchdog Timer Register for Engine 1 (eTPU_ER_WDTR_ENG1) and eTPU Watchdog Timer Register for Engine 2 (eTPU_ER_WDTR_ENG2) if you need to enable and set up the Watchdog(s) mode and timeout.
- Write to eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C*n*HSRR_ENG1) and eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C*n*HSRR_ENG2) to initialize the active channels.
- Write to eTPU Channel Interrupt Enable Register for Engine 1 (eTPU_GCR_CIER_ENG1) and eTPU Channel Interrupt Enable Register for Engine 2 (eTPU_GCR_CIER_ENG2) if interrupts are to be enabled from the appropriate channels. Likewise for Data Transfer Requests (eTPU Channel Data Transfer Request Enable Register for Engine 1 (eTPU_GCR_CDTRER_ENG1) and eTPU Channel Data Transfer Request Enable Register for Engine 2 (eTPU_GCR_CDTRER_ENG2)). This can also be done through eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2).

- Write to eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2) to enable each channel by assigning it a high, middle, or low priority (CPR field).[37]
- Monitor eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C*n*HSRR_ENG1) and eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C*n*HSRR_ENG2) for completion of initialization.
- Write GTBE =1 to start TCR1/TCR2 time base counting at same time in both Engines (may be done before or never, depending on the particular application and use of Red Line bus).

2.15.2 Reset options

2.15.2.1 Hardware reset

Hardware reset is achieved by assertion of device synchronous reset. Both Engines and common logic is reset, and even the System Configuration and Global Channel registers assume their reset values.

Note

All eTPU input clocks must pulse during reset so that both engines are reset, even if they are in Module Disable or Stop mode.

2.15.2.2 Software reset

eTPU has no software reset. To abort infinite microcode loops, the Force END mechanism must be used (see field FEND in eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2)).

2.15.3 Multiple parameter coherency methods

Follows a description of two methods for coherent transfer of multiple parameters between Host and eTPU. Both methods involve the use of two parameter areas: the Transfer Parameter Area (hereafter called TPA), which is the SDM area directly accessed by the Host for reads and writes, and the Permanent Parameter Area (hereafter called PPA), which are the SDM positions where channel parameters are normally accessed by the Function microcode. Note that parameters in either TPA or PPA do not have to be in sequential addresses. TPAs and PPAs allocation are completely defined by the application, and there may be any number of them, independently of the channels.

The methods described here are not the only solutions for the coherent transfer problem, and both can coexist in eTPU and even used in combination. Also note that for transfers of a pair of parameters, the Coherent Dual-parameter Controller is faster and have less impact on both eTPU and Host performance. That said, the methods are:

- **Transfer Service**: a microengine thread transfers, upon Host Service Request, data from/to a TPA to/from a PPA.Coherency is guaranteed by the fact that a thread is atomic with respect to other threads in the same Engine, and so are its transfers. If parameters in PPA are shared by both Engines, hardware semaphores have to be used to access them.
- **Mailbox**: for Host to eTPU transfers, the microcode checks a flag, set by the host, indicating the existence of new parameter data in the TPA. It can, then, either access TPA data directly or copy it to the PPA. For eTPU to Host transfers, when microcode changes PPA, it copies them to the TPA and flags updated TPA data to Host, possibly using an Interrupt or a Data Transfer Request. The Mailbox flag is reset when data is copied: by the eTPU microcode, when it transfers TPA to PPA (possibly followed by an Interrupt); by the Host, when it reads data from the TPA. This indicates that TPA is free for another transfer.

Transfer Service has the advantage of separating the task of data transfer from the functional service thread that accesses the parameters, with less impact to the latter. Compared to the Mailbox method, however, it has bigger average latency, because the Transfer Service thread has to contend for a time slot to execute. This latency can be minimized if Transfer Service thread is assigned to a separate channel with higher priority, but even so it does not guarantee that PPA is updated before the next execution of the functional thread that uses it.

Mailbox method, on the other hand, makes the functional thread check for the existence of new data (Host to eTPU). It does not have to be responsible for the transfer, though: it may access the TPA directly, and a Transfer Service can then be used to copy data from TPA to PPA.

2.15.4 Programming Hints and Caveats

2.15.4.1 Atomic dual access after a call, return

A dual, back-to-back parameter access is not atomic after a call, a jump, or a return if they occurred in parallel with an odd SDM access. It is safer to make a pair of parameter accesses that must be coherent begin at the second instruction after a call/jump/return.

2.15.4.2 Resource polling

The use of polling while waiting for a condition or a resource (except semaphore lock) should be avoided in order not to hang the microengine in long loops. This general programming guideline is greatly enforced in eTPU, as a thread cannot be preempted for any reason. Safer polling, albeit with long and indeterministic latency, can be obtained if a thread issues a channel link to the serviced channel and terminates. The microengine is then free to other tasks, and the next poll happens at the next time the channel is serviced. This mechanism can be combined with finite (timed out) loops for better latency.

2.15.4.3 Changing channel function, parameter base, or entry table scheme

Channel function, parameter base address and entry table scheme are determined by the CFS, CPBA, and ETCS fields in eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2). They cannot be changed when the channel is enabled. If the channel is disabled first, one may still have service requests from the previous function, so before the channel is enabled again one must be sure that:

- the first thread executed in the new function is the initialization one.
- the initialization thread of the new function clears any previously pending service request.

Here is a safe procedure for function changing:

1. Disable the channel (in eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2), write CPR=00).
2. Change the function configuration (fields CFS and/or CPBA and/or ETCS in eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2)).
3. Request the initialization thread, writing eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C*n*HSRR_ENG1) and eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C*n*HSRR_ENG2) with the initialization HSR (channel still disabled).
4. enable the channel (in eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2) write CPR > 0); the initialization HSR is serviced before any other formerly pending service requests, clearing them.

2.15.4.4 Checking and Clearing Interrupts of a Stopped Engine

An Engine may be stopped with interrupts (or DMA requests) pending. This includes the case when the Engine's MDIS bit is set and a thread is still running: the thread will complete execution, possibly issuing an interrupt or DMA request before the engine stops, setting the STF bit.

As soon as the engine stops the channel registers become inaccessible, issuing bus errors when accessed. Interrupts and DMA requests can still be checked and cleared through the Global Channel Registers, though. DMA requests can also be cleared by the hardware handshaking with the DMA controller when the engine is stopped.

2.15.5 Estimating Worst Case Latency

Reliable systems are designed to work under worst-case conditions. This section explains how to estimate worst-case latency (WCL) for any eTPU function in any system. The appendix covers the following topics:

- Introduction to Worst-Case Latency
- Using Worst-Case Latency Estimates to Evaluate Performance
- Priority Scheme Details used in WCL Analyses

- First-Pass WCL Analysis
- Second-Pass WCL Analysis

The first-pass WCL analysis is based on a deterministic, generalized formula that is easy to apply. Because of the generalizations in the formula, the first analysis result is almost always much worse than the real worst case. If the desired system performance is within the limits of this first analysis, then no further analysis is required; the system is well within the performance limits of the eTPU. If the desired system performance exceeds that indicated by the first analysis, the second-pass WCL analysis should be applied. The second-pass analysis is not a generalized formula, but rather uses specific system details for a realistic worst-case estimation.

2.15.5.1 Using Worst-Case Latency estimates to evaluate performance

Once WCL is found for a channel, the user must determine how to use this number to analyze performance. To analyze the performance of a channel running the PWM function, for example, some information about what happens in each thread is necessary.

The following example refers to old TPU PWM function, which is not optimized to the eTPU enhanced hardware. For PWM, thread 1 is the initialization thread, and threads 2 and 3 are used during normal function execution. (PWM threads 4, 5, and 6 are for special modes and will be assumed to be unused on channel 5). Thread 2 writes a time into the channel 5 match register and performs other operations that will cause the channel 5 signal to go from low to high at the time indicated in the match register (match time). At match time, the signal goes high and channel 5 requests service from the eTPU Microengine to execute thread 3. Thread 3 writes a time into the channel 5 match register and performs other operations that will cause the channel 5 signal to go from high to low at match time. At match time, the signal goes low and channel 5 requests service from the eTPU Microengine to execute thread 2. A PWM wave is kept running on the system by the eTPU executing thread 2, then thread 3, then thread 2, then thread 3, and so on.

Since the definition of worse-case latency assumes a fully loaded running system, initialization threads are not part of worst-case calculations. For the channel 5 example, the two PWM threads in are thus the two normal running threads, threads 2 and 3.

does not define which thread is thread 2 and which is thread 3. Since the worst-case latency derived from the first-pass analysis is the worst case between any2 threads (not counting initialization threads), it is safe to say that the worst-case latency shown in represents both the worst-case high time and the worst-case low time.

Notice in that worst-case latency is drawn from the end of the execution of the first PWM thread to the end of the execution of the next PWM thread. It is drawn from end to end because the microcode instructions that make up the threads control the channel hardware. To make sure that all the microcode instructions needed to change the pin thread have been executed, it is necessary to include the execution time of the second thread.

Thread information for each function is found in the programming notes for individual TPU functions. See NXP Programming Note TPUPN00/D, Using the TPU Function Library and TPU Emulation Mode, for a list of available programming notes. Similar documentation will we provided for the eTPU new functions.

2.15.5.2 Priority Scheme Details Used in WCL Analysis

The user assigns functions to channel numbers and gives each active channel a priority level of high, middle, or low. The Scheduler uses the channel number and channel priority level to determine the order in which to grant service.

The scheduler allocates time slots to specific priority levels of high, middle, or low. One function thread is executed in each time slot. The length of a time slot varies according to the length of the executing thread. When fully loaded, the scheduler always assigns time slots in a seven-slot sequence (see the following table). After a seven-slot sequence is completed, another seven-slot sequence begins (see Figure 2-47). Note that in eTPU, when no service request exists, the scheduler goes to thread 1, but WCL calculation considers full load.

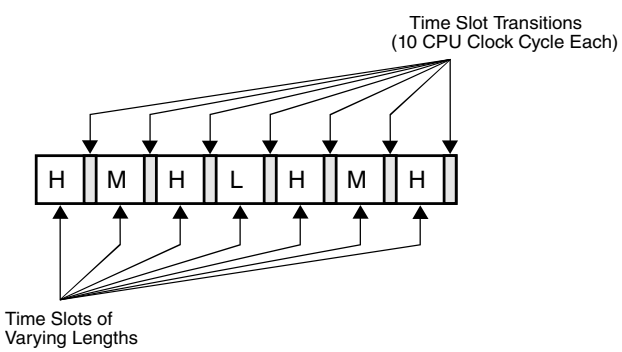


Figure 2-46. Time-Slot Sequence

This sequence scheme gives higher-priority channels more service time than lower-priority channels. High-priority channels are allocated four of seven time slots, middle-priority channels are allocated two of seven time slots, and low-priority channels are allocated one of seven time slots.

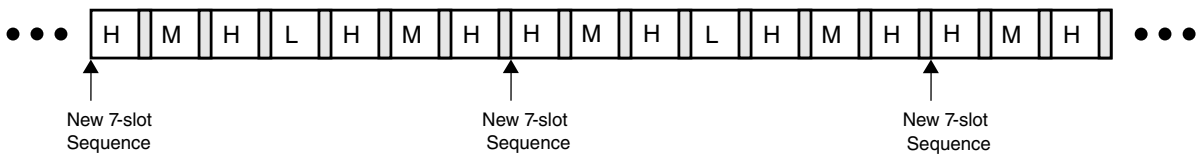


Figure 2-47. Multiple Time-Slot Sequences

2.15.5.2.1 Priority Passing

If no channel of the priority level assigned to the time slot is requesting service, the eTPU scheduler can pass priority to other levels. If no high-level channel is requesting service during a high level time slot, a middle-level channel is granted service; or, if no middle level-channel is requesting service, a low-level channel is granted service. If no middle-level channel is requesting service during a middle-level time slot, a high-level channel is granted service; or, if no high-level channel is requesting service, a low-level channel is granted service. If no low-level channel is requesting service during a low-level time slot, a high-level channel is granted service; or, if no high-level channel is requesting service, a middle-level channel is granted service. If no channel is requesting service, the time slot sequence is reset to state 1 and the scheduler idles until a request is received.

Priority passing is implemented in hardware and does not contribute to worst-case latency.

2.15.5.2.2 Time-slot transition

After each time slot, the eTPU must prepare for the next time slot. This preparation time between each time slot is called a time-slot transition. See Time Slot Transition. Time-slot transitions can take from six up to ten eTPU clocks.

2.15.5.2.3 Channel Number Priority

If more than one channel of a priority level is requesting service, the lowest numbered channel is granted service first. For example, if channels 0, 5, and 9 are all high-level channels requesting service during a high time slot, channel 0 is granted service first. Continuing this example, if channel 0 requests service again immediately after being serviced, it is not serviced again until channels 5 and 9 are serviced. This scheme is implemented so that continuously-requesting low numbered channels do not take all the time on the eTPU execution unit and leave no time for other channels.

The scheduler uses registers to keep track of which channels have been serviced and which require servicing. Each channel has two register bit: a service request register (SRR) and a service grant register (SGR). The SRR is set when a channel requests service. After the channel has been granted service, the SGR is set and the SRR is cleared.

SGRs are not cleared individually by channel, but rather as priority level groups. The clearing of a group of SGRs begins a new cycle for that priority level. An SGR group is cleared on the condition that a channel of that priority level has just been serviced, and no other channel of that priority level is requesting service (has a set SRR) and has not been granted service (has a clear SGR).

For example, if a middle-priority channel has just been serviced (either in a middle-priority time slot or a high or low-priority time slot gained by priority passing), the SRRs and SGRs of all middle-priority channels are compared. If there is no middle-priority channel with its SRR set and SGR cleared, the scheduler clears all middle-level SGRs. If there is a middle-level channel with its SRR set and SGR cleared, the scheduler does not clear the SGR group, and the requesting middle-level channel is serviced on the next middle-level time slot (or possibly sooner by priority passing).

2.15.5.2.4 SDM Collision Rate

Most function threads read or write to the eTPU SDM at least once. Because both the eTPU Microengine and Host can access the SDM but not at the same time, the Microengine may suspend execution during the SDM access while waiting for the Host to finish accessing the SDM. At other times the Host may wait for the Microengine. Wait states can take up to two eTPU clocks, when the Host accesses the SDM directly, without using CDC. Microengine(s) wait-states must be added into the worst-case latency calculation. The system designer should estimate the percentage of SDM accesses in the system that will result in Microengine wait-states. This percentage is called the RAM collision rate (RCR). In each collision with direct Host accesses to the SDM the Microengine(s) wait for two eTPU clocks.

In eTPU the Coherent Dual-parameter Controller (CDC) may also access the SDM for atomic transfers of two parameters. eTPU Microengine may wait on this operation (if it is in service time) until the transfer is complete. CDC always transfers two parameters, making four consecutive accesses (read, write, read, write) of one eTPU clock each. The system designer should estimate the percentage of SDM accesses in the system that will result in a Microengine wait due to coherent transfer, and multiply it with the average number of eTPU clocks the Microengine waits for each transfer. This percentage is called Coherent Parameter Collision Rate (CPCR).

In addition, Microengine to Microengine multiple parameter coherent communication, using the hardware semaphores, may hold one Microengine which waits to lock the semaphore while the other Microengine is holding it. This waiting is due to a software loop, not hardware wait-states. Note that single parameter access of one Microengine does not affect the timing of the other Microengine due to SDM time interlace. This implies that single parameter Microengine to Microengine communication does not affect the performance. The Microengine which waits for the semaphore will loop until it is

freed by the other Microengine. This time depends on the eTPU application. The system designer should estimate the percentage of Microengine to Microengine coherent parameter communication that will result in eTPU semaphore loops, and multiply it with the average number of eTPU clocks the Microengine loops for each such transfer. This percentage is called CCR (Communication Collision Rate).

A 100% collision rate for a system is the theoretical worst case. In many systems, however, the RCR, CPCR and CCR would be very low, sometimes even near 0%. This is because the eTPU is an independent processor capable of servicing most function needs, so that the Host rarely needs to access the eTPU SDM. Also coherent Microengine to Microengine communication of more than one parameter may be rare. To find a realistic RCR, CPCR the system designer should evaluate the Host code and find the percentage of time it accesses the eTPU SDM with or without using the CDC. This percentage gives a good RCR and CPCR. The eTPU application provides a good estimation of CCR.

Note

The programming practice of polling a flag in the eTPU SDM causes a very high RCR and should be avoided in high-performance systems.

After the collision rate for a system is found, it can be applied to the WCL calculations for each channel. The system designer can use the collision percentage and the number of SDM accesses (with and without semaphores) to estimate the eTPU loop time for a function. Note that in old TPU functions CPCR and CCR are both zero.

The estimation of eTPU wait time is as follows:

Variables:

N1 = Number of simple RAM accesses in the longest thread

RCRWait = Maximal eTPU clocks wait time for simple RAM collision = 2

CPCRWait = Average eTPU clocks for Coherent Parameter Transfer (using CDC).

N2 = Number of eTPU-eTPU semaphore RAM accesses in the longest thread

CCRWait = Average eTPU clocks for Microengine-Microengine communication transfer.

Estimated Wait Time:

Function eTPU maximal wait time =

N1 *(RCR * RCRWait + CPCR * CPCRWait) + N2* CCR * CCRWait

2.15.5.3 First-Pass Worst-Case Latency Analysis

Following is the first-pass calculation of worst-case latency for a channel. Remember that this analysis uses generalizations that usually produce a result much worse than the real worst case. If the worst-case result from the first analysis is too long for the desired performance, use the second analysis for a more realistic worst-case analysis.

2.15.5.3.1 Worst-case assumptions and formula

To estimate worst-case latency for a channel, assume this worst-case condition: the channel has just been serviced in a time slot of its priority level, and all other channels in the system are continuously requesting service and have cleared SGRs. The worst-case latency is the time from the end of the channel's service until the end of the channel's next service. See the following figure.

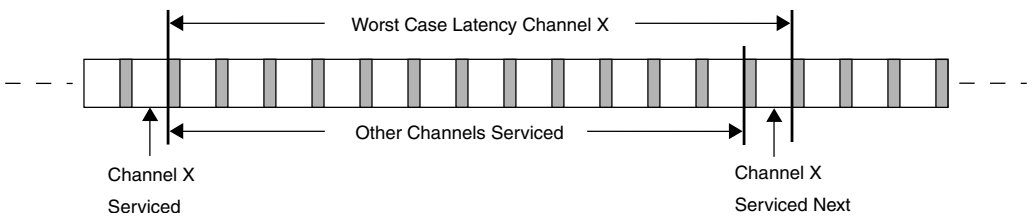


Figure 2-48. First-pass Worst-Case Latency

To estimate worst-case latency:

- Find the worst-case service time for each active channel.
- Using the H-M-H-L-H-M-H time-slot sequence, map the channels that are granted for each time slot.
- Add time for six-clock time-slot transitions.

2.15.5.3.1.1 Mapping the Channels for Each Time Slot

To determine when a channel will be serviced again, it is necessary to determine which other channels will be serviced first. Do this by assuming all channels are continuously requesting service and mapping the channels into the time-slot sequence.

2.15.5.3.1.2 Adding Time for Time-Slot Transitions

Add six eTPU clocks for time-slot transitions which occur after each time slot.

2.15.5.3.2 First-pass analysis Worst-Case Latency examples

The examples in this section assume the system configuration shown in the following table.

Table 2-110. System configuration example

Channel	Priority	Function[1, 2]
0	High	PWM (driving a DC motor)
1	Middle	PPWA (Mode 0, measuring the DC motor speed)
2	Low	DIO (Input)

1. 9% RAM Collision Rate (RCR)
2. CPU clock rate = 40 MHz, or 25 ns per clock period

2.15.5.3.2.1 Finding the WCL for PWM on Channel 0

The following shows how to find the WCL for PWM on channel 0.

1. Find the worst-case service time for each active channel.
   a. Longest thread of PWM is 24 CPU clocks with four RAM accesses.

      24 + ((4 RAM accesses+1) * 0.09 * 2 CPU clock waits) = 24.9 CPU clocks, rounded up to 25 CPU clocks (since there are no partial clock periods)

      Channel 0 worst-case service time = 25 CPU clocks.

   b. Longest thread of PPWA in mode 0 is 44 CPU clocks with nine RAM accesses.

      44 + ((9 RAM accesses+1) * 0.09 * 2 CPU clock waits) = 45.8 CPU clocks, rounded up to 46 CPU clocks

      Channel 1 worst-case service time = 46 CPU clocks.

   c. Longest thread of DIO is ten CPU clocks with four RAM accesses.

      10 + ((4 RAM accesses+1) * 0.09 * 2 CPU clock waits) = 10.9 CPU clocks, rounded up to 11 CPU clocks

      Channel 2 worst-case service time = 11 CPU clocks.

2. Assume channel 0 has just been serviced and that channels 1 and 2 are continuously requesting service. Using the H-M-H-L-H-M-H time-slot sequence, map the channels that are granted for each time slot. See the following figure.

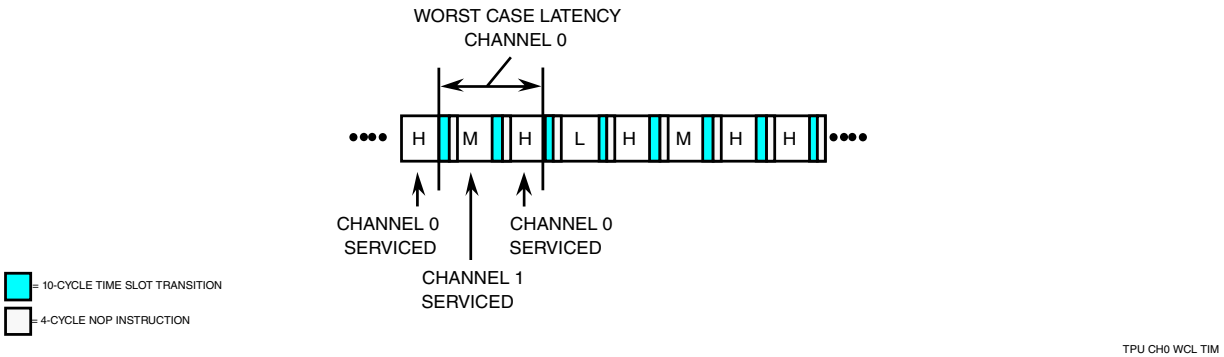


Figure 2-49. Next Servicing for Channel 0

Channel 1 will be serviced in the middle-priority time slot before channel 0 is serviced again.

3. Add time for the six-clock CPU time-slot transitions. See the above figure and the following table.

   A four-clock NOP occurs after each channel is serviced since there is one channel in each priority level, i.e., a new cycle for a priority level is started after each channel is serviced. Time-slot transitions occur after each time slot.

Table 2-111. Worst-Case Latency for Channel 0

Channel 0 worst-case service time	25 clocks
Channel 1 worst-case service time	46 clocks
Two 6-clock time-slot transitions	12 clocks
Total clocks	83 clocks

83 clocks * 25 ns/clock = 2075 ns

Conclusion: in this system configuration PWM can run with a minimum high time or low time of 2075 ns.

Note that in double match eTPU system the PWM can be serviced once in each period, and there is no latency for minimum high time. The latency in eTPU PWM function will represent the minimum PWM period.

2.15.5.3.2.2 Finding the WCL for PPWA on Channel 1

The following shows how to find the WCL for PPWA on channel 1.

1. Find the worst-case service time for each active channel. See step 1 of previous example.
2. Assume channel 1 has just been serviced and that channels 0 and 2 are continuously requesting service. Using the H-M-H-L-H-M-H time-slot sequence, map the channels that are granted for each time slot. See the following figure.

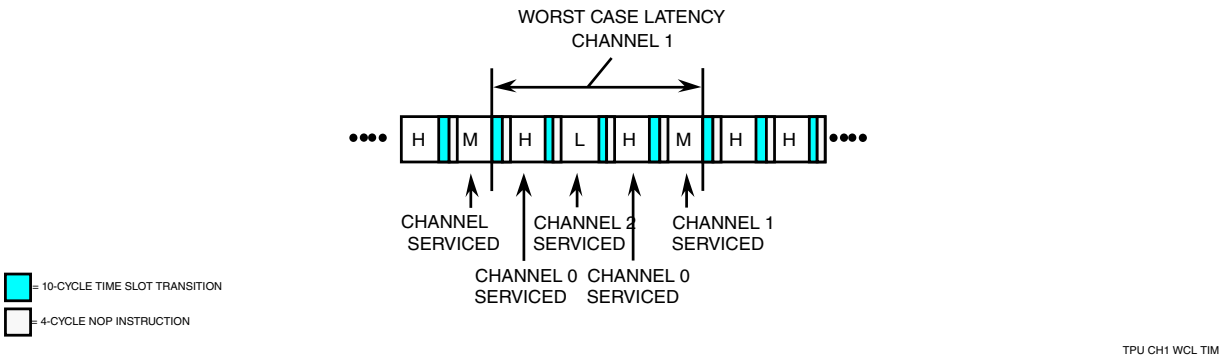


Figure 2-50. Next Servicing for Channel 1

Channel 0 will be serviced twice and channel 2 once before channel 1 is serviced again.

3. Add time for the six-clock CPU time-slot transitions. See the above figure and the following table.

Table 2-112. Worst Case Latency for Channel 1

Two Channel 0 worst-case service times	50 clocks
Channel 1 worst-case service time	46 clocks
Channel 2 worst-case service time	11 clocks
Four 6-clock time-slot transitions	24 clocks
Total clocks	131 clocks

131 clocks * 25 ns/clock = 3275 ns

Conclusion: in this system configuration PPWA can measure a period or pulse of minimum 3275 ns.

Note that PPWA function optimized for eTPU hardware can use double transition mode to measure very narrow pulses with one service after the second transition, and latency will affect only the minimum gap between two input pulses. Also the function threads would have more efficient coding.

2.15.5.3.2.3 Finding the WCL for DIO on Channel 2

The following shows how to find the WCL for DIO on channel 2.

1. Find the worst-case service time for each active channel. See step 1 of previous examples.
2. Assume channel 2 has just been serviced and that channels 0 and 1 are continuously requesting service. Using the H-M-H-L-H-M-H time-slot sequence, map the channels that are granted for each time slot. See the following figure.

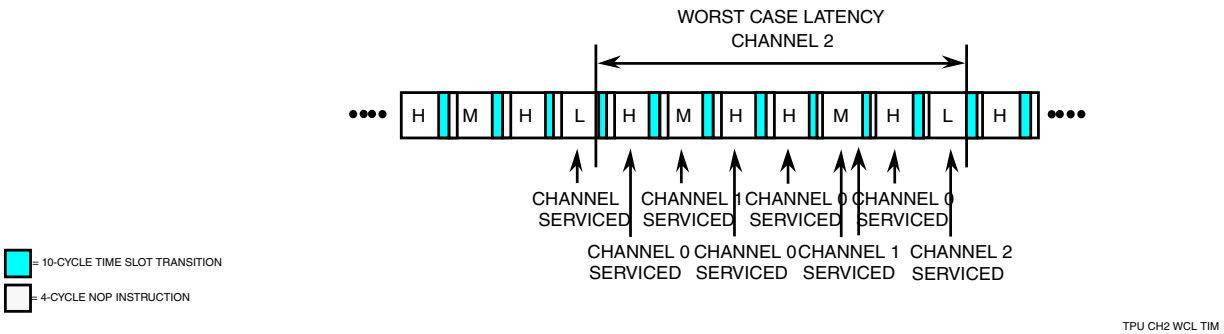


Figure 2-51. Next Servicing for Channel 2

Channel 0 will be serviced four times and channel 1 twice before channel 2 is serviced again.

3. Add time for the ten-clock CPU time-slot transitions and the four-clock NOPs. See the above figure and the following figure.

Table 2-113. Worst Case Latency for Channel 2

Four Channel 0 worst-case service times	100 clocks
Two Channel 1 worst-case service time	92 clocks
Channel 2 worst-case service time	11 clocks
Seven 6-clock time-slot transitions	42 clocks
Total clocks	245 clocks

245 clocks * 25 ns/clock = 6125 ns

Conclusion: in this system configuration DIO can keep track of the input level at a minimum of every 6125 ns.

Note that DIO function optimized for eTPU hardware can use double transition mode to measure two pin transitions at a time and reduce the service time, improving the overall system performance and latency.

2.15.5.4 Second-Pass Worst-Case Latency Analysis

Following is an example of a second-pass analysis for calculating worst-case latency for a channel. The second-pass analysis is useful for higher-performance systems, since it gives a more realistic worst-case latency result than first-pass analysis.

This example uses a relatively simple system in order to illustrate the basic principles of second-pass analysis. For a more complex example of second-pass analysis, refer to Multiphase Motor Commutation TPU Function (COMM)(TPUPN09/ D).

2.15.5.4.1 Second-Pass Analysis Guidelines

Rather than use a fixed formula, a second-pass analysis relies on the application of the following guidelines.

1. The first-pass analysis makes the assumption that all channels in the system are continually requesting service. For many systems this is an unrealistic assumption. For example, if TCR1 is counting at a rate of 2 MHz (500 ns per count) and a channel is running the DIO function with a match rate of 20,000 TCR1 counts, the DIO will request service every 10 ms (20,000 * 500 ns = 10,000,000 ns or 10 ms). It is therefore unrealistic to assume that the channel running this DIO function is continuously requesting service. Figure out a realistic service request rate for each channel. Time slots can then be mapped to each channel at the real rate of request.
2. If a function is active during system initialization but not during the high-speed running mode of the system, then that system does not need to be included in the high-speed worst-case latency calculations.
3. Use a realistic SDM collision rate.
4. Be careful when assigning functions priority levels and channel numbers. Decide which function or functions will be most difficult to make perform at the desired level. Assign those channels high priority and low channel numbers. Try different priority and channel assignments to see how it affects the system.
5. The seven-slot sequence of || H | M | H | L | H | M | H || is asymmetrical when put back-to-back with other seven-slot sequences. Note that in the following sequence there are two high-priority slots next to each other:

|| H | M | H | L | H | M | H |||| H | M | H | L | H | M | H ||

6. Make sure that when mapping out channels to the sequence, you choose a worst-case slot to start the mapping. For example, when estimating WCL for a high-priority channel, do not start the mapping in the last high-priority slot in a seven-slot sequence, as that is a best case for a high-priority channel since another high-priority time slot is next.
7. Instead of always using the longest thread in the function as the worst-case thread, evaluate the threads in the function that will be used in the system and use the appropriate worst-case threads. For example, in the preceding example of first-pass analysis, the PWM was shown to be able to achieve a high time and low time of 2475 ns under worst-case conditions. This was derived using the longest PWM thread of 24 CPU clocks. This longest thread is actually thread 2, the thread that is entered after the pin has just gone high. Thread 3, the thread that is entered after the pin has just gone low, requires only 2 CPU clocks. Therefore, in the first-pass example, the high time was correctly derived, but the low time is actually shorter than was estimated.

2.15.5.4.2 Second-pass analysis example

This example requires three 50% PWM waveforms: one 5 kHz (200 ms/period) and two 50 kHz (20 ms/period), each running DC motors. (Remember that the PWM function requests service from the eTPU after each high time and after each low time, so the eTPU must handle a request every 100 ms for the 5 kHz PWM and every 10 ms for the 50 MHz PWM.)

Note

This example uses square waves for simplicity. Notice that to use a PWM waveform in the typical way, in which the pulse is modulated, the pulse must not be modulated in a way that violates the worst-case latency requirements.

This example also uses one DIO channel monitoring a signal level every millisecond and one PPWA channel in mode 0 monitoring the speed of the 5-kHz DC motor. The PPWA must measure periods of 5 kHz (200 ms/period).

The CPU is interrupted by the channel running the PPWA function after measuring 200 periods (every 40 ms). The interrupt service routine performs an averaging of the period accumulation and checks it against a known parameter. The interrupt service time is so short and infrequent that it is a tiny fraction of total system time. The interrupt service routine contains no polling of the SDM. Therefore a realistic RCR = 0%.

2.15.5.4.2.1 First-Try System Configuration

Try a system configuration that seems likely to work. If it does not, change priority levels or channel numbers.

The 5 kHz and 50 kHz PWMs are the most time-critical functions. Those are assigned high priority. PPWA is assigned middle priority. The DIO is low performance and is assigned low priority. Refer to the following table.

Table 2-114. First-Try System Configuration

Channel	Priority	Function[1], [2]
0	High	PWM at 50 kHz (needs a 4-µs WCL)
1	High	PWM at 50 kHz (needs a 4-µs WCL)
2	High	PWM at 5 kHz (needs a 40-µs WCL)
8	Middle	PPWA at 5 kHz (needs a 80-µs WCL)
15	Low	DIO as input at rate of 1 ms

1. 0% RAM collision rate
2. CPU clock rate = 40 MHz, or 60 ns per clock period

With this system configuration, worst-case service time for each active channel is determined as follows:

1. Longest thread of PWM is 24 CPU clocks with four RAM accesses.

   24 + ((4 RAM accesses+1) * 0 * 2 CPU clock waits) = 24 CPU clocks

   Channels 0-2 worst-case service time = 24 CPU clocks.

2. Longest thread of PPWA in mode 0 is 44 CPU clocks with nine RAM accesses.

   44 + ((9 RAM accesses +1)* 0 * 2 CPU clock waits) = 44 CPU clocks

   Channel 8 worst-case service time = 44 CPU clocks.

3. Longest thread of DIO is ten CPU clocks with four RAM accesses.

   10 + ((4 RAM accesses+1) * 0 * 2 CPU clock waits) = 10 CPU clocks

   Channel 15 worst-case service time = 10 CPU clocks.

   To find the WCL for channel 0, assume channel 0 has just finished service.

   Map the channels in the H-M-H-L-H-M-H sequence. See the following figure.

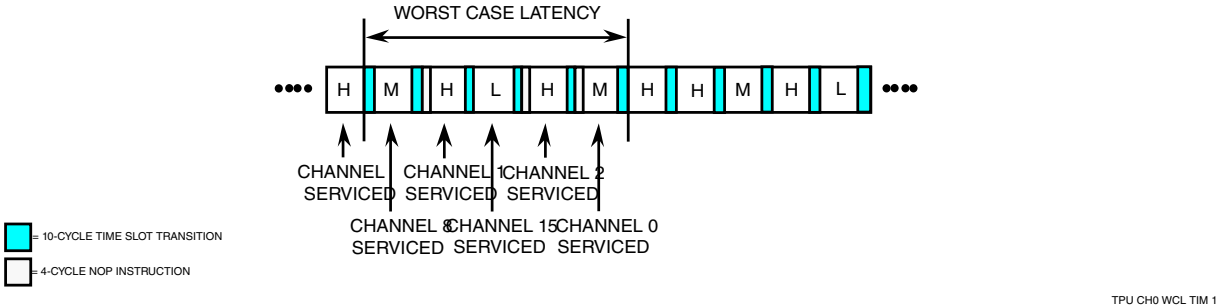


Figure 2-52. Worst-Case Latency for Channel 0 (First Try)

Conclusion: with this system configuration, worst-case latencies for channels 0 and 1 are too high (WCL for channel 1 is the same as WCL for channel 0). Try a different system configuration.

2.15.5.4.2.2 Second-Try System Configuration

The second-try system configuration is shown in the following table.

Table 2-115. Second-Try System Configuration

Channel	Priority	Function[1], [2]
0	High	PWM at 50 kHz (needs a 4-µs WCL)
1	High	PWM at 50 kHz (needs a 4-µs WCL)
2	Middle	PWM at 5 kHz (needs a 40-µs WCL)
8	Middle	PPWA at 5 kHz (needs a 80-µs WCL)
15	Low	DIO as input at rate of 1 ms

1. 0% RAM collision rate
2. CPU clock rate = 40 MHz, or 60 ns per clock period

To find the WCL for channel 0, assume channel 0 has just finished service. Map the channels in the H-M-H-L-H-M-H sequence. See the following figure.

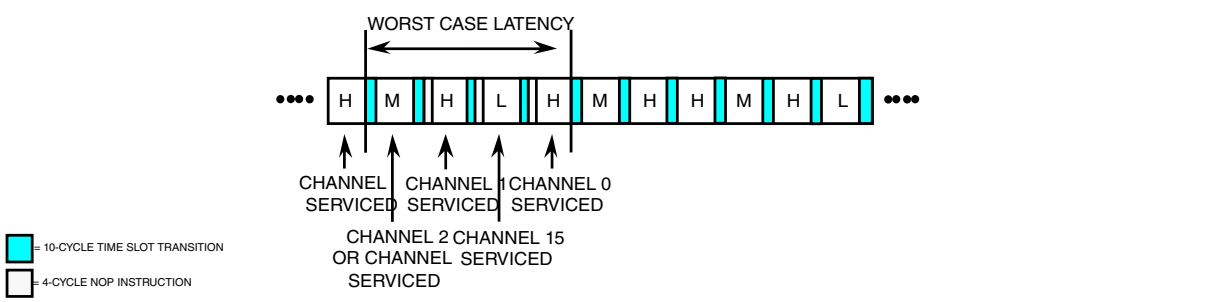


Figure 2-53. Worst-Case Latency for Channel 0 (Second Try)

Conclusion: with this system configuration, the WCL of both channel 0 and channel 1 is 3.85 ms, which is within the limit of 4 ms needed for a 50-kHz PWM.

Next, find the WCL for channel 2. Assume channel 2 has just finished service. Map the channels in the H-M-H-L-H-M-H sequence. See the following figure.

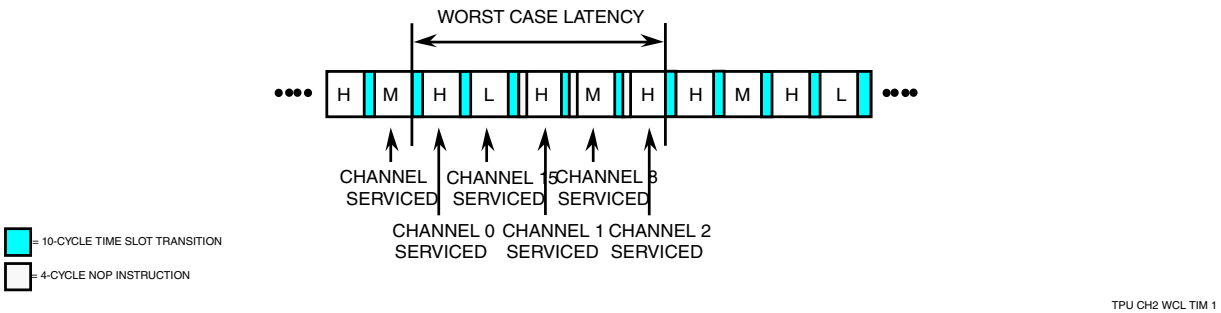


Figure 2-54. Worst-Case Latency for Channel 2

Conclusion: with this system configuration, the WCL for channels 2 and 8 is 4.7 ms, which is within the 40 and 80 ms WCL requirements.

Notice that channels 2 and 8 are well within their WCL requirements. The system could be reconfigured as shown in the following table to give channels 0 and 1 a larger margin while still keeping channels 2, 8 and 15 within their WCL requirements.

Table 2-116. Second-Try System with Channel 0 and 1 Reconfigured

Channel	Priority	Function[1, 2]
0	High	PWM at 50 kHz (needs a 10-µs WCL)
1	High	PWM at 50 kHz (needs a 10-µs WCL)
2	Middle	PWM at 5 kHz (needs a 40-µs WCL)
8	Low	PPWA at 5 kHz (needs a 80-µs WCL)
15	Low	DIO as input at rate of 1 ms

1. 0% RAM collision rate
2. CPU clock rate = 40 MHz, or 60 ns per clock period

2.15.6 Endianness

The following table shows the parameter addresses for big and little endian machines.

Table 2-117. Parameter Addresses and Endianness

Parameter	byte address offset (n = word address offset)	
	big endian	little endian
32-bit	4*n	
24-bit	4*n + 1	4*n
32-bit parameter's most significant byte	4*n	4*n + 3
24-bit parameter's most significant byte	4*n + 1	4*n + 2
least significant byte	4*n + 3	4*n

2.15.7 Memory Error Service Procedures

Follows separate procedures for correctable and non-correctable error interrupt service routines. It is assumed that the non-correctable error ISR has higher priority, and cannot be preempted by the correctable error ISR. For memory error support details, see Memory error support.

Note

The memory error support hardware does not (and is not intended to) guarantee that no error occurred will go unreported. However, combined with the procedures below, the information loss is minimized to statistical irrelevance on most scenarios.

2.15.7.1 Non-correctable error service procedure

The basic procedure below can be followed in a Global Exception or a bus error machine check service routine. It handles the service for both SCM and SDM errors, as they are symmetrical at this level. However, a real implementation is likely to split the procedure in two, one for each memory.

1. Confirm/identify the error source, checking assertion of the eTPU Module Configuration Register (eTPU_SCR_MCR) fields SCMERR and SDMERR.
2. Save the state(s) of eTPU Module Configuration Register (eTPU_SCR_MCR) fields CRDIS/DRDIS.
3. Set the eTPU Module Configuration Register (eTPU_SCR_MCR) CRDIS/DRDIS=1 to disable further updates to the respective report registers, guaranteeing coherency.

4. Check whether the report registers were updated by a correctable or non-correctable error; to do this, check the eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR) syndrome (for SCM) and/or the eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR) syndromes (for SDM, only those with the respective BE flags asserted). If it is a correctable error report, skip the steps 5 and 6 for the respective memory.
5. Identify the error source (host, microengine or MISC/CDC) by checking the field ERR_ACC in the register(s) eTPU Code Error Report Address Register (eTPU_MESR_CERAR) and eTPU Data Error Report Address Register (eTPU_MESR_DERAR); take or schedule the appropriate application-dependent recover actions.
6. Update error statistics as needed; it may include recording the access source, address, data and syndrome from:
   - eTPU Code Error Report Address Register (eTPU_MESR_CERAR)
   - eTPU Data Error Report Address Register (eTPU_MESR_DERAR)
   - eTPU Code Error Report Data Register (eTPU_MESR_CERDR)
   - eTPU Data Error Report Data Register (eTPU_MESR_DERDR)
   - eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR)
   - eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR)
7. Save the status of non-correctable error overflows flagged in the eTPU Memory Error Status Register (eTPU_MESR_MESR) fields CNCOVR/DNCOVR.
8. Set the eTPU Memory Error Control Register (eTPU_MESR_MECR) fields CRDIS/DRDIS to the state(s) saved in the step 2. That assures the access coherency of a preempted Correctable Error Service Procedure (which may have set CRDIS/DRDIS before this procedure) is not disturbed.
9. Clear the eTPU Memory Error Status Register (eTPU_MESR_MESR) fields CNCOVR/DNCOVR and CNCERR/DNCERR at once, in a single write to eTPU_MESR_MESR.
10. Use the overrun status saved on step 7 to update the error statistics, as needed.
11. Check the other Global Exception flags in eTPU Module Configuration Register (eTPU_SCR_MCR); take or schedule the appropriate (application dependent) recover actions.
12. Clear the Global Exception, programming GEC =1.

2.15.7.2 Correctable error service procedure

The basic procedure below can be followed in a Memory Error Interrupt service routine (see Interrupts and data transfer requests). It handles the service for both SCM and SDM errors, as they are symmetrical at this level. However, a real implementation is likely to split the procedure in two, one for each memory.

1. Confirm/identify the error source, checking assertion of the eTPU Memory Error Status Register (eTPU_MESR_MESR) fields CCERR/DCERR.
2. Program the eTPU Memory Error Control Register (eTPU_MESR_MECR) fields CRDIS=1 and/or DRDIS=1, to disable further updates to the respective report registers, guaranteeing coherency.
3. Check whether the report registers were updated by a correctable or non-correctable error; to do this, check the eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR) syndrome (for SCM) and/or the eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR) syndromes (for SDM, only those with the respective BE flags asserted). If it is a non-correctable error report, skip the steps 4 and 5 for the respective memory.
4. Identify the error source (host, microengine or MISC/CDC) by checking the field ERR_ACC in eTPU Code Error Report Address Register (eTPU_MESR_CERAR) and eTPU Data Error Report Address Register (eTPU_MESR_DERAR); take or schedule the appropriate application-dependent recover actions, which may include soft-error fixing (see Error fixing).
5. Update error statistics as needed; it may include recording the access source, address, data and syndrome from:
   - eTPU Code Error Report Address Register (eTPU_MESR_CERAR)
   - eTPU Data Error Report Address Register (eTPU_MESR_DERAR)
   - eTPU Code Error Report Data Register (eTPU_MESR_CERDR)
   - eTPU Data Error Report Data Register (eTPU_MESR_DERDR)
   - eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR)
   - eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR)
6. Save the status of correctable error overflows flagged in the eTPU Memory Error Status Register (eTPU_MESR_MESR) fields CCOVR/DCOVR.
7. Program the eTPU Memory Error Control Register (eTPU_MESR_MECR) fields CRDIS/DRDIS=0.
8. Clear the eTPU Memory Error Status Register (eTPU_MESR_MESR) fields CCOVR/DCOVR and CCERR/DCERR at once, in a single write to eTPU_MESR_MESR.

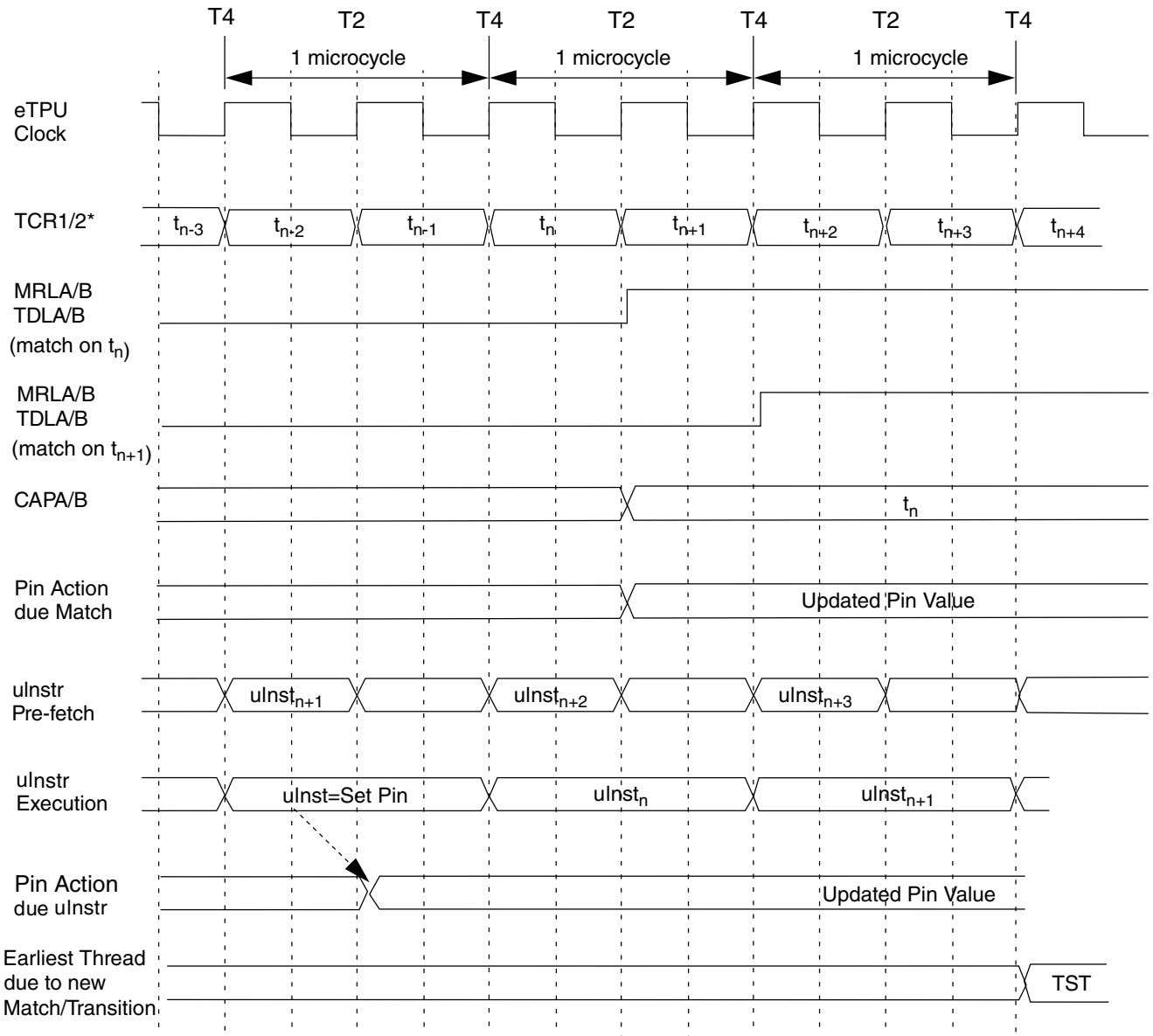
9. Use the saved overrun status to update the error statistics, as needed.

2.16 Appendices

2.16.1 Microcycle and I/O timing

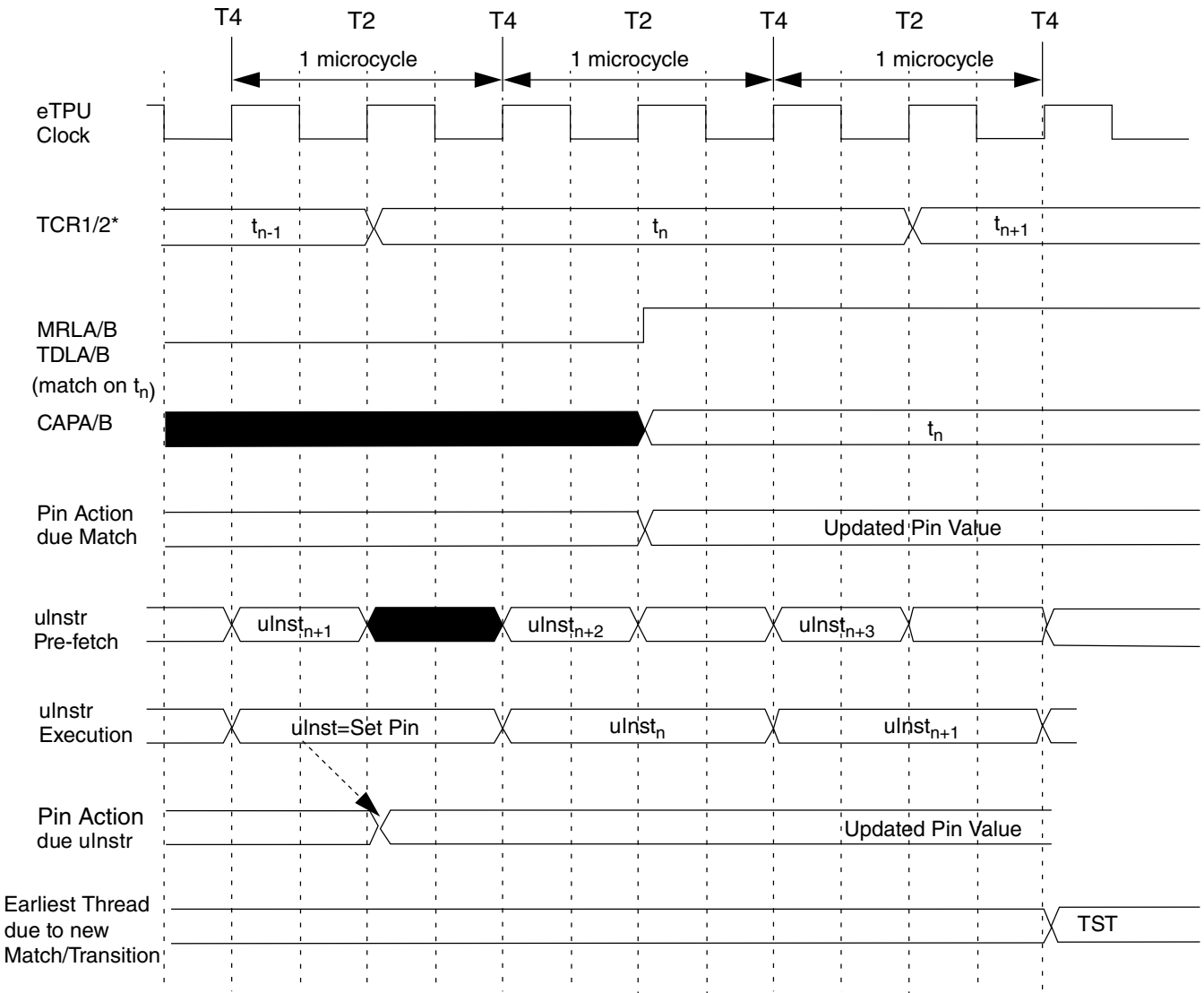
2.16.1.1 Execution and channel timing

The following figure shows the main timings related to microinstruction execution when channels and timebases run on T2 timing.



Note: *TCR clock/prescaler selection = 1 x eTPU clock

Figure 2-55. Execution, timebase and channel T2 timing



Note: *TCR clock/prescaler selection = 4x eTPU clock

Figure 2-56. Execution, timebase and channel T2/T4 timing

The sequential occurrence of the four T states (T1-T4) constitutes a microcycle. Only T2 and T4 are taken as reference for timings, either internal or external. T2 and T4 have the timing of the positive eTPU clock pulses, and are used in most of the eTPU logic in an edge-triggered design style.

Two additional T states are derived from the eTPU clocks: $\overline{T2}$ and $\overline{T4}$. $\overline{T2}$ occurs when the eTPU loses SDM arbitration to a bus master. $\overline{T4}$ occurs in halt state (due to a breakpoint or device debug request assertion, for instance), or in stall state (due to an NDEDI queue full); see Development support features for more details.

$\overline{T2}$ and $\overline{T4}$ states are defined as microcycle timing states (not to be confounded with logic states) of one eTPU clock in which the T clocks continue to run, but the control signals associated with the clocks are unaffected. That is, no operation occurs during these states.

Both $\overline{T2}$ and $\overline{T4}$ states occur in multiples of two eTPU clocks to keep the microengine synchronized with the free running channels, which are unaffected and keep on working as in T2 and T4.

Thus, the eTPU has two types of timing states:

- $\overline{T2}$: hold execution for SDM access, from clock pulse T2 until one of the next T2 clock pulses of another microcycle.
- $\overline{T4}$: hold execution in debug mode or stall, from clock pulse T4 until one of the next T4 clock pulses of another microcycle.

The two following figures show the timing of $\overline{T2}$ and $\overline{T4}$ timing states, respectively.

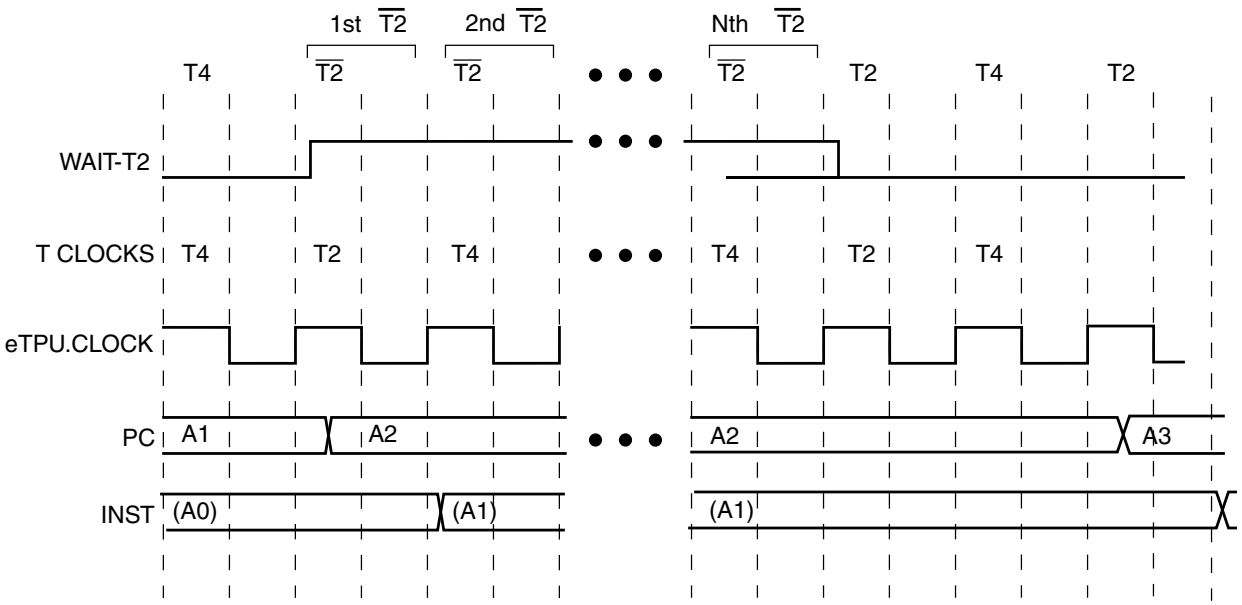


Figure 2-57. $\overline{T2}$ timing

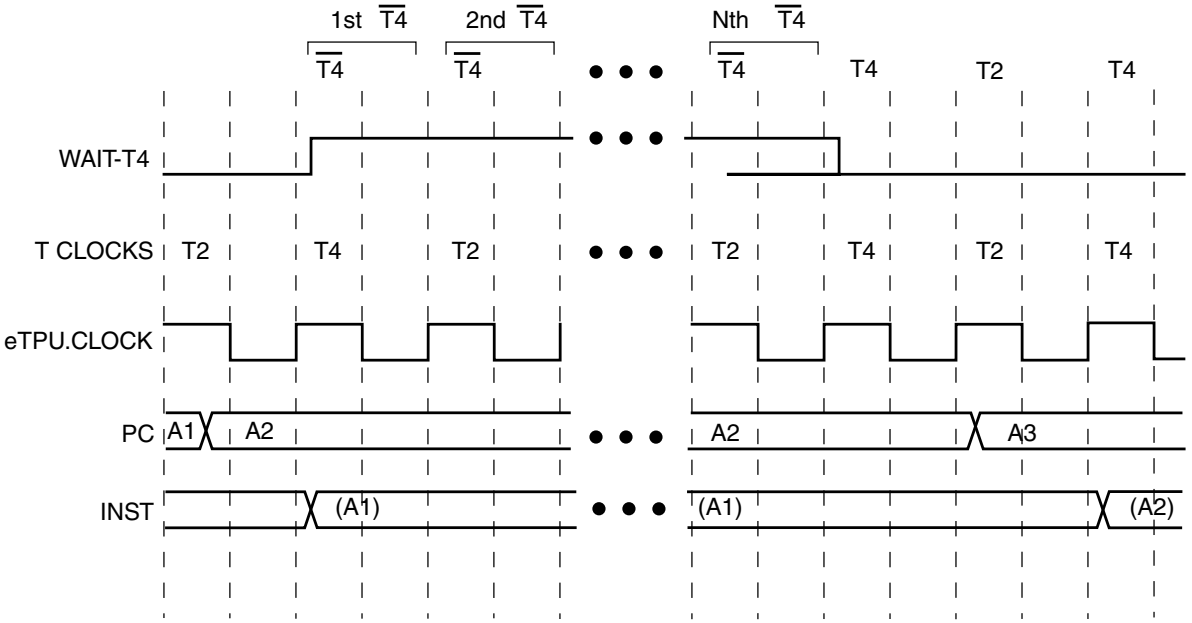


Figure 2-58. $\overline{T4}$ timing

2.16.1.2 Input/Output Signal Delays

The synchronizer, filter and edge detection logic delay the input signal transitions.

The Filter Delay varies with the filter clock (eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2) field FPSCK) and the filter mode used, as shown in the Table 2-45 . For any given transition, it depends on the phase of the filter clock when the input transition happens. In integration mode (TCRCLK filtering only), it also depends on the state of the integrator counter. The Total Delay is defined as the number of eTPU clock rising edges between the input transition and the setting of TDLA/B, TCR1/2 incrementing, or EAC tooth sensing (TCRCLK) in angle mode. The synchronizer delay is 2 or 3 eTPU clocks, depending on the phase of the synchronizer when the input transition happens. The edge detection takes 1 more eTPU clock. The total delays are, thus:

```
Min. Total Delay = Min. Synchronizer Delay + Min. Filter Delay + Edge Detection Delay
Min. Total Delay = 3 + Min. Filter Delay
Max. Total Delay = Max. Synchronizer Delay + Max. Filter Delay + Edge Detection Delay
Max. Total Delay = 4 + Max. Filter Delay
```

The channel filters can be bypassed, so nullifying the filter delays in the equations above.

See the chip-specific eTPU information for connections between eTPU and input glitch filters.

The channel output flip-flops drive the eTPU output signals directly, without any synchronous delays.

2.16.2 Channel Mode Summary

The initial state considered for all submodes is channel flags MRLEA and MRLEB asserted, MRLA, MRLB, TDLA and TDLB negated. From initial state one can follow the table below and verify how each submode behaves in a determined sequence of events. Note that the actions performed by an event type depend on all previous events following the initial state, for a given channel submode.

There are three columns for each event: one for event type, one for enable/disable actions and one for capture. **Event type** column can be matchA, matchB, transA and transB (for double transition modes). **Enable/disable actions** column (identified as "**[blocks] (enables)**" in column head) specifies which other events are enabled or disabled. Initially disabled events (specified in **"initially blocked"** column) are usually enabled by other events.

In double transition submodes, the first transition detected is always considered transA and the second is considered transB. This means that transA event actually enables the detection of transB event. This is not explicit in the table, since it is a general behavior for all double transition submodes.

A sequence of four events (two matches and two transitions) are necessary to describe the behavior of some channel submodes. When a determined sequence of events has less than four events, the other event columns are left blank.

Cells in an **"event type"** column that have light-grayed background indicate that a service request is generated. More than one event in the same event sequence can issue service request.

Note

The two tables below do not exhaust all possibilities of channel logic event sequences, because it does not account for possible microcode interventions. For instance, if matches are blocked by first transition and microcode resets TDLA, the matches become enabled again, and from this point on the channel behaves as if the first transition had never occurred.

Table 2-118. Channel Mode Summary 1 of 2

Mode	Initially Blocked	1st Event			2nd Event		
		Event Type	[Blocks] (Enables)[1]	Capture	Event Type	[Blocks] (Enables)	Capture
em_nb_st	none	match 1/2[2]	none	1/2	match 2/1	none	2/1

Table continues on the next page...

Table 2-118. Channel Mode Summary 1 of 2 (continued)

Mode	Initially Blocked	1st Event			2nd Event		
		Event Type	[Blocks] (Enables)[1]	Capture	Event Type	[Blocks] (Enables)	Capture
					trans 1	[matches]	both
		trans 1	[matches]	both	trans 2	none	2
em_nb_dt	none	match 1/2	none	1/2	match 2/1	none	2/1
		match 1	none	1	trans 1	none	1
		match 2	none	2	trans 1	[match 1]	1
		trans 1	[match 1]	1	match 2	none	2
					trans 2	[match 2]	2
em_b_st	none	match 1	[match 2]	both	trans 1	[matches]	both
		match 2	[match 1]	both			
		trans 1	[matches]	both	trans 2	none	2
em_b_dt	none	match 1	[match 2]	both	trans 1	none	1
		match 2	[match 1]	both			
		trans 1	[match 1]	1	match 2	none	2
					trans 2	[match 2]	2
bm_st	none	match 1/2	none	1/2	match 2/1	none	2/1
					trans 1	[matches]	both
		trans 1	[matches]	both	trans 2	none	2
bm_dt	none	match 1/2	none	1/2	match 2/1	none	2/1
					trans 1	none	1
		trans 1	none	1	match 1/2	none	none/2
					trans 2	[matches]	2
m2_st	trans 1	match 1	(trans 1)	1	match 2	none	2
					trans 1	[matches]	both
		match 1 and match 2	(trans 1)	both	trans 1	[matches]	both
		match 2	[match 1]	2	n/a	n/a	n/a
m2_dt	trans 1	match 1	(trans 1)	1	trans 1	none	1
					match 2	none	2
		match 1 and match2	(trans 1)	both	trans 1	none	1
		match 2	[match 1]	2	n/a	n/a	n/a
m2_o_st	trans 1 match 2	match 1	(match 2) (trans 1)	1	trans 1	[matches]	both
					match 2	[trans 1]	2
m2_o_dt	trans 1 match 2	match 1	(match 2) (trans 1)	1	trans 1	none	1
					match 2	[trans 1]	2
sm_st[3]	match 2	match 1	none	both	trans 1	none	both
		trans 1	[match 1]	both	trans 2	none	2
sm_dt	match 2	match 1	none	both	trans 1	none	1

Table continues on the next page...

Table 2-118. Channel Mode Summary 1 of 2 (continued)

Mode	Initially Blocked	1st Event			2nd Event		
		Event Type	[Blocks] (Enables)[1]	Capture	Event Type	[Blocks] (Enables)	Capture
		trans 1	none	1	match 1	none	2
		trans 1	none	1	trans 2	[match1]	2
sm_st_e[4]	match 2 trans 2	match 1	none	1	trans 1	none	1
		trans 1	[match 1]	1	n/a	n/a	n/a

1. Transition 1 always enables Transition 2.
2. Shading indicates that a service request is generated.
3. sm_st is compatible with TPU3 channel logic.
4. It is not possible to include all functionality of this submode in table.

Table 2-119. Channel Mode Summary 2of 2

Mode	Initially Blocked	3rd Event			4th Event		
		Event Type	[Blocks] (Enables)[1]	Capture	Event Type	[Blocks] (Enables)	Capture
em_nb_st	none	trans 1[2]	[matches]	both	trans 2	none	2
					n/a	n/a	n/a
		trans 2	none	2	trans 2	none	2
em_nb_dt	none	trans 1	none	1	trans 2	none	2
		match 2	none	2	trans 2	none	2
		trans 2	[match 2]	2	n/a	n/a	n/a
		trans 2	[match 2]	2	n/a	n/a	n/a
		trans 2	none	2	n/a	n/a	n/a
em_b_st	none	trans 2	none	2	n/a	n/a	n/a
em_b_dt	none	trans 2	none	2	n/a	n/a	n/a
		trans 2	none	2	n/a	n/a	n/a
bm_st	none	trans 1	[matches]	both	trans 2	none	2
		trans 2	none	2	n/a	n/a	n/a
bm_dt	none	trans 1	none	1	trans 2	[matches]	2
		match 2/1	none	2 / none	trans 2	[matches]	2
		trans 2	[matches]	2	n/a	n/a	n/a
		match 2/1	none	2 / none	trans 2	[matches]	2
		trans 2	[matches]	2	n/a	n/a	n/a
m2_st	trans 1	trans 1	[matches]	both	trans 2	none	2
		trans 2	none	2	n/a	n/a	n/a
		trans 2	none	2	n/a	n/a	n/a
m2_dt	trans 1	trans 2	[match 2]	2	n/a	n/a	n/a
		match 2	none	2	trans 2	none	2
		trans 1	none	1	trans 2	none	2
		trans 2	none	2	n/a	n/a	n/a

Table continues on the next page...

Table 2-119. Channel Mode Summary 2of 2 (continued)

Mode	Initially Blocked	3rd Event			4th Event		
		Event Type	[Blocks] (Enables)[1]	Capture	Event Type	[Blocks] (Enables)	Capture
m2_o_st	trans 1 match 2	trans 2	none	2	n/a	n/a	n/a
m2_o_dt	trans 1 match 2	trans 2	[match 2]	2	n/a	n/a	n/a
		match 2	[trans 2]	2	n/a	n/a	n/a
sm_st[3]	match 2	trans 2	none	2	n/a	n/a	n/a
sm_dt	match 2	trans 2	none	2	n/a	n/a	n/a
		trans 2	none	2	n/a	n/a	n/a
sm_st_e[4]	match 2 trans 2	n/a	n/a	n/a	n/a	n/a	n/a

1. Transition 1 always enables Transition 2.
2. Shading indicates that a service request is generated.
3. sm_st is compatible with TPU3 channel logic.
4. It is not possible to include all functionality of this submode in table.

2.16.3 MISC Algorithm

The MISC generator is based on the following polynomial:

$G(x) = 1 + x^1 + x^2 + x^{22} + x^{31}$ (equivalent to feedback mask = 0x80400007)

The MISC signature generation starts by clearing the MISC Accumulator value to 0 and preloading the MISC Counter with the highest SCM address. It then steps through each address decrementing the counter, reading 32 bit values and following the algorithm below:

```
If the least significant bit in MISC is 1 then
    MISC = MISC right shifted by 1 bit
    MISC = MISC XOR 0x80400007
else
    MISC = MISC right shifted by 1 bit
end if
MISC = MISC XOR RAM data
```

The code example below shows an excerpt of C code that calculates the MISC signature for a given array of data, based on the previous algorithm:

```
#define SCM_size    (MAX_SCM_ADDRESS / 4)    /* last byte address - converted to 32-bit word */
#define POLY        0x80400007               /* G(x) = 1 + x^1 +x^2 + x^22 + x^31 */
/*****************************************************************************
 FUNCTION : void calc_misc()
 PURPOSE : This function calculates the MISC value.
 INPUTS NOTES : none
 RETURNS NOTES : MISC value
 GENERAL NOTES : the array 'unsigned int data[]' represents the actual memory
                 array, organized in 32-bit words.
*****************************************************************************/
unsigned int calc_misc (void)
```

```
{
  int j; /* loop counter */
  unsigned int misc = 0;
  for (j = (SCM_size-1); j >= 0 ; j--) {   /* SCM_size has the number of 32-bit words in SCM
*/
      if (misc & 0x1) {
          misc >>= 1;
          misc ^= POLY;
      }
      else {
          misc >>= 1;
      }
      misc ^= data[j];          /* data[j] is the actual 32-bit word taken from the SCM
array
*/
  }
  return (misc);          /* final signature calculated */

};
```

The value calculated by this algorithm must be loaded into eTPU MISC Compare Register (eTPU_SCR_MISCCMPR) prior to activating the SCM MISC calculator in eTPU. Once the MISC calculator is activated (bit SCMMISEN in register eTPU Module Configuration Register (eTPU_SCR_MCR) is written to 1) eTPU itself will start this procedure[38] reading the SCM whenever allowed by microengine. At the end of the cycle, when all the array has been read and the SCM signature is calculated, the Host CPU can be notified via Global Exception if the MISC Accumulator does not match the value in eTPU_SCR_MISCCMPR.

The average time taken by MISC to complete the signature of the whole SCM can be given by the formula:

```
Average MISC period = S / (4 * F * (1 - L))
```

where f is clock frequency, S is SCM size in bytes and L is eTPU load (as a percentage of execution clocks over a period of time, including TST clocks).

Further detail on MISC calculation can be found on SCM Test - Multiple Input Signature Calculator. The application note **AN2192 - Detecting Errors in the Dual Port RAM (DPTRAM) Module** is also a good source of information (although it refers to TPU) on MISC signature.

38. eTPU MISC hardware is optimized to read 32-bit words from memory and to calculate this CRC in parallel, rather than shifting one bit at a time. The actual implementation inside eTPU, although bringing to the same results, does not match exactly the algorithm shown here.

Chapter 3
Fault Collection and Control Unit (FCCU)

3.1 FCCU chip-specific module information

This section describes the implementation of this module in the device.

3.1.1 FCCU fault recovery after NMI reaction

When an NMI (Non-Maskable Interrupt) is configured as the internal FCCU reaction for a specific NCF (fault state), the application software should not clear the core's MCSR[NMI] bit field during the execution of the NMI interrupt service routine, unless the FCCU fault source which generated the respective NMI reaction is cleared.

For details on MCSR[NMI] bit field clearing, see Machine Check Syndrome Register (MCSR). For details on FCCU Fault recovery see Fault recovery.

3.1.2 FCCU chip-specific register reset values

The following table shows FCCU registers whose reset values are specific to this chip.

Table 3-1. FCCU chip-specific register reset values

Register	Reset value
FCCU_CFG	See FCCU_CFG register bit value sources (N and C) by event.
FCCU_NCF_CFG0	0xFFFFFFFF
FCCU_NCF_CFG1	0xFFFFFFFF
FCCU_NCF_CFG2	0xFFFFFFFF
FCCU_NCFS_CFG0	0x00002000
FCCU_NCFS_CFG1	0x00000080
FCCU_NCFS_CFG2	0x00000000
FCCU_NCFS_CFG3	0x00000000

Table continues on the next page...

Table 3-1. FCCU chip-specific register reset values (continued)

Register	Reset value
FCCU_NCFS_CFG4	0x00000000
FCCU_NCFS_CFG5	0x00000000
FCCU_NCF_E0	0x00080040
FCCU_NCF_E1	0x00000000
FCCU_NCF_E2	0x00000000
FCCU_NCF_TOE0	0x00000000
FCCU_NCF_TOE1	0x00000000
FCCU_NCF_TOE2	0x00000000
FCCU_NCF_TO	0x0000_FFFF
FCCU_CFG_TO	0x0000_0006

3.1.3 FCCU_CFG register event bit values by source (N and C)

For each bit in the Configuration (FCCU_CFG) register, this table specifies the values of *N* and *C* for this chip. For information on how to interpret and use these values, see FCCU_CFG register bit value sources (N and C) by event.

Bit	N (chip-specific or user-specified value provided by the bit's associated NVMCFG signal)	C (chip-specific value provided by the bit's associated constant inside the module)
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	—	0
8	—	0
9	—	0
10	—	0
11	—	0
12	—	0
13	—	0
14	—	0
15	—	0

Table continues on the next page...

Bit	N (chip-specific or user-specified value provided by the bit's associated NVMCFG signal)	C (chip-specific value provided by the bit's associated constant inside the module)
16	0	0
17	0	0
18	0	0
19	—	0
20	0	0
21	0	0
22	0	0
23	0	0
24	0	0
25	0	0
26	0	0
27	0	0
28	0	1
29	0	0
30	0	0
31	0	0

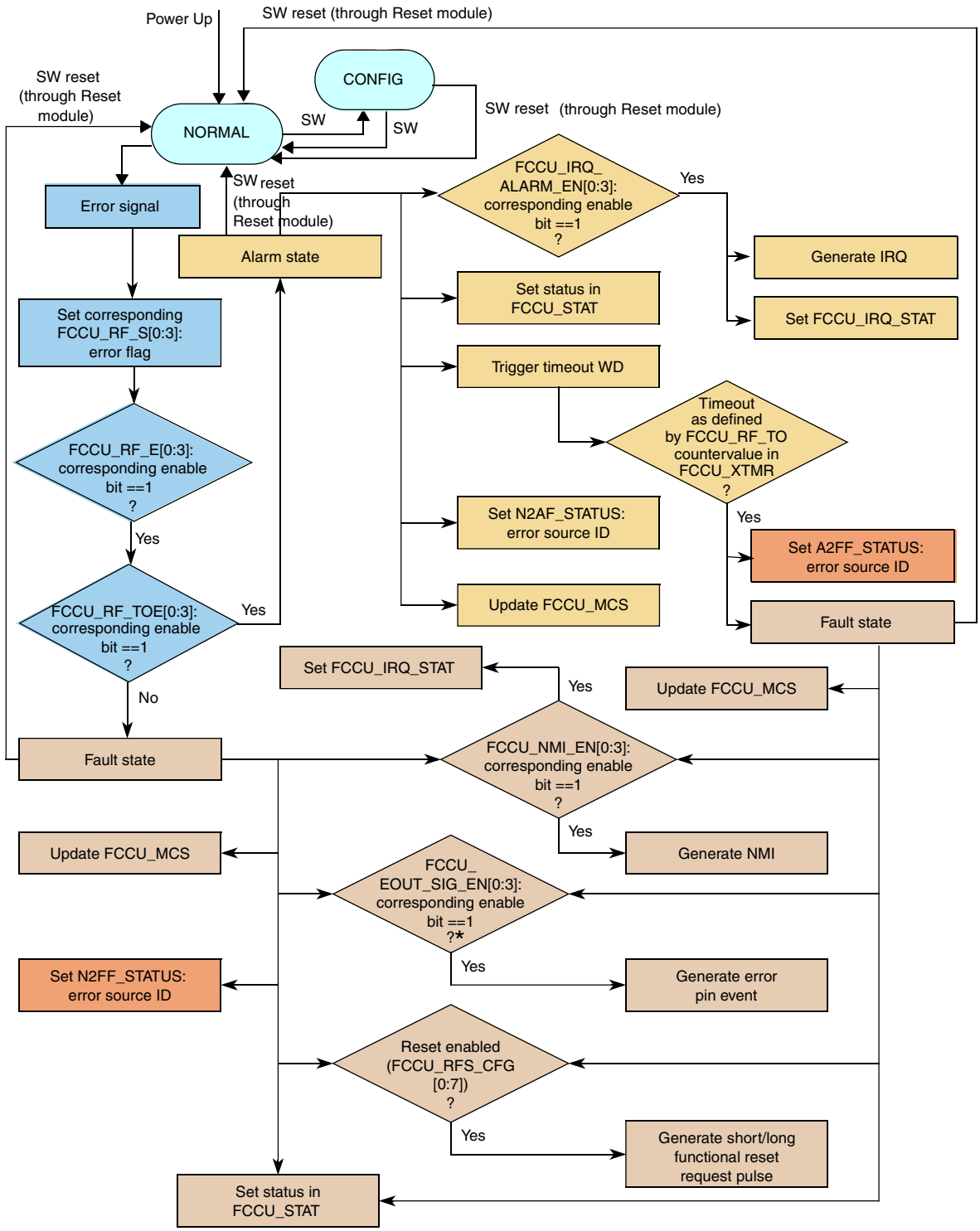
3.1.4 Available FCCU output signals

The following table shows the available FCCU output signals implemented on this chip.

Table 3-2. FCCU output signals

FCCU output signal name	Chip-top signal name
EOUT[0]	ERROR0
EOUT[1]	ERROR1

3.1.5 Error signal flow diagram



* This condition must be satisfied only when FCCU is configured for Bistable fault-output mode (FCCU_CFG[FOM]).

Figure 3-1. Error signal flow

3.1.6 FCCU failure inputs from other modules

The following table shows the different failure input signals of the FCCU, how they can be tested, and their default configuration after reset.

NOTE

All the faults shown in the following table require a long functional reset.

NOTE

FCCU Fault Channels 22, 23, and 24 may be asserted after reset. FCCU Fault Channels 51, 52, 69 through 84, 92, and 93 may be asserted during the execution of MBIST and LBIST. Before running MBIST or LBIST, these channels should be configured to NO REACTION. If any of these channels become asserted during MBIST or LBIST testing, the fault should simply be cleared and no further action should be taken.

Table 3-3. FCCU Fault Inputs

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
0	No Reaction	Temp sensor	Using PMC_TS_FIR register to set the fault at source + Via FCCU Fake bit	Temperature Sensor Event Triggered by PMC_ESR_TD[TEMP1_2, TEMP1_0, TEMP0_2, TEMP0_0] and PMC_TS_FIR[TSNS_FAULT] . For details, see the descriptions of the PMC_ESR_TD and PMC_TS_FIR registers.
1	No Reaction	PMC	Using PMC FIR register to set the fault at source + Via FCCU Fake bit	LVD event to FCCU Triggerred by PMC_LVD_HVD_EVENT_STATUS[LVD_PMC, LVD_SAR_ADC, LVD_IO, LVD_FLASH, LVD_CORE_HOT, LVD_CORE_COLD] and PMC_FIR[LVD_FAULT]. For details, see the descriptions of the PMC_LVD_HVD_EVENT_STATUS and PMC_FIR registers.
2	No Reaction	PMC	Using PMC FIR register to set the fault at source + Via FCCU Fake bit	HVD Event to FCCU Triggered by PMC_LVD_HVD_EVENT_STATUS[HVD_FLASH, HVD_HV, HVD_CORE]. For details, see the descriptions of the PMC_LVD_HVD_EVENT_STATUS register.

Table continues on the next page...

Table 3-3. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
3	No Reaction	PMC	Using PMC FIR register to set the fault at source + Via FCCU Fake bit	PMC Self Test Failure Event PMC hard/soft self-test failed: LVD/HVD **NOTE:** When this occurs, single LVD or HVD self-tests should be executed to identify which detection mechanism failed in the SOC: LVD or HVD. For details, see the description of the PMC_SELF_TEST_UM_VD_REG register as well as the flag bits associated to NCF3 (PMC_SELF_TEST_UM_VD_REG [ST_RESULT]). Self test bit definition indicates all valid values for PMC_SELF_TEST_UM_VD_REG[VD_ST_CTRL] while executing single self-tests. LVD-HVD self test mechanism provides details about the available PMC self-tests
4	No Reaction	SSCM	write wrong data to address + Via FCCU Fake bit to set the fault at error path	See SSCM_STATUS[CERS] field description.
5	No Reaction	SSCM	write wrong data to address + Via FCCU Fake bit to set the fault at error path	See SSCM_STATUS[CER] field description.
6	long reset	STCU	SW check of BIST MISR + Via FCCU Fake bit to set the fault at source	Critical Fault of STCU. Set when STCU2_ERR_STAT[UFSF] is set.
7	No Reaction	STCU	SW check of BIST MISR + Via FCCU Fake bit to set fault at source	Non critical Fault of STCU. Set when STCU2_ERR_STAT[RFSF] is set.
8	No Reaction	STCU	STCU activation during user application run Via FCCU Fake bit to set the fault at source	Critical Fault in case LBIST or MBIST control signals go to wrong condition during User application. This fault is set when the self-test was triggered without STCU2_RUN[RUN] bit programming.
9	No Reaction	Checker Shell	Via FCCU Fake bit	out of lockstep fault detected for miscellaneous safety core output
10	No Reaction	Checker Shell	Via FCCU Fake bit to set the fault at error source	out of lockstep fault detected for safety Core dtcm array interface signal
11	No Reaction	Checker Shell	Via FCCU Fake bit to set the fault at error source	out of lockstep fault detected for safety Core I Cache array interface signals
12	Reserved	Reserved		Reserved
13	No Reaction	Checker Shell	Via FCCU Fake bit to set the fault at error source	out of lockstep fault detected for safety Core Amba interface signals
14	No Reaction	DMA Checker Shell	Via FCCU Fake bit to set the fault at error source	out of lockstep fault detected for DMA

Table continues on the next page...

Table 3-3. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
15	No Reaction	DMC Checker	Via FCCU Fake bit to set the fault at error source	out of lockstep fault detected for core0 TCM DMC
16	RESERVED	RESERVED		RESERVED
17	No Reaction	SWT3	Emulate fault using functionality + Via FCCU Fake bit to set the fault at error path	Software watchdog timer reset request. This input must be mapped only to reset, and must not be mapped to an alarm.
18	RESERVED	RESERVED		RESERVED
19	Long Reset	SWT1	Emulate fault using functionality (SW test by not triggering SWT during startup) + Via FCCU Fake bit to set the fault at error path	Software watchdog timer reset request. This input must be mapped only to reset, and must not be mapped to an alarm.
20	No Reaction	SWT0	Emulate fault using functionality (SW test by not triggering SWT during startup) + Via FCCU Fake bit to set the fault at error path	Software watchdog timer interrupt
21	No Reaction	SWT0	Emulate fault using functionality (SW test by not triggering SWT during startup) +Via FCCU Fake bit to set the fault at error path	Software watchdog timer reset request. This input must be mapped only to reset, and must not be mapped to an alarm.
22	No Reaction	MEMU	Injection of RAM ECC error via ECC bit access and later reading + Via FCCU Fake bit to set the fault at error path+ LBIST	System RAM Correctable Error detected
23	No Reaction	MEMU	Injection of RAM ECC error via ECC bit access and later reading + Via FCCU Fake bit to set the fault at error path + LBIST	System RAM Un-Correctable Error detected
24	No Reaction	MEMU	Via FCCU Fake bit to set the fault at error path + LBIST	System RAM error Overflow
25	No Reaction	MEMU + eTPU Memory Error Correction Interrupt	Via FCCU Fake bit to set the fault at error path + LBIST	Peripheral RAM Correctable error detected. Set eTPU_MESR_MECR[MECIE] to enable this error.
26	No Reaction	MEMU + eTPU Global Exception	Via FCCU Fake bit to set the fault at error path + LBIST	Peripheral RAM UN-Correctable error detected. Set eTPU_MESR_MECR[MECIE] to enable this error.
27	No Reaction	MEMU	Via FCCU Fake bit to set the fault at error path + LBIST	Periph RAM error Overflow
28	No Reaction	MEMU	Reading preprogrammed correctable ECC error from	Flash Correctable Error detected

Table continues on the next page...

Table 3-3. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
			UTEST flash memory + Via FCCU Fake bit to set the fault at error path + LBSIT	NCF correlated to C55FMC_MCR[SBC] bit **NOTE:** From the MEMU perspective this is indicated by MEMU_FLASH_CERR_STSn. For details, see the description of the C55FMC_MCR and MEMU_FLASH_CERR_STSn registers.
29	No Reaction	MEMU	Reading preprogrammed uncorrectable ECC error from UTEST flash memory + Via FCCU Fake bit to set the fault at error path + LBIST	Flash Un-Correctable Error detected NCF correlated to C55FMC_MCR[EER] bit **NOTE:** From the MEMU perspective this is indicated by MEMU_FLASH_UNCERR_STSn. For details, see the description of the C55FMC_MCR and MEMU_FLASH_UNCERR_STS registers.
30	No Reaction	MEMU	Via FCCU Fake bit to set the fault at error path + LBIST	Flash Error Overflow For details on conditions which trigger this NCF, see section Handling overflows (Multiple error reporting). **NOTE:** From the MEMU perspective this is indicated by MEMU_FLASH_OFLWn. For details, see the description of the MEMU_FLASH_OFLWn register.
31	RESERVED	RESERVED		RESERVED
32	No Reaction	PLL0	Forced loss by PLL reconfiguration + Via FCCU Fake bit to set the fault at error path	Loss of Lock detected for NON-FM PLL **NOTE:** This source should be configured to generate a reset instead of an interrupt if the source is used as the system clock.
33	No Reaction	PLL1	Forced loss by PLL reconfiguration + Via FCCU Fake bit to set the fault at error path	Loss of Lock detected for FM PLL **NOTE:** This source should be configured to generate a reset instead of an interrupt if the source is used as the system clock.
34	No Reaction	CMU_PLL	Test by misconfiguration + Via FCCU Fake bit to set the fault at error path	OLR Event from crystal clock monitor circuit of cmu_pll **NOTE:** This source should be configured to generate a reset instead of an interrupt if the source is used as the system clock.

Table continues on the next page...

Table 3-3. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
35	No Reaction	CMU_PLL	Test by misconfiguration + Via FCCU Fake bit to set the fault at error path	FLL/FHH event from PLL Clock Monitor circuit. This fault is set when CMU_0 detects an FHH or FLL event **NOTE:** To identify which CMU module raised this NCF, read the CMU_ISR[FHHI/FLLI] bit fields from all relevant CMU instances. For details, see the description of the CMU_ISR register. For available CMU's, see sections Clock input sources and CMU registers and field availability.
36	No Reaction	System clock CMU	Test by misconfiguration + Via FCCU Fake bit to set the fault at error path	FLL/FHH event from System clock monitor circuit. This fault is set when one of CMU_1 to CMU_3 detects an FHH or FLL event. **NOTE:** To identify which CMU module raised this NCF, read the CMU_ISR[FHHI/FLLI] bit fields from all relevant CMU instances. For details, see the description of the CMU_ISR register. For available CMU's, see sections Clock input sources and CMU registers and field availability.
37	No Reaction	Other CMU	Emulate fault using functionality + Via FCCU Fake bit to set the fault at error path	FLL/FHH event from other clock monitor circuit. This fault is set when one of CMU_4 to CMU_12 detects an FHH or FLL event. **NOTE:** To identify which CMU module raised this NCF, read the CMU_ISR[FHHI/FLLI] bit fields from all relevant CMU instances. For details, see the description of the CMU_ISR register. For available CMU's, see sections Clock input sources and CMU registers and field availability.
38	No Reaction	Periph AXBS	Using XBIC_EIR register to set the fault at source+Via FCCU Fake bit to set the fault at error path	Peripheral Crossbar alarm to FCCU
39	No Reaction	Comp AXBS	Using XBIC_EIR register to set the fault at source+Via FCCU Fake bit to set the fault at error path	Computational Crossbar alarm to FCCU
40	No Reaction	Flash	Via FCCU Fake bit to set the fault at error source + LBIST	alarm indicating the flash detected an error in the ECC correction logic through an EDC

Table continues on the next page...

Table 3-3. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
41	No Reaction	Flash Controller	Via FCCU Fake bit to set the fault at error path + LBIST	alarm indicating the flash controller detected an error in the address ECC manipulation logic through an EDC
42	No Reaction	Flash	Via FCCU Fake bit to set the fault at error path + LBIST	alarm indicating the Flash Array Controller detected an Encode Address Error, such as multiple address lines selected or no address lines selected **NOTE:** Address Encode Errors detected within Flash Array [FCCU channel 42] will also assert the P-Flash [FCCU channel 43]
43	No Reaction	Flash Controller	Via FCCU Fake bit to set the fault at error path + LBIST	alarm indicating the flash controller detected a transaction monitor mismatch when compared to the flash safety feedback outputs
44	No Reaction	Flash Controller	Via FCCU Fake bit to set the fault at error path + LBIST	alarm indicating the flash controller detected a transaction monitor mismatch when compared to the on-chip calibration RAM safety feedback outputs
45	No Reaction	Flash Controller	Via FCCU Fake bit to set the fault at error path + LBIST	alarm indicating the flash controller detected a transaction monitor mismatch in the pseudo-replicated calibration evaluation hardware
46	RESERVED	RESERVED		RESERVED
47	No Reaction	Platform Gasket	Via FCCU Fake bit to set the fault at error path	2:1 bridging gasket transaction monitor alarm
48	No Reaction	Platform Gasket	Via FCCU Fake bit to set the fault at error path	1:2 bridging gasket transaction monitor alarm
49	No Reaction	SMPU	Via FCCU Fake bit to set the fault at error path + LBIST	Computational MPU alarm to the fault collection unit An access error has happened due to restrictions specified by the MPU region descriptors. This is in addition to the MPU responding to the requesting master with an error response.
50	No Reaction	SMPU	Via FCCU Fake bit to set the fault at error path + LBIST	Peripheral MPU alarm to the fault collection unit An access error has happened due to restrictions specified by the MPU region descriptors. This is in addition to the MPU responding to the requesting master with an error response.
51	No Reaction	Decorated memory Controller	No fake faults supported	AIPS decorated storage transaction monitor error: this monitor basically ensure that the Decorated Memory Controller is not modifying regular AHB transactions when a non-decorated storage instruction is performed. If a non-decorated storage instruction is executed, the

Table continues on the next page...

Table 3-3. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
				DMC should pass the transaction through unaltered.
52	No Reaction	Decorated memory Controller	No fake faults supported	AIPS decorated storage transaction monitor error: this monitor basically ensure that the Decorated Memory Controller is not modifying regular AHB transactions when a non-decorated storage instruction is performed. If a non-decorated storage instruction is executed, the DMC should pass the transaction through unaltered.
53	No Reaction	PRAM Controller	Via FCCU Fake bit to set the fault at error path	PRAM Controller/AHB bus transaction monitor error output
54	RESERVED	RESERVED		RESERVED
55	No Reaction	Core	Willfully cause an appropriate exception (for example, access a non-existing address) + Via FCCU Fake bit to set the fault at error path	machine check has occurred
56	No Reaction	Core	Willfully cause an appropriate exception (for example, access a non-existing address) + Via FCCU Fake bit to set the fault at error path	machine check has occurred
57	No Reaction	SOC	Disabling Checker core while safety core is on+ Via FCCU Fake bit to set the fault at error path	Lock Step Disabled
58	No Reaction	RGM	RGM “safe mode” request + Via FCCU Fake bit to set the fault at error path + LBIST	Safe Mode request from RGM
59	RESERVED	RESERVED		-
60	No Reaction	IOMUX	External activation of EIN. External entity has to pull the error input low. + Via FCCU Fake bit to set the fault at error path	Error Input
61	RESERVED	RESERVED		RESERVED
62	No Reaction	DCI	Via FCCU Fake bit to set the fault at error path + LBIST	Debug Fault Condition to MCU • If any JTAG_aux_tms is high from DCI • If Jtag_rst_b is low from DCI • If JTAG mode is high from DCI • If nex_rst_b is high from DCI
63	No Reaction	SAR ADC0/1/2/3	VIA SAR-ADC STCR2[SERR] bit to set the fault at error source + Via FCCU Fake bit to set the fault at error path	Non Critical fault from ALL SARs

Table continues on the next page...

Table 3-3. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
64	No Reaction	SAR ADC0/1/2/3	VIA SAR-ADC STCR2[SERR] bit to set the fault at error source + Via FCCU Fake bit to set the fault at error path	Critical Fault from All SARs
69	No Reaction	CoreA Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data0 feedback check alarm
70	No Reaction	CoreA Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data1 feedback check alarm
71	No Reaction	CoreA Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data2 feedback check alarm
72	No Reaction	CoreA Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data3 feedback check alarm
73	No Reaction	CoreA Instruction Tag	Via FCCU Fake bit to set the fault at error path	IC tag feedback check alarm
74	No Reaction	CoreA DMEM Controller	Via FCCU Fake bit to set the fault at error path	DTCM0 feedback check alarm
75	No Reaction	CoreA DMEM Controller	Via FCCU Fake bit to set the fault at error path	DTCM1 feedback check alarm
76	No Reaction	CoreA IMEM Controller	Via FCCU Fake bit to set the fault at error path	ITCM feedback check alarm
77	No Reaction	CoreB Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data0 feedback check alarm
78	No Reaction	CoreB Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data1 feedback check alarm
79	No Reaction	CoreB Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data2 feedback check alarm
80	No Reaction	CoreB Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data3 feedback check alarm
81	No Reaction	CoreB Instruction Tag	Via FCCU Fake bit to set the fault at error path	IC tag feedback check alarm
82	No Reaction	CoreB DMEM Controller	Via FCCU Fake bit to set the fault at error path	DTCM0 feedback check alarm
83	No Reaction	CoreB DMEM Controller	Via FCCU Fake bit to set the fault at error path	DTCM1 feedback check alarm
84	No Reaction	CoreB IMEM Controller	Via FCCU Fake bit to set the fault at error path	ITCM feedback check alarm

Table continues on the next page...

Table 3-3. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
85	No Reaction	CoreA/0 interrupt controller	This alarm can be triggered by enabling the INTC_MMR and setting the INTC_TIMER value to be larger than the INTC_LAT value (which itself needs to be non-zero). + Via FCCU Fake bit to set the fault at error path	INTC monitor error
86	No Reaction	CoreB/1 Interrupt controller	This alarm can be triggered by enabling the INTC_MMR and setting the INTC_TIMER value to be larger than the INTC_LAT value (which itself needs to be non-zero). + Via FCCU Fake bit to set the fault at error path	INTC monitor error
87	No Reaction	Decorated memory Controller	Via FCCU Fake bit to set the fault at error path	PRAM decorated storage transaction monitor error
88	No Reaction	Computational SMPU	Via FCCU Fake bit to set the fault at error path + LBIST	MPU alarm to the fault collection unit An internal error has happened in the operation of the MPU. Example: MPU erroneously blocked transactions without a real permissions violation will assert this NCF.
89	No Reaction	peripheral SMPU	Via FCCU Fake bit to set the fault at error path + LBIST	MPU transaction monitor error output An internal error has happened in the operation of the MPU. Example: MPU erroneously blocked transactions without a real permissions violation will assert this NCF.
90	No Reaction	Core0/1 IMEM MBIST	Via FCCU Fake bit to set the fault at error path	Mbist multi bit error of IMEM (Core0 or Core1)
91	No Reaction	Flash	Via FCCU Fake bit to set the fault at error path + LBIST	Flash Read Voltage error or reference current error
92	No Reaction	Core0/1 DMEM MBIST	Via FCCU Fake bit to set the fault at error path	Mbist multi bit error of DMEM (Core0 or Core1)
93	No Reaction	Core0/1 ICACHE/TAG MBIST	Via FCCU Fake bit to set the fault at error path	Mbist multi bit error of ICACHE(Core0 or Core1)
94	No Reaction	SRAM MBIST	Via FCCU Fake bit to set the fault at error path	Mbist Multibit error of SRAM
95	No reaction	All Peripheral RAM MBIST	Via FCCU Fake bit to set the fault at error path	Mbist multi bit error from all peripheral RAM

3.1.7 Lockstep Status

RCCU status is evaluated in FCCU_NCF[57] channel.

In debug mode, the cores do not work in lockstep mode and the RCCUs are disabled. The actual enablement/disablement state of the RCCUs (including any automatic disablement by debug) can be read by software.

3.1.8 FOSU Count Value

The fixed preprogrammed value for FOSU_Count is 0x1F40.

3.1.9 FCCU_CFG[FCCU_SET_AFTER_RESET]

FCCU_CFG[FCCU_SET_AFTER_RESET] is not used to control EOUT pads on this chip and hence there will be no effect of writing to this bit. FCCU EOUT pads OBE are controlled either via SIUL or DCF records.

3.2 Introduction

The Fault Collection and Control Unit (FCCU) offers a hardware channel to collect faults and to place the device into a safe state when a failure in the device is detected. No CPU intervention is requested for collection and control operation.

The FCCU offers a systematic approach to fault collection and control. The distinctive features of the module are:

- Redundant collection of fault information from safety relevant modules on the device
- Collection of test results
- Configurable fault control
- Configurable internal reactions for each NCF:
  - No reaction
  - IRQ
  - Short functional reset

  - Long functional reset
  - NMI
- External reactions via configurable output pins
- Watchdog timer for the reconfiguration phase
- Lockable configuration
- Nonvolatile memory interface (NVMCFG signals) for configuration loading
- Self checking capabilities: FSM redundancy and parity check for the configuration register
- One of the fault-output (EOUT) signals is high to indicate operational (OK) state (in bi-stable operation mode). The above is not true in case the error out protocol is configured to be a toggling protocol, for example, dual rail and time switching.
- After power on, the EOUT signals have high impedance.[1] They show operational state only after configuration by software.
- In case of a failure event or on software request for Error pin indication, the pin(s) are set to faulty state for a minimum time T_min, even if SW tries to release it before (for the case of error pin configured in Bi-stable mode only).
- The overall fault detection, processing and indication time is less than 10 ms.
- The actual maximum time is 5 CLKSAFE cycles**.**

The FCCU circuitry is checked at the start-up by the self-checking procedure. The FCCU is operative with the default configuration immediately after the completion of the self-checking procedure. Internal (short or long functional reset request pulse, interrupt request) and external (EOUT signaling) reactions are statically defined or programmable. The default configuration can be modified only in the CONFIG state. FCCU is designed to function when CLKSYS is faster than the CLKSAFE clocks.

3.2.1 Acronyms and abbreviations

Term	Description
CF	Critical fault
CPU	Central processing unit
EOUT	Error out
FOSU	FCCU output supervision unit

Table continues on the next page...

1. Actual value depends on device specific setting at pad level.

Term	Description
FSM	Finite state machine
INTC	Interrupt controller
intf	Interface
IRQ	Interrupt request
MC_ME	Mode entry module
NCF	Noncritical fault
NMI	Nonmaskable interrupt
NVM	Nonvolatile memory
SoC	System-on-chip
STCU	Self-test control unit
SW	Software
WKPU	Wake-up unit

3.3 Main features

- Management of noncritical faults
- HW or SW fault recovery management
- Fault detection and collection
- Fault injection (fake faults)
- External reaction (fault state): EOUT signaling. Error indication via the pin(s) is controlled by the FCCU.
- Internal chip reactions (alarm state): interrupt request
- Internal chip reactions (fault state):
  - long functional reset request pulse
  - short functional reset request pulse
  - NMI
- Bi-Stable, Dual-Rail and Time Switching output protocols on EOUT
- Internal (to the FCCU) watchdog timer for the reconfiguration phase
- Configuration lock
- Nonvolatile memory interface (NVMCFG signals) for configuration loading
- Self-checking capabilities: FSM redundancy and parity check for the configuration registers

3.4 Block diagram

The following figure is a top-level diagram of the FCCU module.

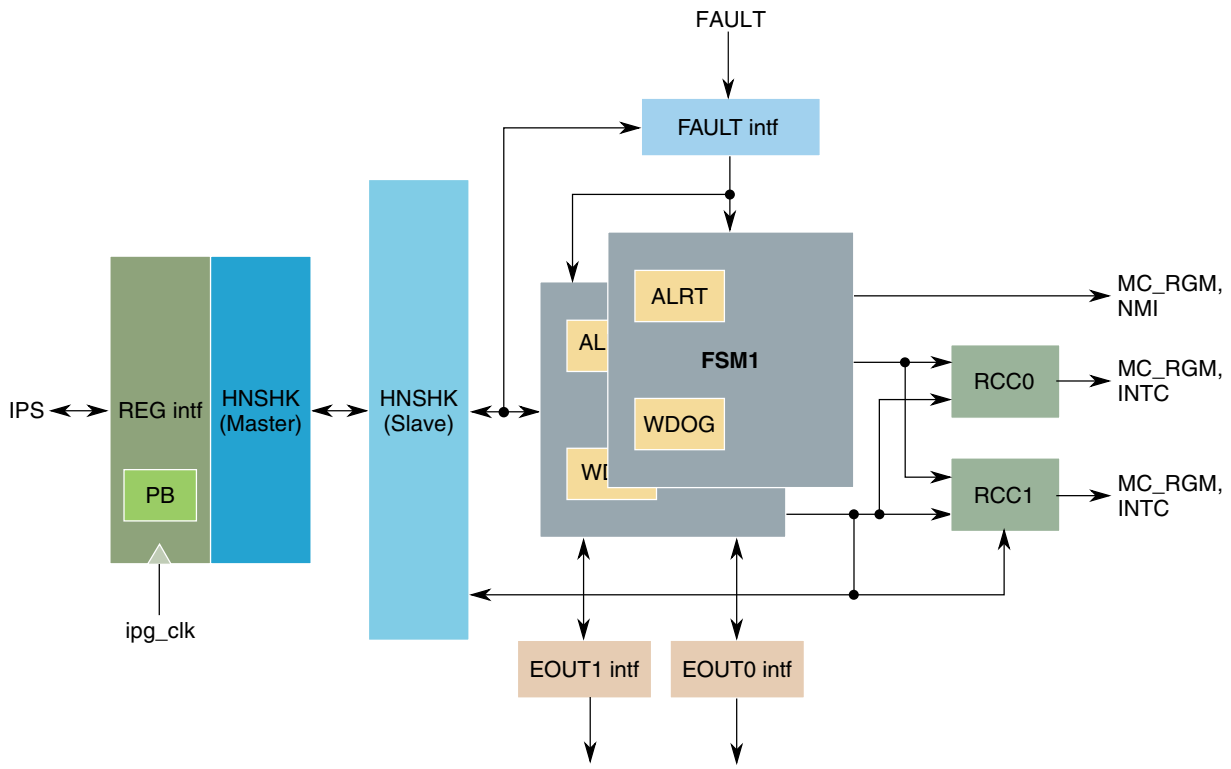


Figure 3-2. FCCU block diagram

This table describes the FCCU submodules.

Table 3-4. FCCU submodules

Submodule	Description
REG intf	Includes the register file, the IPS bus interface, the IRQ interface and the parity block (PB) for the configuration registers
HNSHK blocks (master and slave blocks)	Includes the FSM's ability to support the handshake between the REG interface and the FSM unit due to the usage of 2 asynchronous clocks (IPS system clock and RC oscillator clock)
FSM unit	Implements the main functions of the FCCU. The FSM also includes the: • Watchdog timer (WDG) • Alarm timer (ALRT)
FAULT intf	Implements the interface for the fault conditioning and management
EOUTx units	Implement the output stage to manage the EOUT interfaces
RCCx units	Implement the redundancy control checker to monitor the FSMx unit state and its configuration

3.5 Signal description

3.5.1 Signals

This table describes the signals on the boundary of FCCU.

Signal	Direction	Function	Active level	Clock
Clock and reset				
CLKSYS	Input	System Clock—Supplies the primary clock.	Rising edge	—
CLKSAFE0	Input	Safe Clock 0—Supplies one of two 16 MHz clocks for use by various redundant submodules. This clock and CLKSAFE1 are often referred to as the same clock (CLKSAFE).	Rising edge	—
CLKSAFE1	Input	Safe Clock 1—Supplies one of two 16 MHz clocks for use by various redundant submodules. This clock and CLKSAFE0 are often referred to as the same clock (CLKSAFE).	Rising edge	—
EOUT interface				
EIN[1:0]	Input	Error Input—Provide the mechanism for FCCU to capture the states that off-chip logic drives on the associated EOUT[1:0] signals. (On the chip, the EOUT signals are connected to their corresponding EIN signals.) FCCU captures the states in the EIN1 and EIN0 fields.	—	Asynchronous
EOUT[1:0]	Output	Error Output—Indicate FCCU's condition (Faulty, Nonfaulty, or Configuration) to off-chip logic.	Programmable	CLKSAFE0, CLKSAFE1
FOSU interface				
FIF	Output	FCCU In Fault—Indicates to the FOSU module, if present, that FCCU is recognizing a fault. FCCU asserts and then deasserts this signal in two cases: *Case 1* When the EOUT signals are programmed to be always low (FCCU_SET_CLEAR), FCCU asserts this signal until the EOUT signals are programmed not to be always low. *Case 2*	High	CLKSAFE0, CLKSAFE1

Table continues on the next page...

Signal	Direction	Function	Active level	Clock
		When the EOUT signals are programmed not to be always low (FCCU_SET_CLEAR), FCCU asserts this signal when all of the following are true: • FCCU enters Fault state as the result of a fault. • If the fault is a noncritical fault, and the EOUT signals are in a fault-output mode (FOM) other than Bistable, the associated EOUTENx field is set to enabled. FCCU deasserts this signal when FCCU leaves Fault state.		
WKPU interface				
NMIWKPU	Input	WKPU Nonmaskable Interrupt Request—Receives nonmaskable interrupt requests from the WKPU module.	Low	CLKSYS
WAKEWKPU	Output	Wake WKPU—Wakes the WKPU module to bring the chip out of dormant state when FCCU enters Fault state as the result of a noncritical fault for which NMI is enabled as the reaction (FCCU_NMI_ENa[NMIENx]).	High	CLKSAFE0, CLKSAFE1
Interrupt interface				
NMIOUT	Output	Nonmaskable Interrupt Output—Sends a nonmaskable interrupt request to the processor core or cores when FCCU enters Fault state as the result of a noncritical fault for which NMI is enabled as the reaction (FCCU_NMI_ENa[NMIENx]).	Low	CLKSYS
NVM interface				
NVMCFG[21:0]	Input	Nonvolatile Memory Configuration—Provide the values of certain bits in the Configuration (FCCU_CFG) register when certain events occur. See Configuration (FCCU_CFG). Some or all of these signals may be driven by bits in a data record in nonvolatile (flash) memory or may be tied high (1) or low (0). See FCCU_CFG register bit value sources (N and C) by event.	Static value	—
SSCMDONE	Input	SSCM Done—After an FCCU reset, indicates that the SSCM module is now driving any NVMCFG signals	High	CLKSYS

Signal	Direction	Function	Active level	Clock
		that are associated with a data record stored in nonvolatile (flash) memory.		

3.6 Put FCCU in Configuration or Normal state

3.6.1 Introduction

You put FCCU in Configuration state to change its configuration. You put FCCU in Normal state to save changes to the configuration.

3.6.2 About changing states

When changing states, keep the following in mind:

- To put FCCU in Configuration state, FCCU must be in Normal state.
- After FCCU is reset, the configuration is temporarily locked. You must temporarily unlock the configuration before you can put FCCU in Configuration state.
- When FCCU is in Configuration state, FCCU doesn't actually save the changes you make to the configuration. To save changes to the configuration, you must manually put FCCU in Normal state. If FCCU automatically leaves Configuration state and enters Normal state because the configuration-timer interval (TO) expires (called a Configuration-state timeout), FCCU changes the value of the Configuration (FCCU_CFG) register to its Configuration-state-timeout value and the value of each of the other configuration registers to its reset value. For information on the Configuration-state timeout value, see FCCU_CFG register bit value sources (N and C) by event. For a list of configuration registers, see Configuration registers.

3.6.3 Put FCCU in Configuration state

1. Temporarily unlock the configuration (see Temporarily unlock the configuration).
2. To determine FCCU's state, run the OP3 operation (see Run an operation).
3. Check the operation status (OPS).
   - If the operation status is Successful, go to step 4.
   - If the operation status is Aborted, go to step 2.
4. Check the FCCU status (STATUS).
   - If the FCCU status is Configuration, FCCU is in Configuration state. Stop.

- If the FCCU status is Normal, go to step 5.
- If the FCCU status is Alarm or Fault, go to step 7.

5. To put FCCU in Configuration state, run the OP1 operation (see Run an operation).
6. Check the operation status (OPS).
   - If the operation status is Successful, FCCU is in Configuration state. Stop.
   - If the operation status is Aborted, the configuration is probably permanently locked. Go to step 9.
7. Recover all faults.
8. Go to step 2.
9. Reset FCCU.

3.6.4 Put FCCU in Normal state

1. To determine FCCU's state, run the OP3 operation (see Run an operation).
2. Check the operation status (OPS).
   - If the operation status is Successful, go to step 3.
   - If the operation status is Aborted, go to step 1.
3. Check the FCCU status (STATUS).
   - If the FCCU status is Normal, FCCU is in Normal state. Go to step 8.
   - If the FCCU status is Configuration, go to step 4.
   - If the FCCU status is Alarm or Fault, go to step 6.
4. To put FCCU in Normal state, run the OP2 operation (see Run an operation).
5. Check the operation status (OPS).
   - If the operation status is Successful, FCCU is in Normal state. Go to step 8.
   - If the operation status is Aborted, go to step 4.
6. Recover all faults.
7. Go to step 1.
8. If you want to prevent changes to the configuration, lock it:
   - To require a key to unlock the configuration, temporarily lock the configuration (see Temporarily lock the configuration).
   - To require a reset of FCCU to unlock the configuration, permanently lock the configuration (see Permanently lock the configuration).

3.7 Run operations

3.7.1 Introduction

You run operations to perform actions such as putting FCCU in Configuration state or reading a timer. For a complete list of operations you can run, see OPR.

3.7.2 About running operations

When running operations, keep the following in mind:

- FCCU ignores any operations initiated while the operation status is In Progress.
- Certain operations must be unlocked before you can initiate them. After you initiate them, they are again locked.

3.7.3 Run an operation

1. Check the operation status (OPS).
2. If the operation status is In Progress, go to step 1.
3. If the operation must be unlocked before you can initiate it, unlock the operation (CTRLK) using a 32-bit write.

NOTE

The Control Key (FCCU_CTRLK) register used in this step and the Control (FCCU_CTRL) register used in the next step must be written with consecutive instructions. Do not use read-modify-write instructions, such as bit-field instructions, to modify these registers.

4. Initiate the operation (OPR) using a 32-bit write.
5. Check the operation status (OPS).
6. If the operation status is In Progress, go to step 5.

The operation status is now Idle (OP15 only), Aborted, or Successful.

3.8 Lock and unlock the configuration

3.8.1 Introduction

After you configure FCCU, you can lock its configuration to prevent accidental changes.

3.8.2 About locking the configuration

When locking the configuration, keep the following in mind:

- If you attempt to lock the configuration while FCCU is in Configuration state, FCCU doesn't actually lock the configuration until FCCU leaves Configuration state—that is, either you put FCCU in Normal state, or FCCU puts FCCU in Normal state because the configuration-timer interval (TO) expires.
- After you permanently lock the configuration, you must reset FCCU before you can put FCCU in Configuration state.

3.8.3 Temporarily lock the configuration

1. From Supervisor mode, write a lock value to TRANSKEY.
2. Put FCCU in Normal state if it's in Configuration state (see Put FCCU in Normal state).

3.8.4 Temporarily unlock the configuration

- From Supervisor mode, write the unlock value to TRANSKEY.

3.8.5 Permanently lock the configuration

1. From Supervisor mode, write the lock value to PERMNTKEY.
2. Put FCCU in Normal state if it's in Configuration state (see Put FCCU in Normal state).

The configuration is permanently locked until FCCU is reset.

3.9 Functional description

3.9.1 Definitions

In general, the following definitions are applicable for the fault management:

- HW recoverable fault: the fault indication is an edge-triggered and level-sensitive signal that remains asserted until the fault cause is deasserted. That is, if logical 0 on the fault signal indicates fault, then the status flags are valid as long as the fault line

stays at 0. The status is automatically cleared when the fault signal goes to 1. Typically the fault signal is latched external to the FCCU in the module where the fault occurred. The FCCU state transitions are consequently executed on the state changes of the input fault signal. No SW intervention in the FCCU is required to recover the fault condition.

- SW recoverable fault: the fault indication is a signal asserted without a defined time duration. The fault signal is latched in the FCCU. The fault recovery is executed following a SW recovery procedure (status/flag register clearing).

HW recoverable is an option to exclude the handling of error source/s by FCCU management SW, in case it is known that the fault is recoverable by itself when the fault condition gets corrected.

These type of chip resets reset the FCCU:
- Power-on
- Destructive
- Long functional, when initiated by the assertion of the chip reset pin (RESET_B)

These types of chip resets don't reset the FCCU:
- Long functional, when initiated by anything other than the assertion of the chip reset pin (RESET_B)
- Short functional reset

For more information on each type of reset, see the MC_RGM and reset chapters.

3.9.2 FSM description

FCCU functionality is depicted by the finite state machine (FSM) state diagram.

Basically four states are identified with the following meaning:

- Configuration: Used only to modify the configuration of FCCU from its default. A subset of the FCCU registers, dedicated to define the FCCU configuration (global configuration, reactions to fault, timeout, noncritical fault masking) can be accessed in write mode only in Configuration state.

  Configuration state is accessible only in Normal state and if the configuration is not locked. The permanent configuration lock can be disabled by a reset which also resets the FCCU., The transient lock register is unlocked by writing BCh into it. FCCU gets transiently locked again if an invalid key is written into FCCU_TRANS_LOCK register (that is, other than BCh). Key to lock FCCU permanently is FFh, written in the FCCU_PERMNT_LOCK register.

After the release of reset the state of the transient lock will be locked. The state of the permanent lock will be unlocked.

The locking feature only restricts the FSM movement into Configuration state. Once the user enters Configuration state and then tries to lock the configuration, the locking of configuration will be effective after FCCU moves to Normal state, it will not be effective in the current Configuration state.

The Configuration to Normal state transition can be executed by SW or automatically following a timeout condition of the watchdog. In case the timeout information and the SW request to mode change to Normal appears at the same time, watchdog timeout has the priority and hence the Configuration registers (those that are writable only in Configuration state) are reset to their default values. The movement to Normal is made.

The incoming faults, occurring during the configuration phase (Configuration state) are latched in order to process them when FCCU is moved into Normal state, according to the new configuration.

All pending faults that occur during Configuration state result in both of the following:

- Highest-priority state transition
- Interrupt generation (NMI or alarm IRQ)

If the state transition occurs, it gives the reset reaction corresponding to the worst case based on all the faults (pending or nonpending faults) that occurred during Configuration state.

- Normal: This is FCCU's operating state when no faults are occurring. It is also the default state on the reset exit. Following state transitions occur on one of the following events:
  - unmasked noncritical faults with the timeout disabled → FCCU moves to Fault state
  - unmasked noncritical faults with the timeout enabled → FCCU moves to Alarm state
  - masked noncritical faults → FCCU stays in Normal state
- Alarm: FCCU moves into Alarm state when an unmasked noncritical fault occurs and the timeout is enabled. The transition to Alarm state goes along with an interrupt alarm, if enabled. By definition, this fault may be recovered within a programmable

timeout period, before it generates a transition to Fault state. The timeout is reinitialized if FCCU enters Normal state. The timeout restarts following the recovery from Fault state.

- Fault: FCCU moves into Fault state when one of the following condition occurs:
  - timeout related to a noncritical fault when FCCU is in Alarm state
  - unmasked noncritical faults with the timeout disabled

The transition from Normal/Alarm to Fault state goes along with the generation of:

- NMI interrupt (optional)
- EOUT signaling (optional)
- SW option: Soft reaction (Short functional reset request pulse if configured)
- SW option: Hard reaction (Long functional reset request pulse if configured)
- Non Maskable Interrupt (NMI) is routed only to the Safety Core

After moving to Fault state, if there is either a previous pending fault or a new fault for which NMI is enabled, NMI generation takes place.

Multiple faults can occur at the same time. Consider the case where there is a fault for which Alarm state is enabled and IRQ is programmed along with a fault for which Alarm state is disabled. In that case, the FSM moves to Fault state, but with the generation of alarm IRQ for the fault which has IRQ enabled.

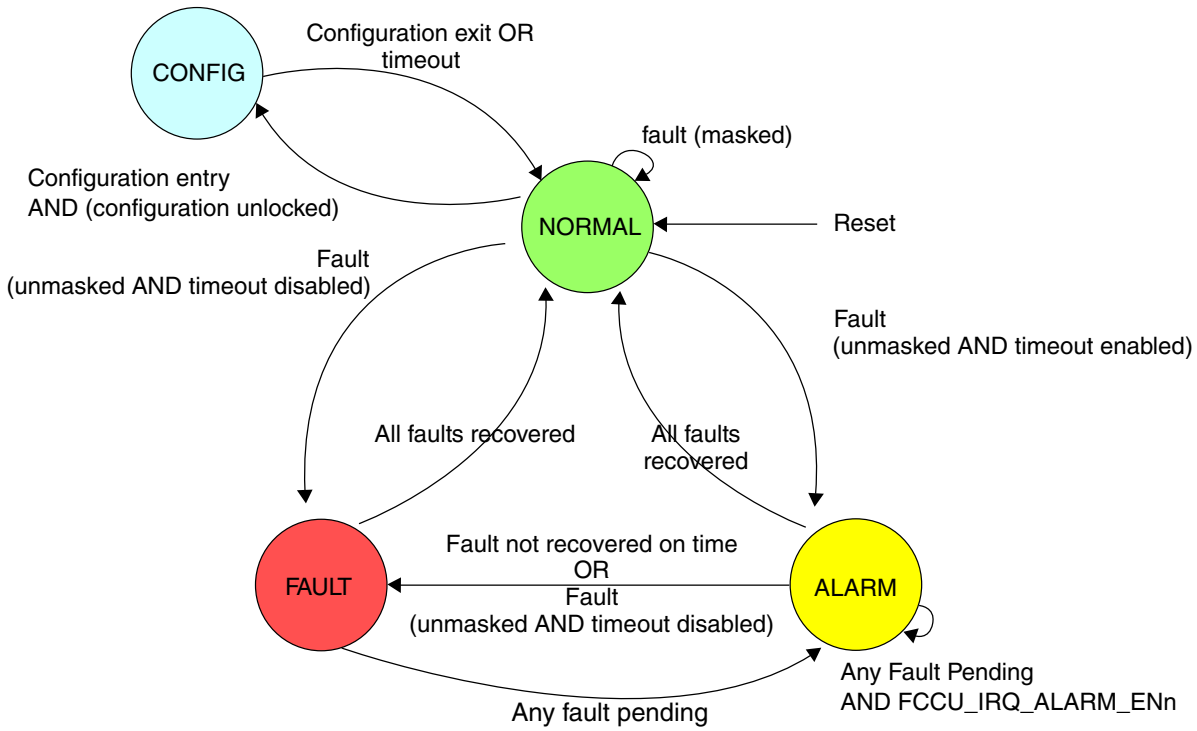


Figure 3-3. FCCU state diagram

3.9.3 Self-checking capabilities

The FCCU includes some features to support self-checking capabilities of the main FSM in running mode. The FSM unit is duplicated and its state (internal state and relevant outputs) is checked by 2 RCCx units (cycle accurate). The following checks (per CLKSAFE cycle) are provided by the RCCx units:

- all the FSM outputs and its internal state for the redundant FSM instances are checked to detect runtime faults on the FSM outputs and its internal state
- parity bits (computed at byte level) on the configuration registers are checked to detect run time faults on the configuration inputs of the FSM
- parity bits (computed at byte level) on the interface used to clear the FCCU_CF_Sx FCCU_UF_Sx and FCCU_RF_Sx status registers
- EOUT protocol state in dual-rail, time-switching and bi-stable mode
- common mode signals (handshake interface signals activated only in CONFIG state) congruence with the FSM state

In case of failure each RCCx unit provides the following reaction:

- an interrupt request to the INTC module.

The RCCx state is freeze in a SC freeze status register (FCCU_SCFS).

To guarantee the self checking capabilities through the FAULT interface, each critical fault source must be duplicated on a couple of critical fault inputs.

Two separate clock inputs (CLKSAFE0 and CLKSAFE1) are routed to the redundant modules (FSMx, RCCx, EOUTx, FAULT-if modules). An off-module monitor of the CLKSAFE clocks and an off-module reaction (functional reset to the MC) is required to cover a potential fault on the CLKSAFE clocks.

Self checking capabilities cover the HW reaction of the FCCU due to the assertion of an external fault. The IPS bus interface and the handshake modules are not covered by self-checking capabilities. Any operation executed by SW (configuration, fault recovery) must be cross-verified by congruence checks via SW (registers read following a write/ clear operation, internal FCCU state, ..).

A typical sequence with self checking failure and related reaction is given inFigure 3-4.

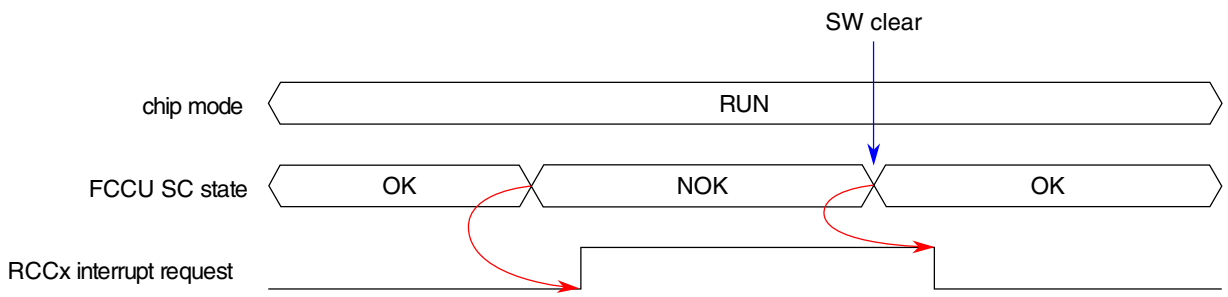


Figure 3-4. Self-checking reaction (interrupt)

For more information about how to enable or disable FCCU Self-checking, see Configuration (FCCU_CFG) and SC Freeze Status (FCCU_SCFS).

3.9.4 Fault priority scheme and nesting

The FAULT state has a higher priority than the ALARM state in case of concurrent fault events (noncritical) that occur in the NORMAL state.

The ALARM to FAULT state transition occurs if a noncritical fault (unmasked and with time-out disabled) is asserted in the ALARM state.

The ALARM to NORMAL state transition occurs only if all the noncritical faults (including the faults that have been collected after the entry in the ALARM state) have been cleared (SW or HW recovery); otherwise the FCCU will remain in the ALARM state.

The FAULT to NORMAL state transition occurs only if all the noncritical faults (including the faults that have been collected after the entry in the FAULT/ALARM state) have been cleared (SW or HW recovery); otherwise the FCCU will move to the ALARM state (if any noncritical fault is still pending and the time-out is not elapsed).

In general, no fault nesting is supported except for the noncritical faults that cause an ALARM to FAULT state transition. In this case the NCF timer is stopped until the FAULT state is recovered. If FCCU is in ALARM state and another fault comes, which has its alarm timeout enabled, then the alarm timer shall not reload and shall not start again.

3.9.5 Fault recovery

The following timing diagrams describe the main use cases of the FCCU in terms of fault events and related recovery.

A typical sequence related to a noncritical FAULT management (ALARM state), see Figure 3-5 and Figure 3-6, is as follows:

- Noncritical fault assertion
- FCCU state transition (automatic): NORMAL → ALARM
  - alarm interrupt request (if enabled)
  - time-out running
- system state: RUN
- alarm interrupt management: FAULT recovery (by SW)
  - FCCU state transition ALARM → NORMAL

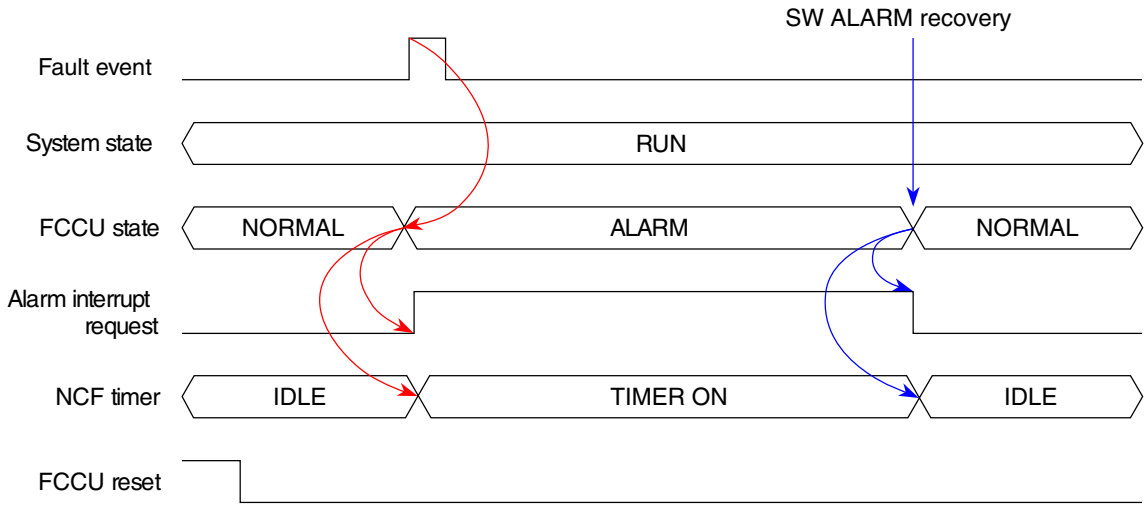


Figure 3-5. Noncritical FAULT (ALARM state) recovery (a)

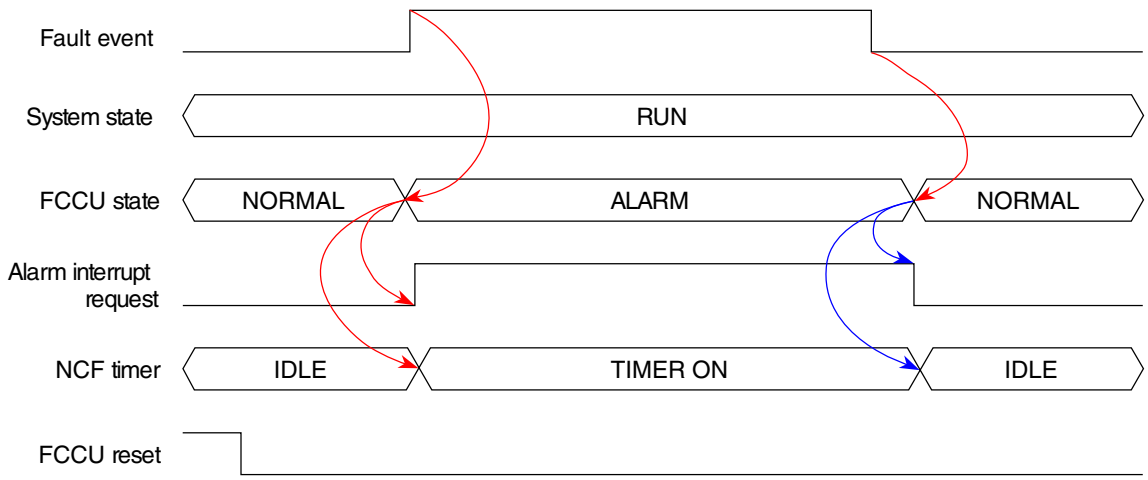


Figure 3-6. Noncritical FAULT (ALARM state) recovery (b)

A typical sequence related to a noncritical FAULT management (ALARM → FAULT state), see Figure 3-7, is as follows:

- Noncritical fault assertion
- FCCU state transition (automatic): NORMAL → ALARM
  - alarm interrupt request (if enabled)
  - time-out running
- FCCU state transition (following the time-out trigger): ALARM → FAULT
  - NMI assertion (if enabled)
- system state transition: RUN → SAFE
- NMI interrupt management (if enabled)

- FAULT recovery (by SW): FCCU state transition FAULT → NORMAL
- system state transition: SAFE → RUN

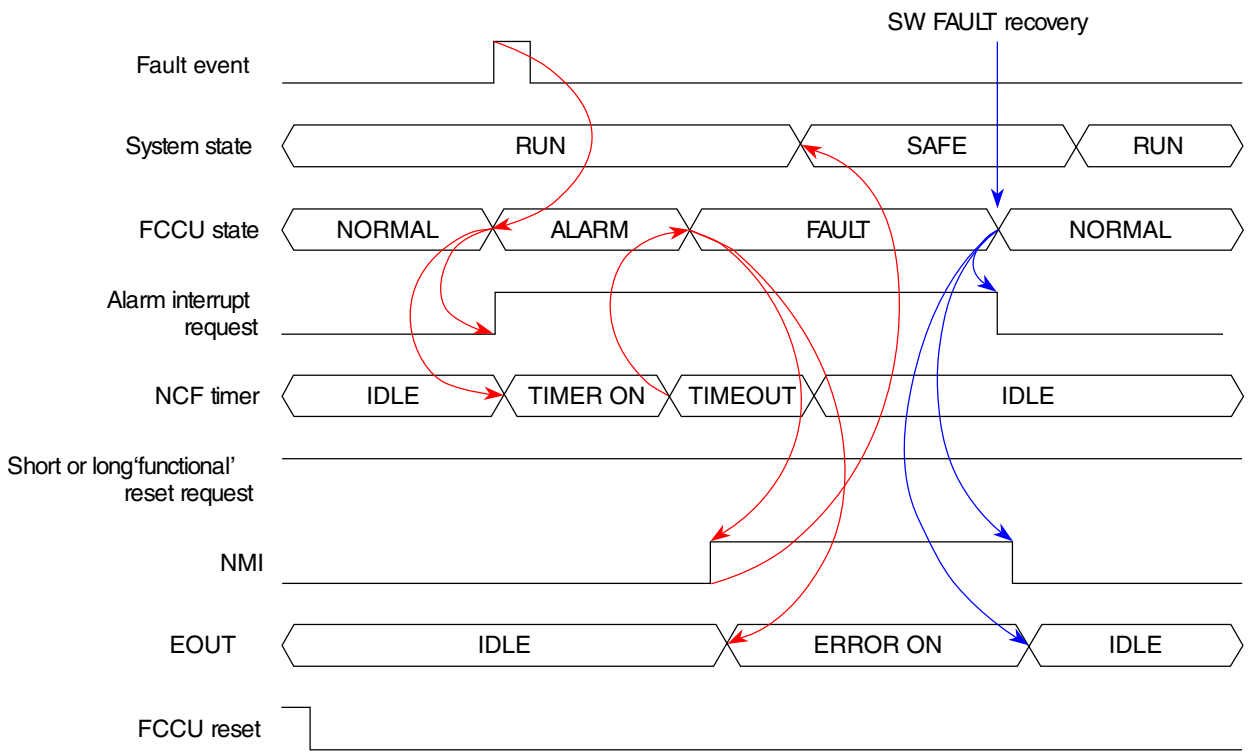


Figure 3-7. FAULT (ALARM → FAULT state) recovery

3.9.6 NMI/WKPU interface

NMIOUT is connected to all processor cores on the chip but is masked when any one of the following is true:

- An FCCU asynchronous reset is in progress.
- The chip is in RESET mode.
- The chip has automatically entered DRUN mode from RESET mode and software has not yet requested another mode change.

This figure shows the logical scheme.

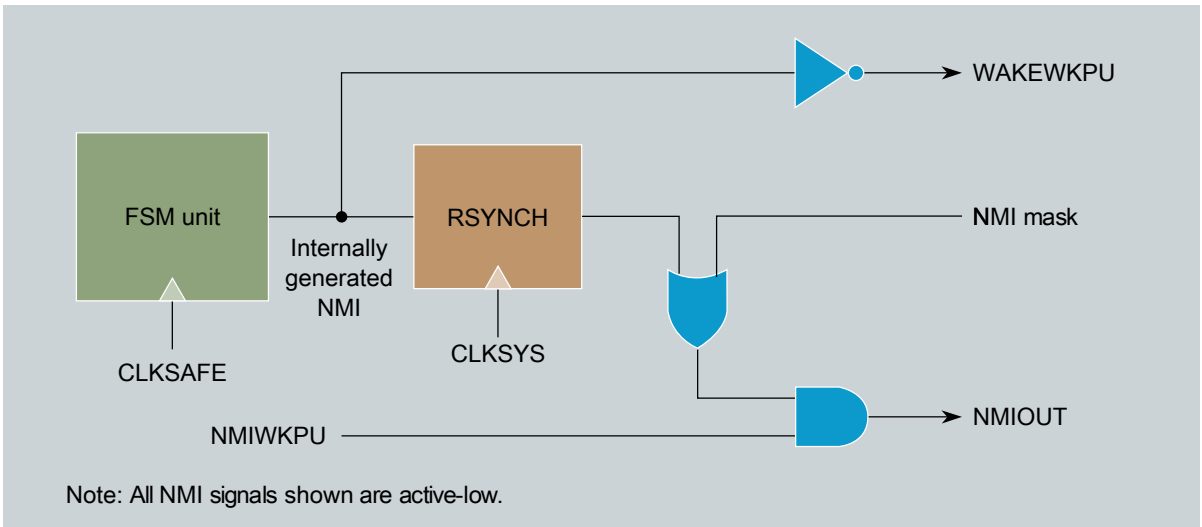


Figure 3-8. NMI/WKPU scheme

3.9.7 STCU interface

The STCU interface includes:

- A set of signals resulting from the self-checking procedure connected externally at the FCCU.

  The STCU fault signals are processed by the FCCU when the SoC is re-booted following the self-testing procedure. The STCU includes a status register that stores the self-testing results (flags).

- a mask that inhibits the EOUT dummy signaling until the STCU self-checking procedure has been completed.

During the self-testing procedure, depending on the STCU results, three cases are applicable:

1. STCU completes the self-testing procedure successfully. The SoC reboots and the FCCU is responsible for fault collection and indication.
2. STCU completes the self-testing procedure with low severity failures. The FCCU records the fault condition but does not, by default, trigger a reset cycle.
3. STCU completes the self-testing procedure with serious failures. The FCCU triggers a reset cycle in response to this fault input from the STCU. Because the fault condition survives reset, new reset cycles are subsequently triggered until the reset escalation feature takes control, freezing the chip in the reset state until a POR. This feature is optionally programmable using MC_RGM.

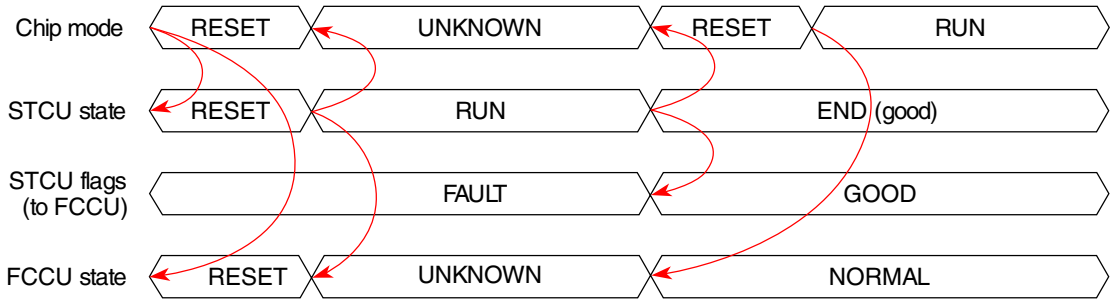


Figure 3-9. STCU-FCCU (case 1)

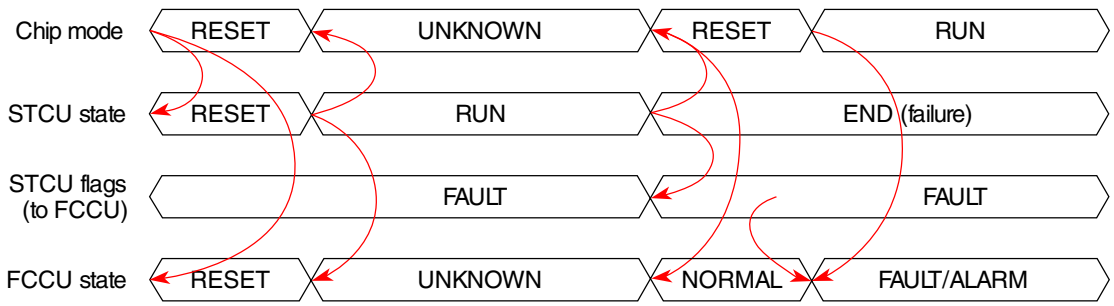


Figure 3-10. STCU-FCCU (case 2)

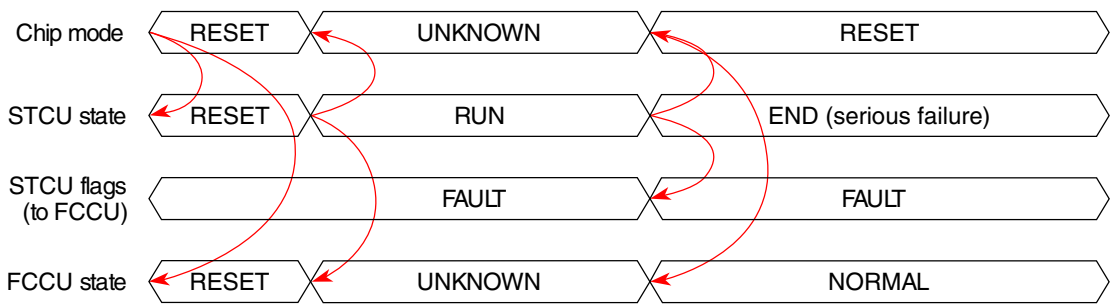


Figure 3-11. STCU-FCCU (case 3)

3.9.8 Nonvolatile memory interface

FCCU includes a nonvolatile memory interface (NVMCFG signals) that provides the values of certain bits in the Configuration (FCCU_CFG) register when certain events occur. See Configuration (FCCU_CFG). Whether any of these signals are actually connected to nonvolatile (flash) memory is chip-specific. Some or all of these signals may be driven by bits in a data record in nonvolatile (flash) memory or may be tied high (1) or low (0). See FCCU_CFG register bit value sources (N and C) by event.

The timing diagram related to the NVM interface is given in Figure 3-12.

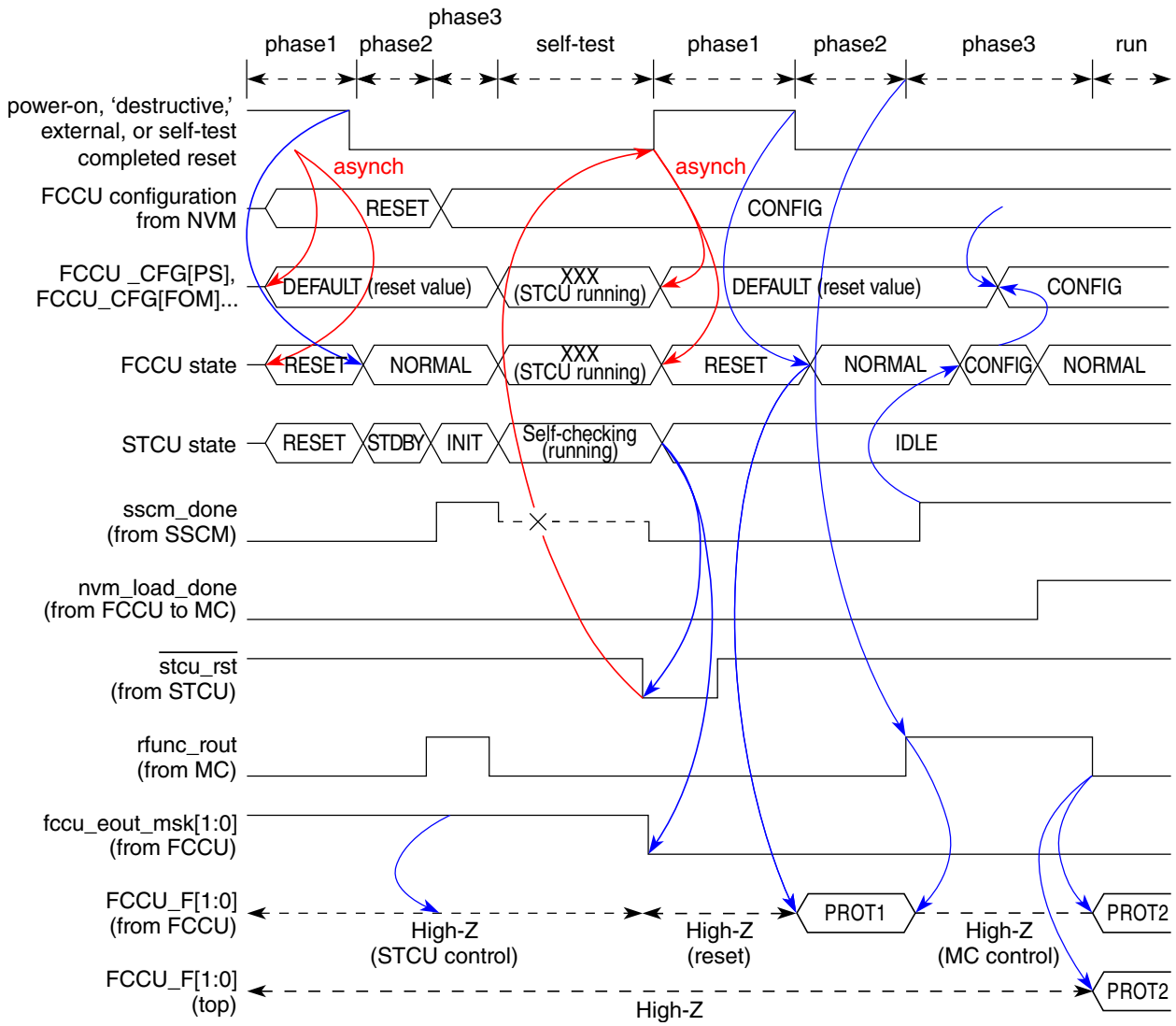


Figure 3-12. NVM interface

3.9.9 EOUT interface

3.9.9.1 Introduction

You use the EOUT[1:0] signals to indicate FCCU's condition to off-chip logic. For information on the availability and names of these FCCU signals on the boundary of this chip, see the chip-specific FCCU information.

3.9.9.2 The FCCU conditions

There are three FCCU conditions:

Condition	Description
Faulty	The fault-output (EOUT) timer is running (see How the fault-output (EOUT) timer works).
Nonfaulty	All of the following are true: • The fault-output (EOUT) timer is not running. • FCCU is in Alarm or Normal state.
Configuration	All of the following are true: • The fault-output (EOUT) timer is not running. • FCCU is in Configuration state.

3.9.9.3 How the fault-output (EOUT) timer works

FCCU starts the fault-output (EOUT) timer when all of the following are true:

- The fault-output (EOUT) signals are not programmed to be always low (FCCU_SET_CLEAR).
- The fault-output (EOUT) timer isn't already running.
- FCCU enters Fault state as the result of a fault.

When the fault-output (EOUT) timer is already running and a new fault occurs:

- When FCCU is in Configuration state: FCCU doesn't restart the fault-output (EOUT) timer.
- When FCCU is in Normal state or Alarm state:
  - And Alarm state is enabled for the fault (noncritical): FCCU enters (or remains in) Alarm state but doesn't restart the fault-output (EOUT) timer.
  - And Alarm state is disabled for the fault (noncritical): FCCU enters Fault state and restarts the fault-output (EOUT) timer.
- When FCCU is in Fault state: FCCU restarts the fault-output (EOUT) timer.

FCCU stops and reinitializes the fault-output (EOUT) timer when all of the following are true:

- If the EOUT signals are in Bistable fault-output mode (FOM), T_min has expired.
- All faults that caused FCCU to enter or remain in Fault state since FCCU started the fault-output (EOUT) timer have been cleared, causing FCCU to return to Normal state.

3.9.9.4 Prepare the EOUT signals to indicate FCCU's condition

Make sure all of the following are true:

- The EOUT signals are active (FCCU_SET_AFTER_RESET).
- The EOUT signals are controlled by FCCU's finite state machine (FCCU_SET_CLEAR).

3.9.9.5 More about the EOUT interface

Different fault-output modes (protocols) for the fault-output (EOUT) interface are supported (FCCU_CFG[FOM]):

- Dual-Rail
- Time-Switching
- Bistable
- Test

You can further configure the fault-output modes using the following attributes:

Attribute	Field	Setting used in the example diagrams and tables that follow
Switching mode	FCCU_CFG[SM]	Slow
Polarity selection	FCCU_CFG[PS]	For the faulty indication, EOUT1 is high, and EOUT0 is low.

The EOUT frequency is generated by dividing the CLKSAFE frequency by a fixed factor of 2^{18}.

With a CLKSAFE frequency of 16 MHz, this drives a signal of 61 Hz on EOUT.

The external monitor of the EOUT protocol should oversample the EOUT signals in order to synchronize periodically the external clock (used by the monitor) and CLKSAFE (which detects the edge transition of the EOUT protocol) in Dual-Rail or Time-Switching fault-output mode.

In case of a failure event or on software request for EOUT indication, the signal(s) are set to the faulty state for a minimum time T_min, even if software tries to release it before. If software configures the error pins to OK(1), and if a fault comes trying to drive the pin to NOK(0), then priority is given to the fault indication and the error signals indicate NOK, such as an incoming fault is not masked when software has set the error signal to high. During the T_min by a non-software fault, the FCCU FSM moves independently of this

signal state (low), and as soon as the timer expires, the pin behavior is dictated by the state in which the FSM finds itself in, and it is not possible to set the signals to OK by software moving FCCU to Configuration state, as long as this timer is running. No software intervention is needed to bring the signal from the low state.

Software can bring the pin back to OK state by clearing the faults and waiting for the T_min interval to expire, after which the FCCU automatically enters Normal state and the error signal indicates OK.

In case another failure event happens within T_min after a first one, the T_min counter is restarted.

NOTE

The transition from Fault to Configuration state is not possible but is shown to display the behavior in all four states—reset, normal, error, and configuration.

NOTE

FCCU remains in Normal state when a fault is disabled; therefore, *NA* (not applicable) appears in the following table.

NOTE

The following table is valid only when the EOUT signals are active (FCCU_SET_AFTER_RESET).

Table 3-5. FCCU error pin behavior on state transitions

Fault signaling state	Transition from Normal to Fault state			Normal+Configuration states			Transition from Normal to Alarm state		
	with Fault disabled | enabled						with Fault disabled | enabled		
	Internal error pin	Error pin enable	Error pad	Internal error pin	Error pin enable	Error pad	Internal error pin	Error pin enable	Error pad
Disabled	NA | OK	NA | 1	NA | OK	OK	1	OK	NA | OK	NA | 1	NA | OK
Enabled	NA | NOK	NA | 1	NA | NOK	OK	1	OK	NA | OK	NA | 1	NA | OK

NOTE

The first time FCCU is in Configuration state, the EOUT pads are in a high-impedance state; thereafter, their states reflect the programmed EOUT protocol.

3.9.9.6 Dual-rail protocol

Dual-rail encoding is an alternate method for encoding bits. In contrast with classical encoding, where each signal carries a single-bit value, dual-rail encoded circuits use two wires to carry each bit. The encoding scheme is given in Table 3-6 and the related timing diagram is given in Figure 3-13 and Figure 3-14.

Table 3-6. Dual-rail encoding

Logical state	Dual-rail encoding (output pins EOUT[1:0])	Note
non-faulty	10	toggling
non-faulty	01	
faulty	00	toggling
faulty	11	
reset	Hi impedence[1]	no toggling
configuration	same as NORMAL	toggling

1. Final value depends on the SoC setting at pad level.

As long as FCCU is in NORMAL or ALARM state, output will show "non-faulty" signal. Output pins EOUT[0:1] will toggle between 01 and 10 with a given frequency of 61 Hz.

In the RESET phase the output pins are set as "high impedance".

Note

Figure 3-13 and Figure 3-14 are formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

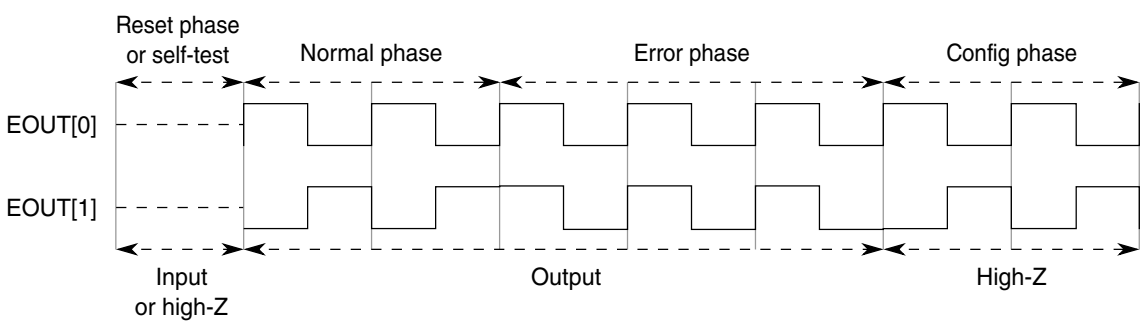


Figure 3-13. Dual-rail protocol (slow switching mode)

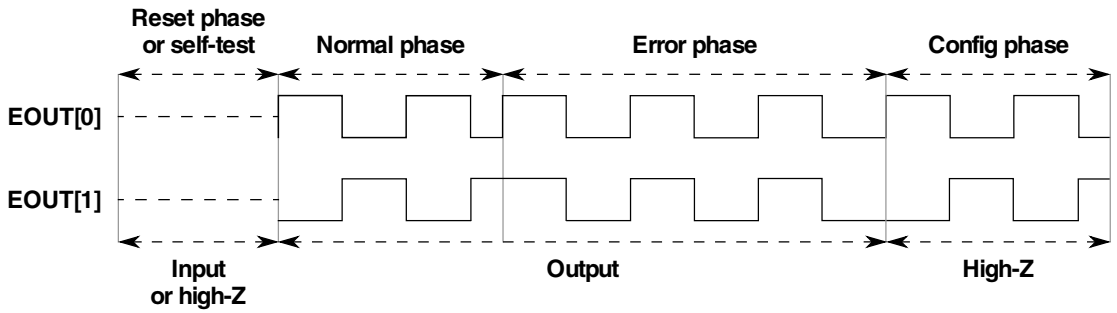


Figure 3-14. Dual-rail protocol (fast switching mode)

3.9.9.7 Time-switching protocol

The encoding scheme is given in Table 3-7 and the related timing diagram is given in Figure 3-15.

Table 3-7. Time-switching encoding

Logical state	Time-switching encoding (output pins EOUT[1:0])	Note
non-faulty	10	toggling
non-faulty	01	
faulty	10	no toggling
reset	high-Z[1]	no toggling
configuration	same as NORMAL	toggling

1. Final value depends on device specific settings at pad level.

As long as FCCU is in NORMAL or ALARM state, outputs will show "non-faulty" signal. Output pins #0, #1 will toggle between 01 and 10 with frequency of 61 Hz.

In the FAULT state, the output pin EOUT[0] is set as low.

In Time Switching mode the second output (EOUT[1]) is the inverted signal of first output (EOUT[0]).

In the RESET phase the output pins are set as "high impedance".[2]

Note

Figure 3-15 is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

2. Actual value depends on device specific settings at pad level.

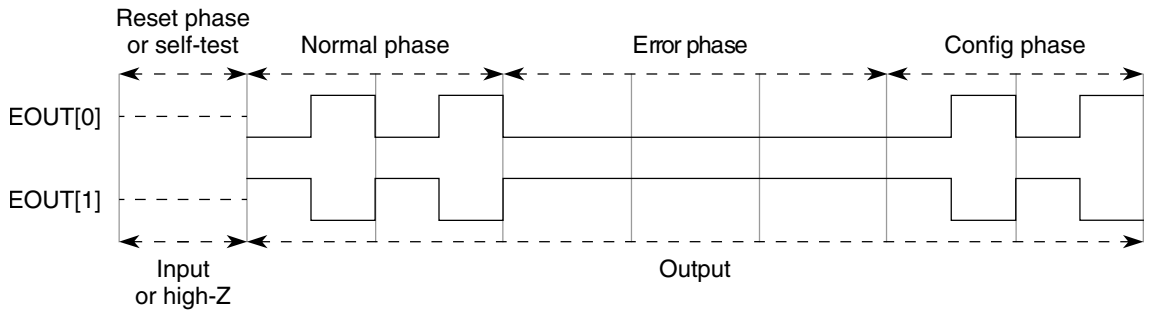


Figure 3-15. Time-switching protocol

3.9.9.8 Bi-stable protocol

The encoding scheme is given in Table 3-8 and the related timing diagram is given in Figure 3-16.

Table 3-8. Bi-stable encoding

Logical state	Bi-stable encoding (output pins EOUT[1:0])	Note
non-faulty	01	no toggling
faulty	10	no toggling
reset	high-Z[1]	no toggling
configuration	same as NORMAL	no toggling

1. Final value depends on device specific settings at pad level.

In the FAULT state, the faulty logical state is indicated. In NORMAL or ALARM state, "no-faulty" state is indicated. In Bi-stable mode the second output (EOUT[1]) is the inverted signal of first output (EOUT[0]).

In the RESET phase the output pins are set as "high impedance".[3]

Note

Figure 3-16 is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

3. Actual value depends on device specific settings at pad level.

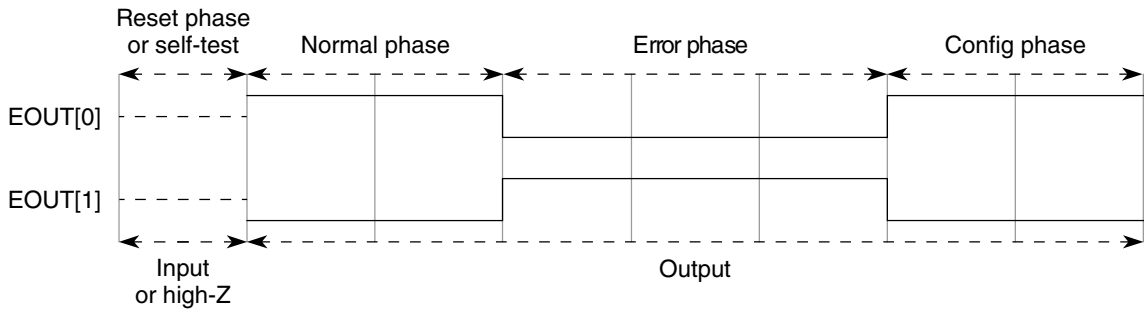


Figure 3-16. Bi-stable protocol

3.9.10 Error signal flow

See the chip-specific FCCU information for the diagram of the signal flow.

3.10 Register descriptions

The FCCU registers are listed in the table below. Any address offset not explicitly mentioned in this table is reserved.

The FCCU supports word (32-bit), half-word (16-bit), and byte (8-bit) read and write accesses.

Follow these register-access guidelines:

- Don't read from or write to any addresses that are not shown in the following table. Doing so may result in a transfer error.
- Don't write to any of the configuration registers unless FCCU is in Configuration state. Doing so results in a transfer error.
- Don't write to FCCU_TRANS_LOCK or FCCU_PERMNT_LOCK unless your code runs in Supervisor mode. Doing so results in a transfer error.

All the registers accessible in write mode only in CONFIG state are referred as configuration registers. These configuration registers return to the default value after configuration watchdog timer expires. These are the registers protected by FCCU_TRANS_LOCK and FCCU_PERMNT_LOCK registers. The configuration register setting has effect only when the FCCU state exits from the CONFIG state.

For each possible NCF failure source, a different reaction—including no reaction—is configurable through the use of NMI, IRQ, and long/short reset selection registers. It is not possible for a single event upset to switch off all reactions on failures as implementation is per fault source (but it will be possible to switch them all off by SW if intended). Failures themselves are not able to disable all reactions and indications.

The FCCU is not reset by short or long functional resets.

FCCU memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control (FCCU_CTRL)	32	R/W	0000_00C0h	3.10.1/464
4	Control Key (FCCU_CTRLK)	32	W	0000_0000h	3.10.2/466
8	Configuration (FCCU_CFG)	32	R/W	See section	3.10.3/467
1C	Noncritical Fault Configuration (FCCU_NCF_CFG0)	32	R/W	See section	3.10.4/471
20	Noncritical Fault Configuration (FCCU_NCF_CFG1)	32	R/W	See section	3.10.4/471
24	Noncritical Fault Configuration (FCCU_NCF_CFG2)	32	R/W	See section	3.10.4/471
4C	Noncritical Fault State Configuration (FCCU_NCFS_CFG0)	32	R/W	See section	3.10.5/472
50	Noncritical Fault State Configuration (FCCU_NCFS_CFG1)	32	R/W	See section	3.10.5/472
54	Noncritical Fault State Configuration (FCCU_NCFS_CFG2)	32	R/W	See section	3.10.5/472
58	Noncritical Fault State Configuration (FCCU_NCFS_CFG3)	32	R/W	See section	3.10.5/472
5C	Noncritical Fault State Configuration (FCCU_NCFS_CFG4)	32	R/W	See section	3.10.5/472
60	Noncritical Fault State Configuration (FCCU_NCFS_CFG5)	32	R/W	See section	3.10.5/472
80	Noncritical Fault Status (FCCU_NCF_S0)	32	R/W	0000_0000h	3.10.6/473
84	Noncritical Fault Status (FCCU_NCF_S1)	32	R/W	0000_0000h	3.10.6/473
88	Noncritical Fault Status (FCCU_NCF_S2)	32	R/W	0000_0000h	3.10.6/473
90	Noncritical Fault Key (FCCU_NCFK)	32	W	0000_0000h	3.10.7/474
94	Noncritical Fault Enable (FCCU_NCF_E0)	32	R/W	See section	3.10.8/475
98	Noncritical Fault Enable (FCCU_NCF_E1)	32	R/W	See section	3.10.8/475
9C	Noncritical Fault Enable (FCCU_NCF_E2)	32	R/W	See section	3.10.8/475
A4	Noncritical Fault Timeout Enable (FCCU_NCF_TOE0)	32	R/W	See section	3.10.9/476
A8	Noncritical Fault Timeout Enable (FCCU_NCF_TOE1)	32	R/W	See section	3.10.9/476
AC	Noncritical Fault Timeout Enable (FCCU_NCF_TOE2)	32	R/W	See section	3.10.9/476
B4	Noncritical Fault Timeout (FCCU_NCF_TO)	32	R/W	See section	3.10.10/477
B8	Configuration-State Timer Interval (FCCU_CFG_TO)	32	R/W	See section	3.10.11/478
BC	IO Control (FCCU_EINOUT)	32	R/W	See section	3.10.12/479
C0	Status (FCCU_STAT)	32	R	0000_0010h	3.10.13/481
C4	NA Freeze Status (FCCU_N2AF_STATUS)	32	R	0000_0000h	3.10.14/483
C8	AF Freeze Status (FCCU_A2FF_STATUS)	32	R	0000_0000h	3.10.15/484
CC	NF Freeze Status (FCCU_N2FF_STATUS)	32	R	0000_0000h	3.10.16/485
D0	FA Freeze Status (FCCU_F2A_STATUS)	32	R	0000_0000h	3.10.17/486
D4	SC Freeze Status (FCCU_SCFS)	32	R/W	0000_0000h	3.10.18/487

Table continues on the next page...

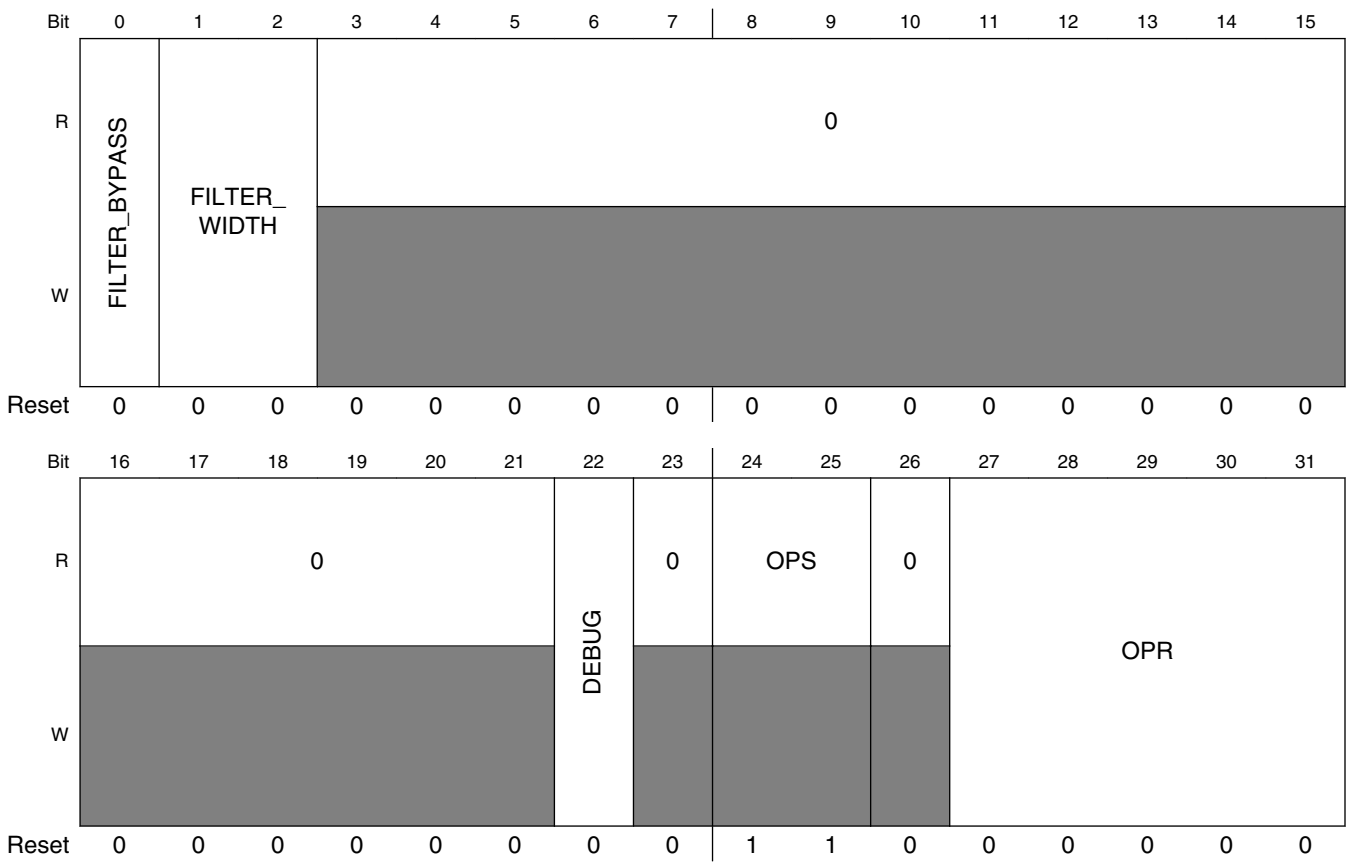
FCCU memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
DC	Noncritical Fault Fake (FCCU_NCFF)	32	R/W	0000_0000h	3.10.19/489
E0	IRQ Status (FCCU_IRQ_STAT)	32	R/W	0000_0000h	3.10.20/489
E4	IRQ Enable (FCCU_IRQ_EN)	32	R/W	0000_0000h	3.10.21/491
E8	XTMR (FCCU_XTMR)	32	R	0000_0000h	3.10.22/492
EC	Mode Controller Status (FCCU_MCS)	32	R	See section	3.10.23/493
F0	Transient Configuration Lock (FCCU_TRANS_LOCK)	32	W	0000_0000h	3.10.24/495
F4	Permanent Configuration Lock (FCCU_PERMNT_LOCK)	32	W	0000_0000h	3.10.25/496
F8	Delta T (FCCU_DELTA_T)	32	R/W	0000_0000h	3.10.26/496
FC	IRQ Alarm Enable (FCCU_IRQ_ALARM_EN0)	32	R/W	0000_0000h	3.10.27/497
100	IRQ Alarm Enable (FCCU_IRQ_ALARM_EN1)	32	R/W	0000_0000h	3.10.27/497
104	IRQ Alarm Enable (FCCU_IRQ_ALARM_EN2)	32	R/W	0000_0000h	3.10.27/497
10C	NMI Enable (FCCU_NMI_EN0)	32	R/W	0000_0000h	3.10.28/498
110	NMI Enable (FCCU_NMI_EN1)	32	R/W	0000_0000h	3.10.28/498
114	NMI Enable (FCCU_NMI_EN2)	32	R/W	0000_0000h	3.10.28/498
11C	Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN0)	32	R/W	0000_0000h	3.10.29/499
120	Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN1)	32	R/W	0000_0000h	3.10.29/499
124	Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN2)	32	R/W	0000_0000h	3.10.29/499

3.10.1 Control (FCCU_CTRL)

Initiates and indicates the status of operations, indicates that FCCU successfully loaded configuration data from the NVMCFG signals, controls glitch-filter settings, and enables Debug mode.

Address: 0h base + 0h offset = 0h



FCCU_CTRL field descriptions

Field	Description
0 FILTER_ BYPASS	Filter bypass 0 glitch filter not bypassed 1 glitch filter bypassed
1–2 FILTER_WIDTH	Filter width 00 filters glitches up to 50 µs 01 filters glitches up to 75 µs 10 filters glitches up to 100 µs 11 filters glitches up to 100 µs

Table continues on the next page...

FCCU_CTRL field descriptions (continued)

Field	Description
3–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 DEBUG	Debug Mode Enable Specifies whether Debug mode is enabled. If so, FCCU enters Debug mode when the Debug signal is asserted. When FCCU enters Debug mode, it halts operation and remains in the state it was in before it entered Debug mode. **NOTE:** FOSU doesn't halt when FCCU enters Debug mode. Therefore, FOSU can still cause a reset if a fault occurs while FCCU is in Debug mode. 0 Disabled 1 Enabled
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 OPS	Operation status. This bit can be read and cleared (via OP15 operation) by the software. 00 Idle 01 In progress 10 Aborted 11 Successful
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 OPR	Operation Run Initiates operations that perform actions such as putting FCCU in Configuration state or reading a timer. For information on how to run operations, see Run operations. FCCU ignores any write to this field while the operation status (OPS) is In Progress. After completion of an operation, FCCU sets this field to OP0. The following events result in an operation status (OPS) of Aborted: • Writing to a Noncritical Fault Status (FCCU_NCF_S*n*) register (which automatically initiates the OP12 operation) without first successfully unlocking the register (NCFK) • Initiating an OP1 operation when FCCU is not in Normal state or the configuration is locked • Initiating an OP1, OP2, or OP31 operation without first unlocking the operation (CTRLK) 00000 OP0—No operation 00001 OP1—Applies only when the configuration is unlocked, when FCCU is in Normal state, and immediately after you unlock the operation (CTRLK). Put FCCU in Configuration state. 00010 OP2—Applies only immediately after you unlock the operation (CTRLK). Put FCCU in Normal state. 00011 OP3—Capture FCCU's state and the states of the EOUT signals in the Status (FCCU_STAT) register. 00100 OP4—Read the FCCU frozen status flags (NA Freeze Status (FCCU_N2AF_STATUS)). 00101 OP5—Read the FCCU frozen status flags (AF Freeze Status (FCCU_A2FF_STATUS)). 00110 OP6—Read the FCCU frozen status flags (NF Freeze Status (FCCU_N2FF_STATUS)). 00111 OP7—Read the FCCU frozen status flags (FA Freeze Status (FCCU_F2A_STATUS)). 01000 OP8—Read the FCCU frozen status flags (SC Freeze Status (FCCU_SCFS)). 01001 Reserved

Table continues on the next page...

FCCU_CTRL field descriptions (continued)

Field	Description
	01010 OP10—Read the Noncritical Fault Status (FCCU_NCF_S*n*) register. 01011 Reserved 01100 OP12—Do not initiate this operation; it is automatically initiated by FCCU. A Noncritical Fault Status (FCCU_NCF_S*n*) register status clear operation is in progress. 01101 OP13—Clear the freeze status registers (see the freeze registers). 01110 OP14—Do not initiate this operation; it is automatically initiated by FCCU. A Configuration-state timeout is in progress. For more information, see Configuration registers. 01111 OP15—Set the operation status (OPS) to Idle. 10000 Reserved 10001 OP17—Read the Alarm-state timer (XTMR). 10010 Reserved 10011 OP19—Read the Configuration-state timer (XTMR). 10100 Reserved 10101–11110 Forbidden. Writing any of these values returns an operation status (OPS) of Aborted with no side effect. 11111 OP31—Applies only immediately after you unlock the operation (CTRLK). Load configuration data from the NVMCFG signals to certain fields in the Configuration (FCCU_CFG) register, but without resetting FCCU (for testing and debugging). See FCCU_CFG register bit value sources (N and C) by event. The operation status (OPS) for this operation is always Successful.

3.10.2 Control Key (FCCU_CTRLK)

Writable only with a 32-bit write. Unlocks locked operations so you can initiate them (OPR). For information on how to unlock locked operations before you initiate them, see Run an operation.

NOTE

Reading from this register always returns the value 0000_0000h.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	CTRLK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_CTRLK field descriptions

Field	Description
0–31 CTRLK	Locked-Operation Control Key

FCCU_CTRLK field descriptions (continued)

Field	Description
	Unlocks locked operations so you can initiate them (OPR). For information on how to unlock locked operations before you initiate them, see Run an operation. **NOTE:** You must initiate an operation in the instruction that immediately follows the instruction that unlocks it; otherwise, the operation is again locked. 9137_56AFh: Unlock OP1. 825A_132Bh: Unlock OP2. 29AF_8752h: Unlock OP31. Any other value: Do nothing.

3.10.3 Configuration (FCCU_CFG)

Writable only when FCCU is in Configuration state. Specifies the global configuration for FCCU.

The values of the bits in this register are affected by writes as well as any of the following events:

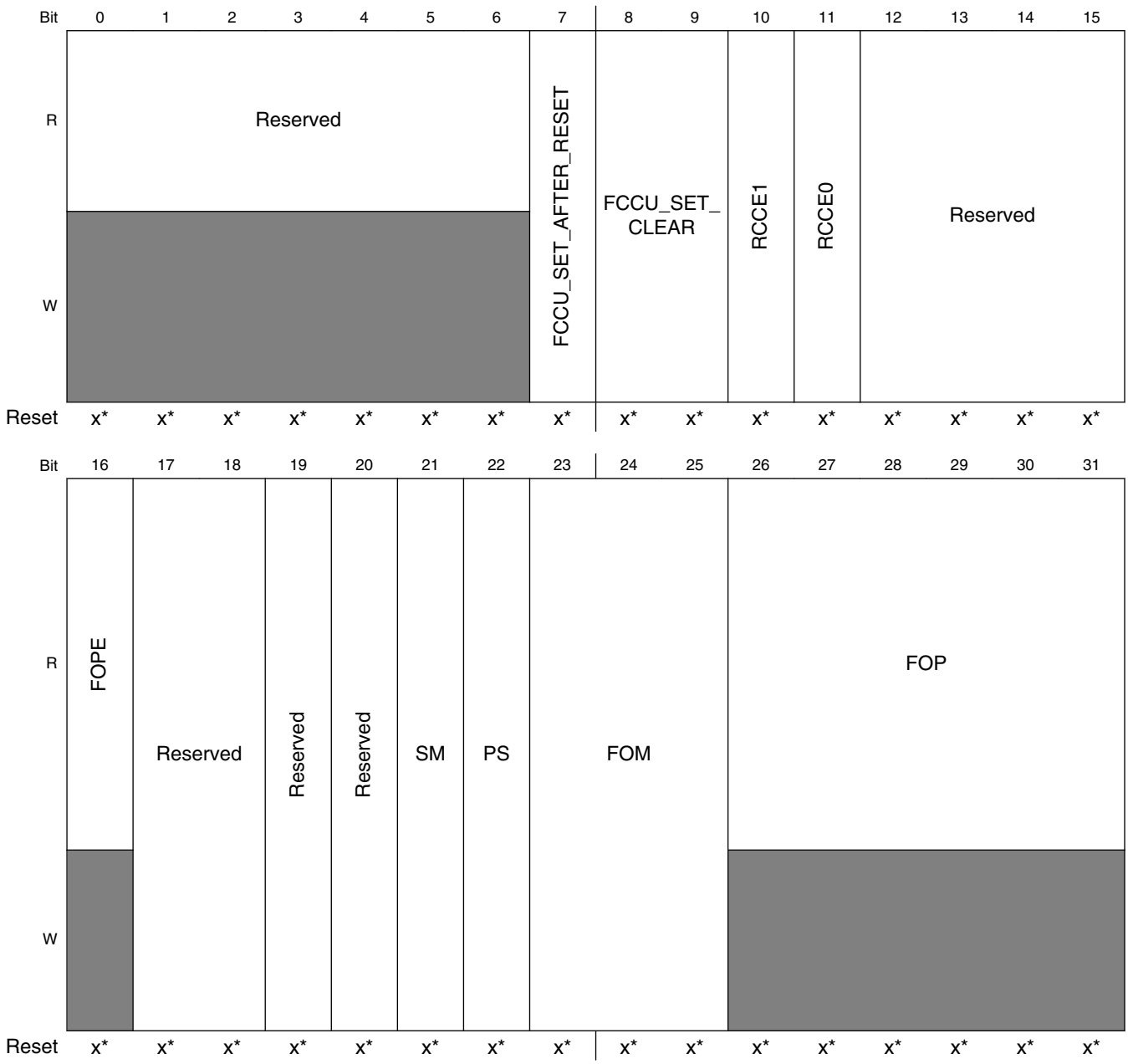
- The chip resets FCCU.
- FCCU automatically leaves Configuration state and enters Normal state because the configuration-timer interval (TO) expires. This is called a Configuration-state timeout.
- You run an OP31 operation.

For information on the sources of bit values for these events, see FCCU_CFG register bit value sources (N and C) by event.

NOTE

If you specify a new value for any of the fields in this register that affect the EOUT signals while the fault-output (EOUT) timer is running (FCCU is indicating a fault on the EOUT signals), FCCU doesn't use the new settings you specified until after the fault-output (EOUT) timer expires (FCCU stops indicating a fault on the EOUT signals).

Address: 0h base + 8h offset = 8h



* Notes:

- For information on this register's reset value, see FCCU_CFG register bit value sources (N and C) by event.x = Undefined at reset.

FCCU_CFG field descriptions

Field	Description
0–6 Reserved	This field is reserved.

Table continues on the next page...

FCCU_CFG field descriptions (continued)

Field	Description
7 FCCU_SET_ AFTER_RESET	Fault-Output (EOUT) Activate For fault-output (EOUT) signaling, controls whether the EOUT signals are active. 0 Inactive (the EOUT signals are in a high-impedance state) 1 Active (the EOUT signals indicate FCCU's condition)
8–9 FCCU_SET_ CLEAR	Fault-Output (EOUT) Control Applies only when the EOUT signals are active (FCCU_SET_AFTER_RESET). Controls whether the fault-output (EOUT) signals are controlled by FCCU's finite state machine (FSM). 00 Controlled by the FSM 01 Always low 10 Controlled by the FSM 11 High until a fault occurs on a channel, regardless of whether that fault is disabled; thereafter, controlled by the FSM. Note: FCCU ignores an attempt to write this value if the fault-output (EOUT) timer is already running.
10 RCCE1	Redundancy Control Checker 1 Enable For the self-checking feature, controls whether FCCU's redundancy control checker 1 (RCC1) is enabled. **NOTE:** • Always enable any RCCs in a separate register access after activating the EOUT signals (FCCU_SET_AFTER_RESET). • If only one RCC is enabled and a self-check fails, both RCCs assert an interrupt. 0 Disabled 1 Enabled
11 RCCE0	Redundancy Control Checker 0 Enable For the self-checking feature, controls whether FCCU's redundancy control checker 0 (RCC0) is enabled. **NOTE:** • Always enable any RCCs in a separate register access after activating the EOUT signals (FCCU_SET_AFTER_RESET). • If only one RCC is enabled and a self-check fails, both RCCs assert an interrupt. 0 Disabled 1 Enabled
12–15 Reserved	This field is reserved. Always write the reset value to this field.
16 FOPE	Fault-Output (EOUT) Prescaler Extension For fault-output (EOUT) signaling, specifies the most significant bit of the derived prescaler that controls the signaling frequency. For more information, see the description of the FOP field.
17–18 Reserved	This field is reserved. Always write the reset value to this field.
19 Reserved	This field is reserved. Always write the reset value to this field.
20 Reserved	This field is reserved. Always write the reset value to this field.
21 SM	Fault-Output (EOUT) Switching Mode For fault-output (EOUT) signaling, controls how quickly the signals switch between faulty and nonfaulty indication. Applies only to Dual-Rail and Time-Switching fault-output modes.

Table continues on the next page...

FCCU_CFG field descriptions (continued)

Field	Description
	0 Slow: No EOUT frequency violation during the FCCU state transition (Normal to Fault or vice versa, and Configuration to Normal). The indication transition occurs after a maximum delay equal to the duration of the semiperiod of the EOUT frequency. 1 Fast: The indication transition (Normal to Fault or vice versa, and Configuration to Normal) occurs immediately. A pulse with the minimum duration corresponding to the CLKSAFE period can occur in this mode. It implies a frequency violation of the EOUT protocol.
22 PS	Fault-Output (EOUT) Polarity Selection For fault-output (EOUT) signaling, controls the polarity of the signals for fault-output-mode indications that hold the signals low or high (versus toggling them or placing them in a high-impedance state). Applies only to Time-Switching fault-output mode (for faulty and configuration indications) and Bistable fault-output mode (for all indications). 0 For the faulty indication, EOUT1 is high, and EOUT0 is low. 1 For the faulty indication, EOUT1 is low, and EOUT0 is high.
23–25 FOM	Fault-Output (EOUT) Mode For fault-output (EOUT) signaling, controls the protocol of the signaling. **NOTE:** In the test modes, a simple double-stage resynchronization stage is used to resynchronize the EOUT input/outputs with CLKSAFE. 000 Dual-Rail (default state) [EOUT[1:0]= outputs] 001 Time-Switching [EOUT[1:0]= output to be used] 010 Bistable 011 Reserved 100 Reserved 101 Test 0 (controlled by the FCCU_EINOUT register; EOUT1 is an output; EOUT0 is an input) 110 Test 1 (controlled by the FCCU_EINOUT register; EOUT1 and EOUT0 are both outputs) 111 Test 2 (controlled by the FCCU_EINOUT register; EOUT1 is an input; EOUT0 is an output)
26–31 FOP	Fault-Output (EOUT) Prescaler For fault-output (EOUT) signaling, specifies the least significant bits of the derived prescaler that controls the signaling frequency. The derived prescaler is the concatenation of FOPE and FOP and is represented by *{FOPE, FOP}*. The following equation defines the signaling frequency in terms of the CLKSAFE frequency ($CLKSAFE_{freq}$) and the derived prescaler: $EOUT_{freq} = CLKSAFE_{freq} / ((\{FOPE, FOP\} + 1) \times 2 \times 2048)$ The following example values and meanings are for the derived (7-bit) prescaler, not FOP alone. The meanings show the derived prescaler followed by the corresponding value by which the input clock frequency is then divided. 0000000: 0; divide by 2 × 2048 (4096) 0000001: 1; divide by 4 × 2048 (8192) 0000010: 2; divide by 6 × 2048 (12,288) … 0001000: 8; divide by 18 × 2048 (36,864) … 1111101: 125; divide by 252 × 2048 (516,096)

Table continues on the next page...

FCCU_CFG field descriptions (continued)

Field	Description
	1111110: 126; divide by 254 × 2048 (520,192) 1111111: 127; divide by 256 × 2048 (524,288) **NOTE:** Writing to FCCU_CFG sets FOP to 0.

3.10.4 Noncritical Fault Configuration (FCCU_NCF_CFG*n*)

Contains the configuration of noncritical faults in terms of fault-recovery management. The configuration depends on the type of signaling of a fault event. Hardware-recoverable faults should be configured only if a previous latching stage captures and holds the physical fault; otherwise, the fault can be lost. All other faults should be configured as software faults.

NOTE

These registers are writable only when FCCU is in Configuration state.

Noncritical Fault Configuration (FCCU_NCF_CFG*n*) register (value of n)	Offset	NCFCx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	1Ch	31	0
1	20h	63	32

Address: 0h base + 1Ch offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCFCx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_CFG*n* field descriptions

Field	Description
0–31 NCFCx	Noncritical Fault Configuration. Defines the fault recovery mode. Hardware-recoverable faults are self-recovered (status flag clearing and related) if the root cause (input fault) has been removed. Software-recoverable faults are recovered (status flag clearing) by software clearing of the related status flag. For the mapping of the NCFCx fields among registers, see the earlier table in this register description.

FCCU_NCF_CFG*n* field descriptions (continued)

Field	Description
	0 Hardware-recoverable fault 1 Software-recoverable fault

3.10.5 Noncritical Fault State Configuration (FCCU_NCFS_CFG*n*)

The FCCU_NCFS_CFGx registers contain the configuration of each noncritical fault in terms of fault reaction (short or long functional reset request pulse) when it is the root cause for the FAULT state transition.

NOTE

These registers are writable only when the FCCU is in the CONFIG state.

Table 3-9 shows FCCU configuration register channels.

Table 3-9. FCCU NCFS configuration register channels

Address offset	Register name	Channel range (x)
4Ch	FCCU_NCFS_CFG0	NCFSCx[15:0]
50h	FCCU_NCFS_CFG1	NCFSCx[31:16]
54h	FCCU_NCFS_CFG2	NCFSCx[47:32]
58h	FCCU_NCFS_CFG3	NCFSCx[63:48]
5Ch	FCCU_NCFS_CFG4	NCFSCx[79:64]
60h	FCCU_NCFS_CFG5	NCFSCx[95:80]

Address: 0h base + 4Ch offset + (4d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCFSCx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCFS_CFG*n* field descriptions

Field	Description
0–31 NCFSCx	Noncritical fault state configuration See Table 3-9 for register offset to channel number relationship. 00 No reset reaction 01 Short functional reset request pulse (FAULT state reaction) 10 Long functional reset request pulse (FAULT state reaction) 11 No reset reaction

3.10.6 Noncritical Fault Status (FCCU_NCF_S*n*)

The FCCU_NCF_Sx register contains the latched fault indication collected from the noncritical fault sources. Faults are latched also in the CONFIG state and independently from the enabling or reactions programmed for the NCF.

No reactions are executed until the FCCU moves in the NORMAL state.

FCCU reacts and moves from the NORMAL state into the ALARM state only if the respective enable bit for a fault is set in the FCCU_NCF_Ex register and the respective enable bit for the timeout is set in the FCCU_TOEx register.

FCCU reacts and moves from the NORMAL or ALARM state into the FAULT state if the respective enable bit for a fault is set in the FCCU_NCF_Ex register and the respective enable bit for the timeout is disabled in the FCCU_TOEx register.

FCCU reacts and moves from the ALARM state into the FAULT state if the timeout (FCCU_TO register) is elapsed before recovering from the fault.

The timeout is stopped only when the FCCU returns in the NORMAL state.

The FCCU_NCF_Sx register is encoded respectively into the N2FF_STATUS or A2FF_STATUS register to freeze the entry condition in the FAULT state. The status bits of the FCCU_NCF_Sx register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the proper key into the FCCU_NCFK register.
2. Clear the status (flag) bit NCFSx => the opcode OP12 is automatically set into the FCCU_CTRL.OPR field.
3. Wait for the completion of the operation (FCCU_CTRL.OPS field).
4. Read the FCCU_NCF_Sx register in order to verify the effective deletion and in case of failure to repeat the sequence.

As result of the above sequence, in addition the FAULT interface provides support to clear the external fake FAULT root .

NOTE

There should not be any other operation in between the above steps.

The FCCU moves from the FAULT or ALARM state into the NORMAL state if all the source faults that caused the transition into the FAULT state have been removed (HW recoverable fault) or cleared via SW (SW recoverable fault). In case of nested faults that are not all recovered, the FCCU will remain in the FAULT or ALARM state.

The SW application executes the FCCU_NCF_Sx read operation by the following sequence:

1. Set the OP10 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU_NCF_Sx register.

In case of re-configuration of the FCCU (CONFIG state), before to return in NORMAL state the pending status bits into the FCCU_NCF_Sx must be cleared in order to avoid a false transition in ALARM/FAULT state.

The following faults are ignored:

- to write a wrong key into the FCCU_NCFK register
- to attempt to clear a HW recoverable fault

Noncritical Fault Status (FCCU_NCF_S*n*) register (value of n)	Offset	NCFSx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	80h	31	0
1	84h	63	32
2	88h	95	64

Address: 0h base + 80h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NCFSx																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NCF_S*n* field descriptions

Field	Description
0–31 NCFSx	Noncritical fault status. The status bits related to the noncritical fault configured as HW recoverable faults are read-only and the flag is self cleared when the fault source is removed. For the mapping of the NCFSx fields among registers, see the earlier table in this register description. 0 No "noncritical" fault latched 1 "Noncritical" fault latched

3.10.7 Noncritical Fault Key (FCCU_NCFK)

The FCCU_NCFK register implements the key access to clear the status flags of the FCCU_NCF_Sx register.

The status bits of the FCCU_NCF_Sx register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the key into the FCCU_NCFK register.
2. Clear the status (flag) bit NCFSx .

The key must be written for each FCCU_NCF_Sx clear operation.

The FCCU_NCFK register is not readable; a 0000_0000h value is always returned in case of read operation.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																															
W	NCFK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NCFK field descriptions

Field	Description
0–31 NCFK	Noncritical fault key = AB34_98FEh

3.10.8 Noncritical Fault Enable (FCCU_NCF_E*n*)

The FCCU_NCF_En registers enable the fault sources to allow a transition from the NORMAL into the FAULT or ALARM state. In case of fault masking, the respective status bit into the FCCU_NCF_Sn register is set (for debugging purposes), only the reaction is masked.

NOTE

These registers are writable only when the FCCU is in the CONFIG state.

NOTE

Any enabled fault should be programmed to result in a defined action. For example, set up an ALARM IRQ action by programming the IRQ Alarm Enable Register (FCCU_IRQ_ALARM_ENn).

Noncritical Fault Enable (FCCU_NCF_E*n*) register (value of n)	Offset	NCFEx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	94h	31	0
1	98h	63	32
2	9Ch	95	64

Address: 0h base + 94h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCFEx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_E*n* field descriptions

Field	Description
0–31 NCFEx	Noncritical fault enable For the mapping of the NCFEx fields among registers, see the earlier table in this register description. 0 No actions following the respective noncritical fault assertion 1 FCCU moves to ALARM or FAULT state

3.10.9 Noncritical Fault Timeout Enable (FCCU_NCF_TOE*n*)

The FCCU_NCF_TOEx registers enable a transition from the NORMAL state into the ALARM state if the respective noncritical fault is enabled (NCFEx and NCFTOEx are set). In case the respective timeout is disabled (NCFTOEx is cleared) and the noncritical fault is enabled (NCFEx is set) the FCCU moves into the FAULT state if the related noncritical fault is asserted. The timer (preset with the timeout value defined by FCCU_TO register) is started when the FCCU moves into the ALARM state. If the fault is not recovered within the timeout the FCCU moves from the ALARM state to the FAULT state.

NOTE

These registers are writable only when the FCCU is in the CONFIG state.

Noncritical Fault Timeout Enable (FCCU_NCF_TOE*n*) register (value of n)	Offset	NCFTOEx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	A4h	31	0
1	A8h	63	32
2	ACh	95	64

Address: 0h base + A4h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCFTOEx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_TOE*n* field descriptions

Field	Description
0–31 NCFTOEx	Fault timeout enable For the mapping of the NCFTOEx fields among registers, see the earlier table in this register description. 0 FCCU moves into the FAULT state if the respective fault is enabled 1 FCCU moves into the ALARM state if the respective fault is enabled

3.10.10 Noncritical Fault Timeout (FCCU_NCF_TO)

Defines the preset value of the timer for the recovery of enabled noncritical faults. Once FCCU enters Alarm state, following the assertion of a noncritical fault that is enabled (NCFEa and NCFTOEa are set), the timer starts the countdown.

If the fault is not recovered within the timeout period, the FCCU moves from the Alarm state to the Fault state. The alarm timeout value should be programmed to be less than FOSU_COUNT, or destructive resets may be generated by FOSU timeouts.

NOTE

This register is writable only when FCCU is in Configuration state.

Address: 0h base + B4h offset = B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TO																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_TO field descriptions

Field	Description
0–31 TO	Noncritical Fault Timeout Timeout = (TO) × $T_{RC16MHz}$

3.10.11 Configuration-State Timer Interval (FCCU_CFG_TO)

Writable only when FCCU is in Normal, Alarm, or Fault state (not in Configuration state). Not accessible while a Configuration-state timeout (OP14 operation) is in progress. Controls the maximum amount of time that FCCU can be in Configuration state.

NOTE

When a Configuration-state timeout occurs, FCCU changes the value of this register to its reset value.

Address: 0h base + B8h offset = B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																													TO		
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_CFG_TO field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 TO	Configuration-State Timer Interval Controls the maximum amount of time (T_max) that FCCU can be in Configuration state according to this equation: T_max = $T_{16MHz} \times 2^{(TO + 10)}$ If you put FCCU in Configuration state and this timer interval expires (called a Configuration-state timeout), then FCCU: • Automatically leaves Configuration state and enters Normal state. • Changes the value of the Configuration (FCCU_CFG) register to its Configuration-state-timeout value and the value of each of the other configuration registers to its reset value. For information on the Configuration (FCCU_CFG) register's Configuration-state-timeout value, see FCCU_CFG register bit value sources (N and C) by event. For a list of configuration registers, see Configuration registers.

Table continues on the next page...

FCCU_CFG_TO field descriptions (continued)

Field	Description
	NOTE: Make sure the Configuration-state timer interval is less than the FOSU module's timer interval; otherwise, a fault that occurs while FCCU is in Configuration state can cause FOSU to generate a chip reset. The FOSU timer interval (FOSU_COUNT) is chip-specific. See the chip-specific FCCU information. 000: 64 µs ... 111: 8192 µs

3.10.12 IO Control (FCCU_EINOUT)

The FCCU_EINOUT register allows the following operations typically in NORMAL state:

- to control the EOUT[1] output level when the FCCU is configured in "Test1" or "Test0" fault output mode (FCCU_CFG.FOM)
- to control the EOUT[0] output level when the FCCU is configured in "Test1" or "Test2" fault output mode (FCCU_CFG.FOM)
- to observe the EOUT[1:0] signals in input mode

Table 3-10 shows Bi-stable encoding.

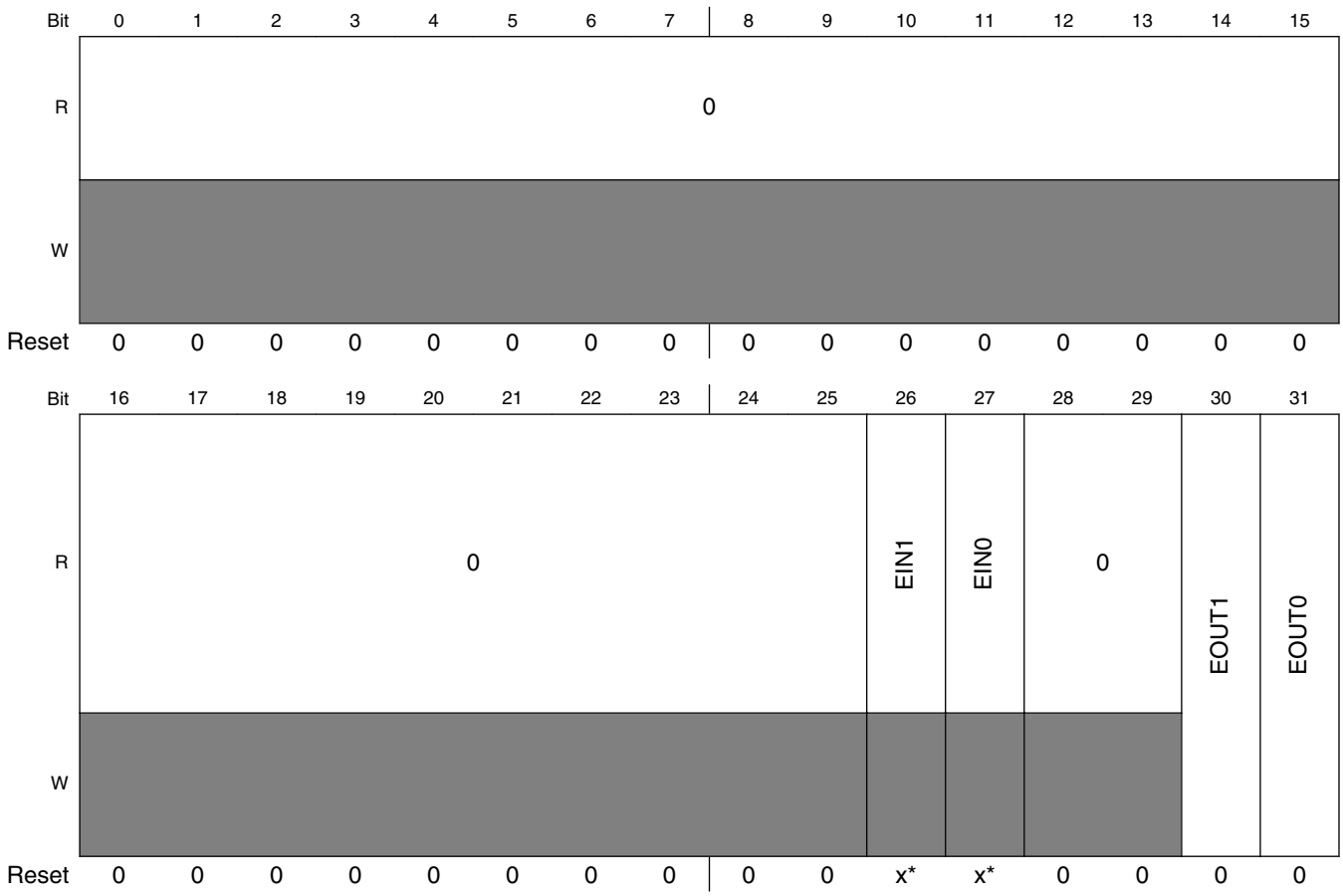
Table 3-10. Bi-stable encoding

Mode = FCCU_CFG.FOM	EOUT[0]	EOUT[1]
Test1	output	output
Test2	output	input
Test0	input	output

NOTE

Due to the resynchronization stage of the EOUT interface, there is a latency of a few CLKSAFE cycles following a write/read operation of the FCCU_EINOUT register.

Address: 0h base + BCh offset = BCh



* Notes:
- x = Undefined at reset.

FCCU_EINOUT field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 EIN1	Error Input 1 Applies only when the EOUT signals are active (FCCU_SET_AFTER_RESET). Indicates the state of the EIN1 signal. 0 Low 1 High
27 EIN0	Error Input 0 Applies only when the EOUT signals are active (FCCU_SET_AFTER_RESET). Indicates the state of the EIN0 signal. 0 Low 1 High
28–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FCCU_EINOUT field descriptions (continued)

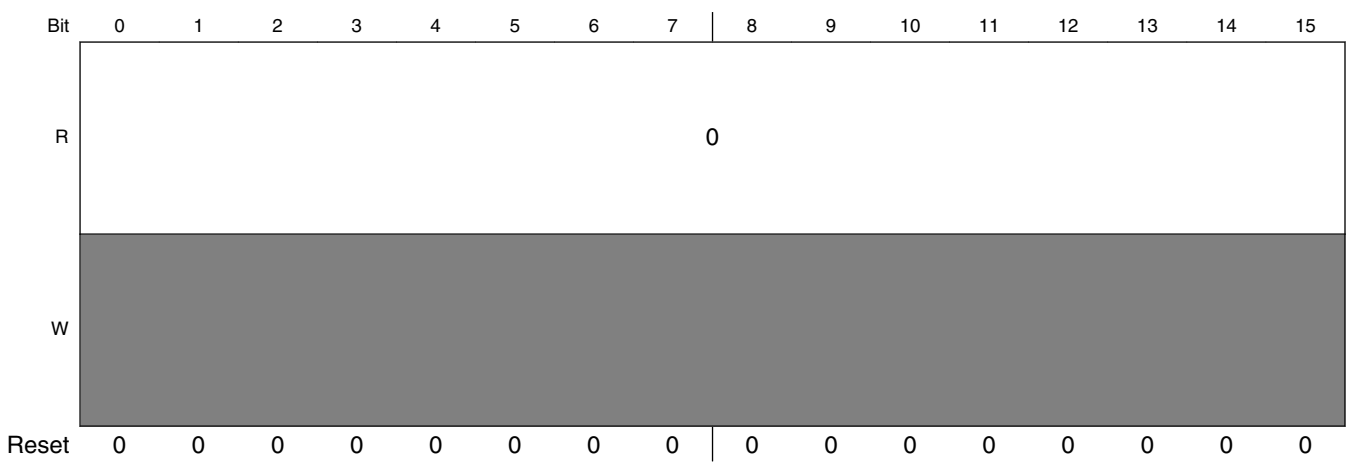
Field	Description
30 EOUT1	Error out 1 (significant only if the FCCU_CFG.FOM = Test1 or Test0 => EOUT[1] configured in output mode). The EOUT1 set/clear the respective EOUT[1] output signal if FCCU_CFG.FOM = 110 or 101, otherwise it is a "don't-care" value. **NOTE:** When the configuration watchdog timer expires, FCCU changes the value of this field to its reset value. 0 force EOUT[1] = 0 1 force EOUT[1] = 1
31 EOUT0	Error out 0 (significative only if the FCCU_CFG.FOM = Test1 or Test2 => EOUT[0] configured in output mode). The EOUT0 set/clear the respective EOUT[0] output signal if FCCU_CFG.FOM = 110 or 111, otherwise it is a "don't care" value. **NOTE:** When the configuration watchdog timer expires, FCCU changes the value of this field to its reset value. 0 force EOUT[0] = 0 1 force EOUT[0] = 1

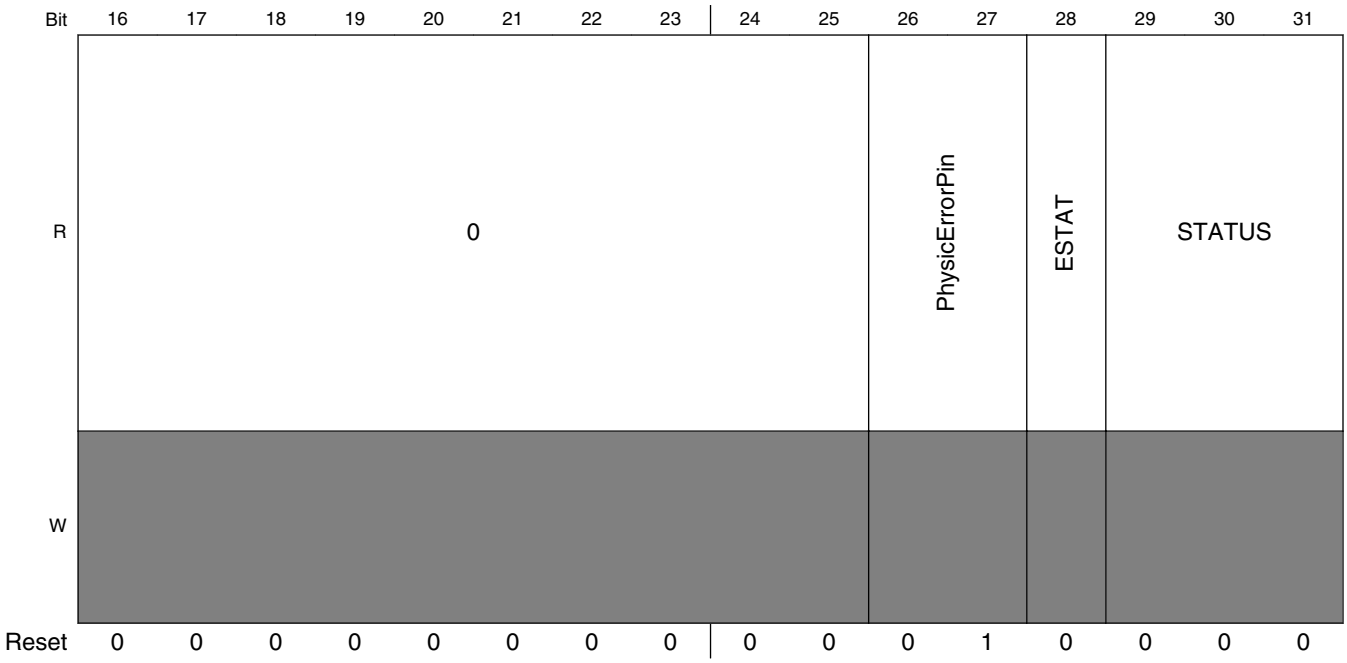
3.10.13 Status (FCCU_STAT)

The FCCU_STAT register includes the FCCU status for debugging/test purposes. The FCCU finite state machine operates by the RC oscillator clock asynchronous with the system clock. The FCCU status read operation requires a safe mechanism operated by a HW/SW synchronization sequence. The SW application executes a FCCU status read operation by the following sequence:

1. Set the OP3 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU status (FCCU_STAT register).

Address: 0h base + C0h offset = C0h





FCCU_STAT field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–27 PhysicErrorPin	Fault Output (EOUT) States Applies only after successfully running an OP3 operation (OPR) and only when the EOUT signals are active (FCCU_SET_AFTER_RESET). Indicates the states that FCCU is driving on the EOUT signals. 00 EOUT1 is low; EOUT0 is low. 01 EOUT1 is low; EOUT0 is high. 10 EOUT1 is high; EOUT0 is low. 11 EOUT1 is high; EOUT0 is high.
28 ESTAT	Fault State Applies only after successfully running an OP3 operation (OPR). Indicates whether FCCU is in Fault state. 0 Not in Fault state (in Normal, Alarm, or Configuration state) 1 In Fault state
29–31 STATUS	FCCU State Applies only after successfully running an OP3 operation (OPR). Indicates FCCU's state. 000 Normal 001 Configuration 010 Alarm 011 Fault 100 Reserved 101 Reserved 110 Reserved 111 Reserved

3.10.14 NA Freeze Status (FCCU_N2AF_STATUS)

The N2AF_STATUS register contains a unique code to identify the "noncritical fault" source (fccu_ncf[x]) that caused the state transition from the NORMAL state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

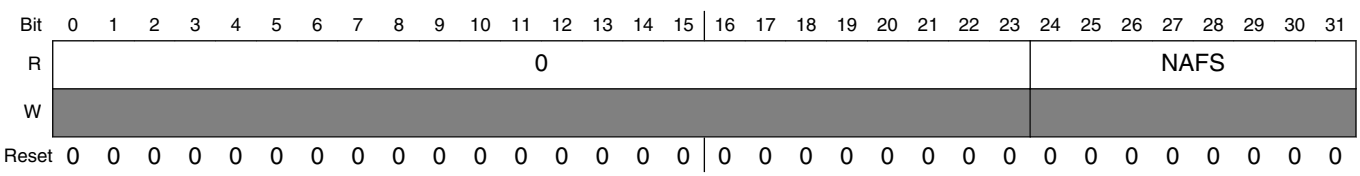
The SW application executes the N2AF_STATUS read operation by the following sequence:

1. Set the OP4 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2AF_STATUS register.

The SW application executes the N2AF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Address: 0h base + C4h offset = C4h



FCCU_N2AF_STATUS field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 NAFS	Normal to Alarm Frozen Status 00000000 No transition from NORMAL to ALARM state 00000001 NORMAL to ALARM state transition cause => fccu_ncf[0] fault 00000010 NORMAL to ALARM state transition cause => fccu_ncf[1] fault 00000011 NORMAL to ALARM state transition cause => fccu_ncf[2] fault ... 10000000 NORMAL to ALARM state transition cause => fccu_ncf[127]/ fault 11111111 NORMAL to ALARM state transition cause => multiple fccu_ncf[]/ faults

3.10.15 AF Freeze Status (FCCU_A2FF_STATUS)

The A2FF_STATUS register contains a unique code to identify the timeout trigger (noncritical fault) that caused the state transition from the ALARM state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

The SW application executes the A2FF_STATUS read operation by the following sequence:

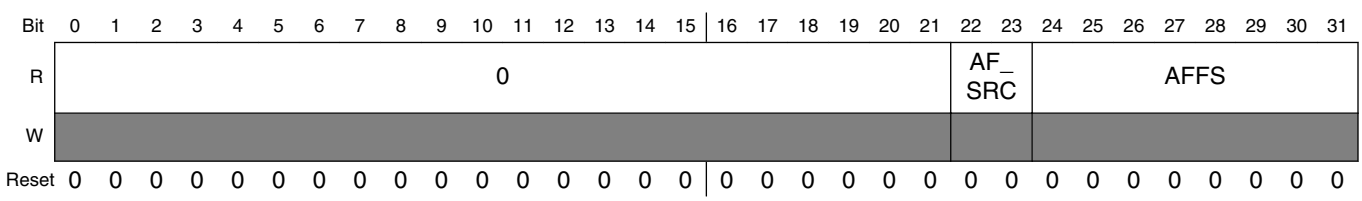
1. Set the OP5 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the A2FF_STATUS register.

The SW application executes the FCCU_AFFS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).

All the freeze registers are cleared by this operation.

Address: 0h base + C8h offset = C8h



FCCU_A2FF_STATUS field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 AF_SRC	Fault source 00 No fault 01 Reserved 10 "Noncritical" fault 11 Multiple and/or "noncritical" faults
24–31 AFFS	Alarm to Fault Frozen Status 00h: No transition from ALARM to FAULT state 01h: ALARM to FAULT state transition cause => fccu_ncf[0]/ fault timeout fault 02h: ALARM to FAULT state transition cause => fccu_ncf[1]/ fault timeout fault 03h: ALARM to FAULT state transition cause => fccu_ncf[2]/ fault timeout fault

Table continues on the next page...

FCCU_A2FF_STATUS field descriptions (continued)

Field	Description
	... 80h: ALARM to FAULT state transition cause => fccu_ncf[127]/ fault timeout fault FFh: ALARM to FAULT state transition cause => multiple fccu_ncf[]/ faults timeout faults

3.10.16 NF Freeze Status (FCCU_N2FF_STATUS)

The N2FF_STATUS register contains a unique code to identify the source of the noncritical fault that caused the state transition from the NORMAL state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes. The SW application executes the N2FF_STATUS read operation by the following sequence:

1. Set the OP6 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2FF_STATUS register.

The SW application executes the N2FF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Address: 0h base + CCh offset = CCh

Bit	0–15	16–21	22–23	24–31
R	0		NF_SRC	NFFS
W				
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0	0 0	0 0 0 0 0 0 0 0

FCCU_N2FF_STATUS field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 NF_SRC	NF_SRC: Fault source 00 No fault 01 Reserved 10 "Noncritical" fault 11 Multiple "noncritical" faults

Table continues on the next page...

FCCU_N2FF_STATUS field descriptions (continued)

Field	Description
24–31 NFFS	Normal to Fault Frozen Status 00h: No transition from NORMAL to FAULT state 01h: NORMAL to FAULT state transition cause => fccu_ncf[0] fault 02h: NORMAL to FAULT state transition cause => fccu_ncf[1] fault 03h: NORMAL to FAULT state transition cause => fccu_ncf[2] fault ... 80h: NORMAL to FAULT state transition cause => fccu_ncf[127] fault FFh: NORMAL to FAULT state transition cause => multiple fccu_ncf[] faults

3.10.17 FA Freeze Status (FCCU_F2A_STATUS)

The F2AF_STATUS register contains a unique code to identify the source of the noncritical fault (fccu_ncf[x]) that caused the state transition from the FAULT state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

The SW application executes the F2AF_STATUS read operation by the following sequence:

1. Set the OP7 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the F2AF_STATUS register.

The SW application executes the F2AF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Address: 0h base + D0h offset = D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																							FAFS								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_F2A_STATUS field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 FAFS	Fault to Normal Frozen Status 00h: No transition from FAULT to ALARM state 01h: FAULT to ALARM state transition cause => fccu_ncf[0]/ fault 02h: FAULT to ALARM state transition cause => fccu_ncf[1]/ fault 03h: FAULT to ALARM state transition cause => fccu_ncf[2]/ fault ... 80h: FAULT to ALARM state transition cause => fccu_ncf[127]/ fault FFh: FAULT to ALARM state transition cause => multiple fccu_ncf[] faults

3.10.18 SC Freeze Status (FCCU_SCFS)

The FCCU_SCFS register contains the status of the RCCx checkers. This register is for test/debug purposes.

Two interrupt lines are asserted by the correspondent status bits (FCCU_SCFS.RCCSx).

The SW application executes the FCCU_SCFS read operation by the following sequence:

1. Set the OP8 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU_SCFS register.

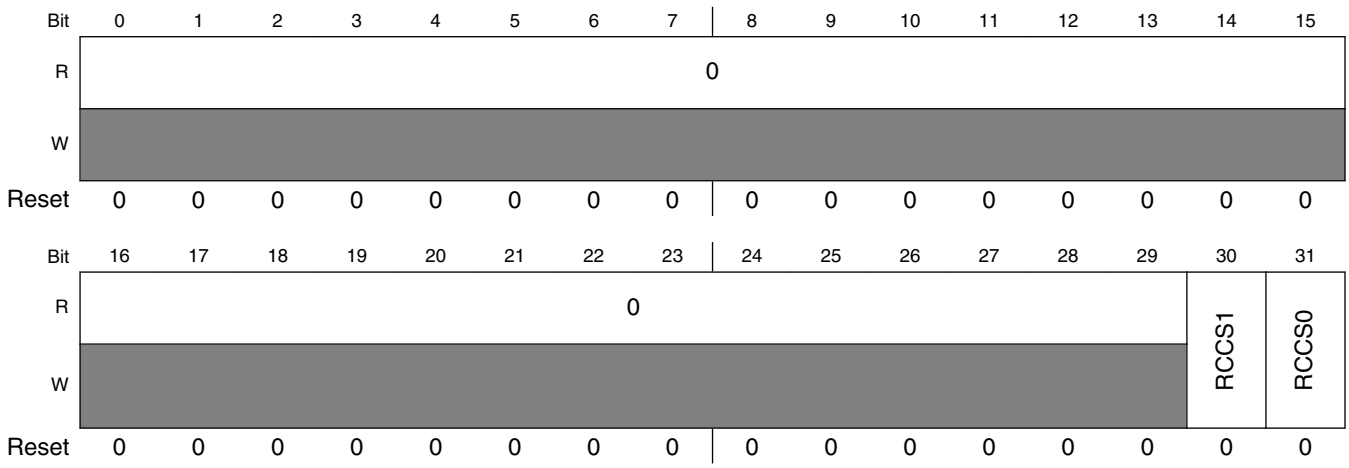
The SW application executes the FCCU_SCFS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). All the freeze registers are cleared by this operation. The RCCx checkers must be disabled (FCCU_CFG.RCCEx = 00b) during a clear operation otherwise the RCCSx flags are asserted again.

NOTE

The INTC will never attend the second interrupt line since RCCS0 and RCCS1 are always cleared together by the operation above. From SW perspective the INTC should be configured to attend RCCS0 or RCCS1 interrupt line only, not the pair.

Address: 0h base + D4h offset = D4h



FCCU_SCFS field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 RCCS1	RCC1 Status **NOTE:** This field can be read and written but writing 1 has no effect. 0 No miscompares 1 A miscompare has been detected. An interrupt line is asserted.
31 RCCS0	RCC0 Status **NOTE:** This field can be read and written but writing 1 has no effect. 0 No miscompares 1 A miscompare has been detected. The interrupt line is asserted.

3.10.19 Noncritical Fault Fake (FCCU_NCFF)

The FCCU_NCFF register contains a unique code to set a noncritical fault in mutually exclusive mode by the external FAULT interface (signal setting). It allows the SW emulation of the noncritical faults, by the injection of the fault directly in the FAULT root, in order to verify the entire path and reaction. The reaction following a fake noncritical fault cannot be masked. The FCCU_NCFF is a write-only register with a set of codes corresponding to each noncritical fault injection.

Address: 0h base + DCh offset = DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																															
W																										FNCFC						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NCFF field descriptions

Field	Description
0–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 FNCFC	FNCFC/ : Fake noncritical fault code **NOTE:** Only writing to this register, fake fault injection occurs, writing 00 and default value being zero give different results. 00h: Fake noncriticalfault injection at noncriticalfault source 0 01h: Fake noncriticalfault injection at noncriticalfault source 1 02h: Fake noncriticalfault injection at noncriticalfault source 2 ... 7Fh: Fake noncriticalfault injection at noncriticalfault source 127

3.10.20 IRQ Status (FCCU_IRQ_STAT)

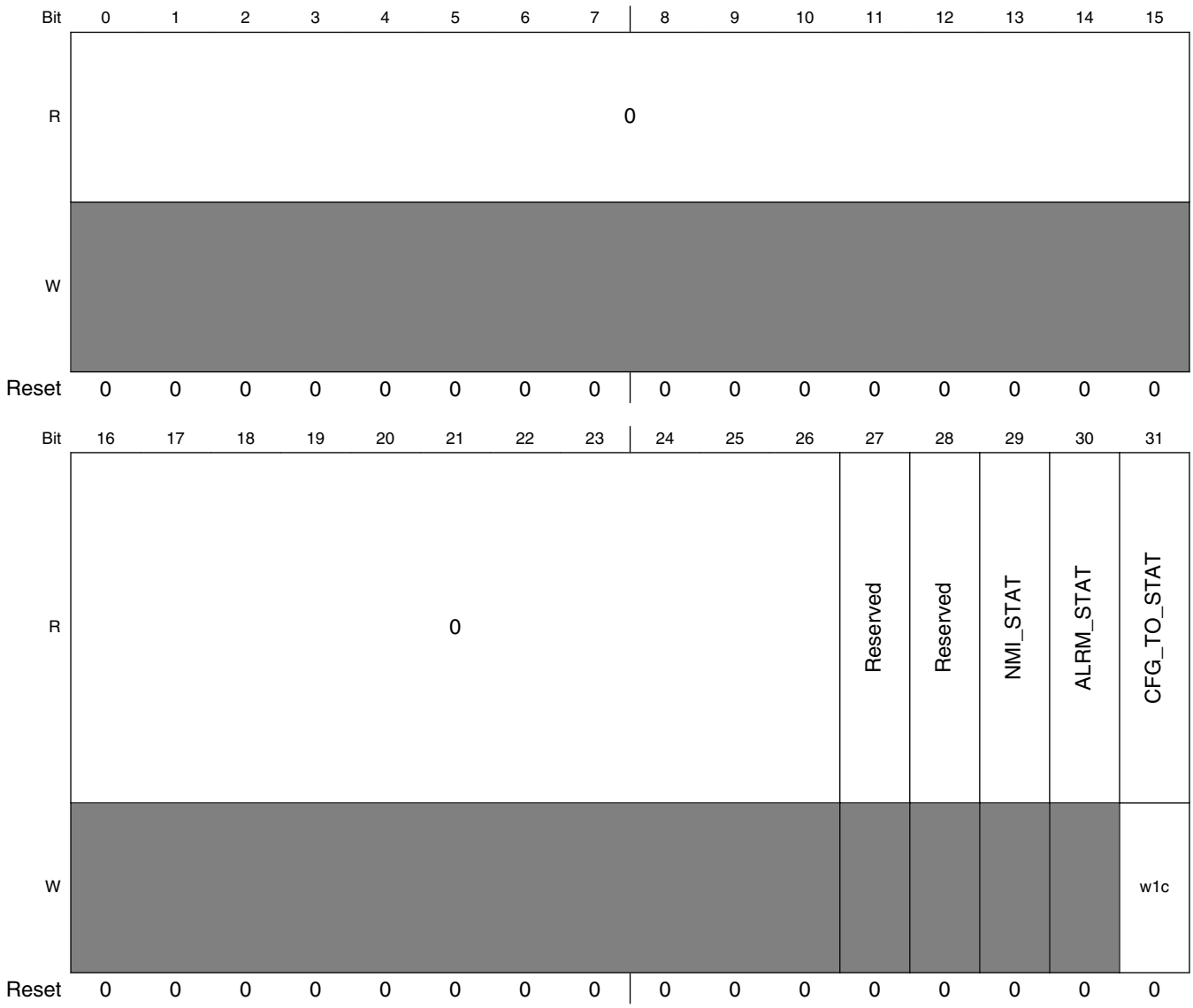
The FCCU_IRQ_STAT register provides the FCCU interrupt status related to the following events:

- Configuration timeout error
- Alarm interrupt
- NMI interrupt

The configuration timeout interrupt is asserted if the CFG_TO_STAT bit of the FCCU_IRQ_STAT register is set and the CFG_TO_IEN bit of the FCCU_IRQ_EN register is also set. It is cleared when a 1 is written to the CFG_TO_STAT bit.

The NMI and ALARM interrupts are asserted and cleared according to the FCCU state. The status bits of the FCCU_IRQ_STAT trace the status of the related interrupt lines.

Address: 0h base + E0h offset = E0h



FCCU_IRQ_STAT field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 Reserved	This field is reserved.

Table continues on the next page...

FCCU_IRQ_STAT field descriptions (continued)

Field	Description
28 Reserved	This field is reserved.
29 NMI_STAT	NMI Interrupt Status 0 NMI interrupt is OFF 1 NMI interrupt is ON
30 ALRM_STAT	Alarm Interrupt Status 0 Alarm interrupt is OFF 1 Alarm interrupt is ON
31 CFG_TO_STAT	Configuration Timeout Status 0 No configuration timeout error 1 Configuration timeout error

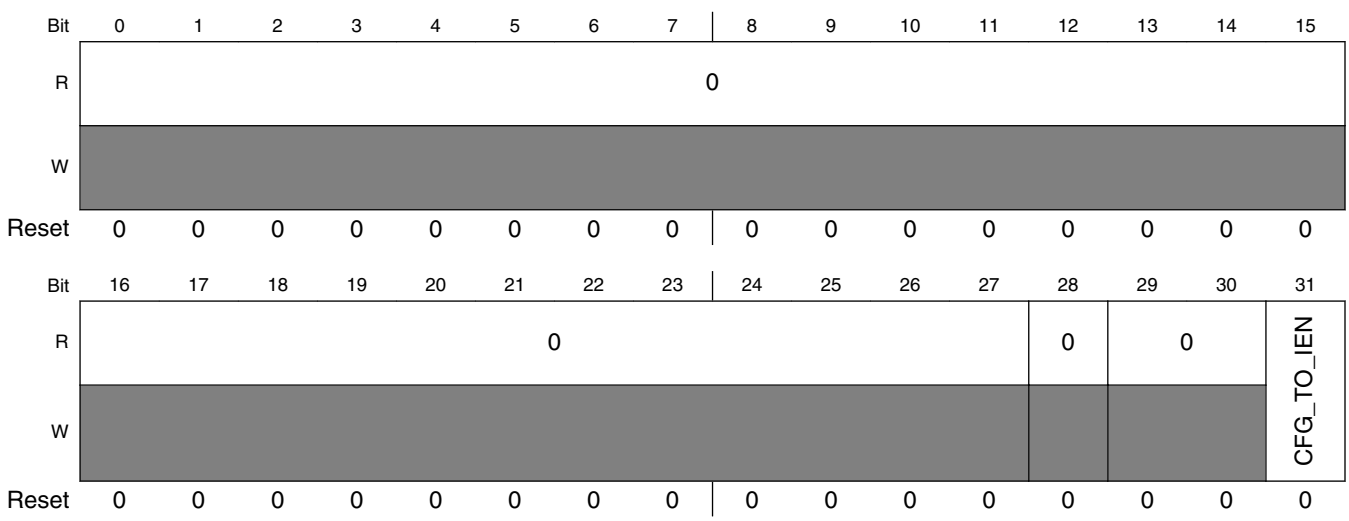
3.10.21 IRQ Enable (FCCU_IRQ_EN)

The FCCU_IRQ_EN register defines the FCCU interrupt enable register related to the following event:

- Configuration timeout error

The configuration timeout interrupt is asserted if the CFG_TO_STAT bit of the FCCU_IRQ_STAT register is set and the CFG_TO_IEN bit of the FCCU_IRQ_EN register is also set.

Address: 0h base + E4h offset = E4h



FCCU_IRQ_EN field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CFG_TO_IEN	Configuration Timeout Interrupt Enable 0 Configuration timeout interrupt disabled 1 Configuration timeout interrupt enabled

3.10.22 XTMR (FCCU_XTMR)

The FCCU_XTMR register contains the read values of the Alarm or Watchdog Timer. These timers are clocked by CLKSAFE.

The SW application executes the timer read operation by the following sequence:

1. Set any of the following operations into the FCCU_CTRL.OPR field:
   - OP17
   - OP19
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU_XTMR register.

Table 3-11 shows Timer state/value.

Table 3-11. Timer state/value

TIMER	CONFIG state	NORMAL state	ALARM state	FAULT state
ALARM	00000000h	Initial value	Running	Idle/End of count
CFG	Running	0001FFFFh	0001FFFFh	0001FFFFh

Address: 0h base + E8h offset = E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	XTMR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_XTMR field descriptions

Field	Description
0–31 XTMR	Alarm/Watchdog/Safe request timer

FCCU_XTMR field descriptions (continued)

Field	Description
	The current timer value is measured in CLKSAFE cycles. These bits can be read by the software.

3.10.23 Mode Controller Status (FCCU_MCS)

Indicates the last four chip modes—as defined by the MC_ME module—and whether FCCU was in Fault state and NMIOUT was asserted (as the result of a noncritical fault) when FCCU captured each mode.

Address: 0h base + ECh offset = ECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VL3	FS3	0		MCS3				VL2	FS2	0		MCS2			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VL1	FS1	0		MCS1				VL0	FS0	0		MCS0			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information. (The MC_ME module changes the value of the register immediately after the chip leaves RESET mode.)

FCCU_MCS field descriptions

Field	Description
0 VL3	Valid 3—Indicates whether MCS3 and FS3 are valid. 0 Invalid 1 Valid
1 FS3	Fault Status 3 Indicates whether FCCU was in Fault state and NMIOUT was asserted (as the result of a noncritical fault) when FCCU captured MCS3. **NOTE:** This field might occasionally be inaccurate because the MC_ME module provides the chip modes to FCCU in synchronization with CLKSYS, but FCCU captures the chip modes in synchronization with CLKSAFE. 0 Not in Fault state or NMIOUT not asserted 1 In Fault state and NMIOUT asserted

Table continues on the next page...

FCCU_MCS field descriptions (continued)

Field	Description
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 MCS3	Mode Controller State 3—Indicates the fourth most recent chip mode.
8 VL2	Valid 2—Indicates whether MCS2 and FS2 are valid. 0 Invalid 1 Valid
9 FS2	Fault Status 2 Indicates whether FCCU was in Fault state and NMIOUT was asserted (as the result of a noncritical fault) when FCCU captured MCS2. **NOTE:** This field might occasionally be inaccurate because the MC_ME module provides the chip modes to FCCU in synchronization with CLKSYS, but FCCU captures the chip modes in synchronization with CLKSAFE. 0 Not in Fault state or NMIOUT not asserted 1 In Fault state and NMIOUT asserted
10–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 MCS2	Mode Controller State 2—Indicates the third most recent chip mode.
16 VL1	Valid 1—Indicates whether MCS1 and FS1 are valid. 0 Invalid 1 Valid
17 FS1	Fault Status 1 Indicates whether FCCU was in Fault state and NMIOUT was asserted (as the result of a noncritical fault) when FCCU captured MCS1. **NOTE:** This field might occasionally be inaccurate because the MC_ME module provides the chip modes to FCCU in synchronization with CLKSYS, but FCCU captures the chip modes in synchronization with CLKSAFE. 0 Not in Fault state or NMIOUT not asserted 1 In Fault state and NMIOUT asserted
18–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 MCS1	Mode Controller State 1—Indicates the second most recent chip mode.
24 VL0	Valid 0—Indicates whether MCS0 and FS0 are valid. 0 Invalid 1 Valid
25 FS0	Fault Status 0 Indicates whether FCCU was in Fault state and NMIOUT was asserted (as the result of a noncritical fault) when FCCU captured MCS0.

Table continues on the next page...

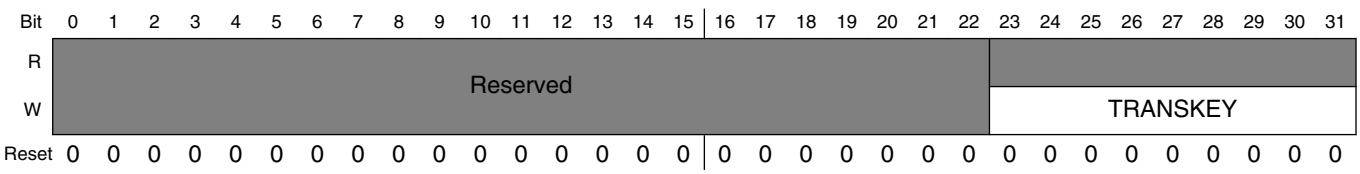
FCCU_MCS field descriptions (continued)

Field	Description
	NOTE: This field might occasionally be inaccurate because the MC_ME module provides the chip modes to FCCU in synchronization with CLKSYS, but FCCU captures the chip modes in synchronization with CLKSAFE. 0 Not in Fault state or NMIOUT not asserted 1 In Fault state and NMIOUT asserted
26–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 MCS0	Mode Controller State 0—Indicates the most recent (current) chip mode.

3.10.24 Transient Configuration Lock (FCCU_TRANS_LOCK)

Writable only by code running in Supervisor mode. Temporarily locks and unlocks the configuration. Locking the configuration prevents FCCU from entering Configuration state.

Address: 0h base + F0h offset = F0h



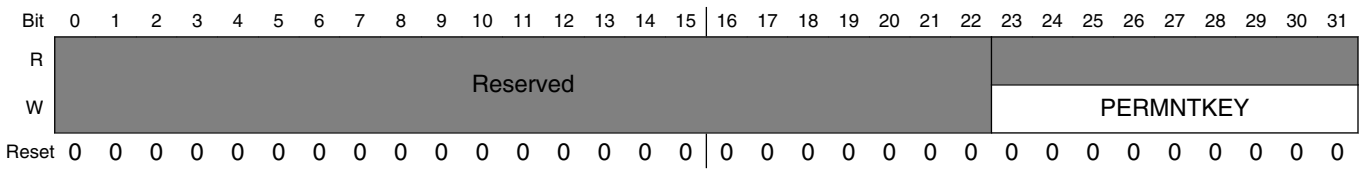
FCCU_TRANS_LOCK field descriptions

Field	Description
0–22 Reserved	This field is reserved.
23–31 TRANSKEY	Transient Configuration Lock Writable only by code running in Supervisor mode. Temporarily locks and unlocks the configuration. Locking the configuration prevents FCCU from entering Configuration state. BCh: Unlock. Any other value: Lock.

3.10.25 Permanent Configuration Lock (FCCU_PERMNT_LOCK)

Writable only by code running in Supervisor mode. Permanently locks the configuration, which prevents FCCU from entering Configuration state until FCCU is reset.

Address: 0h base + F4h offset = F4h



FCCU_PERMNT_LOCK field descriptions

Field	Description
0–22 Reserved	This field is reserved.
23–31 PERMNTKEY	Permanent Configuration Lock Writable only by code running in Supervisor mode. Permanently locks the configuration, which prevents FCCU from entering Configuration state until FCCU is reset. FFh: Lock. Any other value: Do nothing.

3.10.26 Delta T (FCCU_DELTA_T)

The FCCU_DELTA_T register is used for programming the value of delta_T constant, in microseconds.

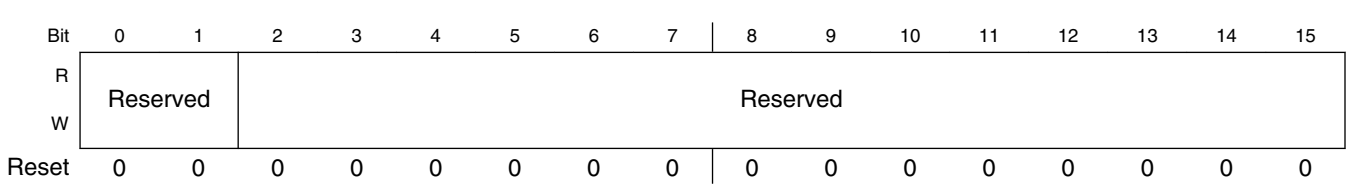
NOTE

This register can be written only when the FCCU is in CONFIG state.

NOTE

Reserved bits should always be written as all 0's.

Address: 0h base + F8h offset = F8h



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved		DELTA_T													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_DELTA_T field descriptions

Field	Description
0–1 Reserved	This field is reserved.
2–15 Reserved	This field is reserved.
16–17 Reserved	This field is reserved.
18–31 DELTA_T	Bistable Minimum Fault-Output (EOUT) Faulty Interval Applies only to Bistable fault-output mode (FOM). Controls the minimum amount of time (T_min) that the fault-output (EOUT) timer runs according to this equation: T_min = 250 µs + DELTA_T The following values and meanings are for DELTA_T: 00_0000_0000_0000: 0 µs 00_0000_0000_0001: 1 µs … 11_1111_1111_1110: 16,382 µs 11_1111_1111_1111: 16,383 µs **NOTE:** The durations shown for the DELTA_T values are for CLKSAFE clock signals running at exactly 16 MHz. However, the CLKSAFE signals are sourced from an internal chip clock signal (for example, IRC)—or an integer division of it (see the chip-specific FCCU information)—that has a trimmed frequency variation. Therefore, the DELTA_T durations may vary. See your chip's data sheet for the trimmed frequency variation (for example, δF_{var}).

3.10.27 IRQ Alarm Enable (FCCU_IRQ_ALARM_EN*n*)

These registers enable the corresponding IRQ alarm.

IRQ Alarm Enable (FCCU_IRQ_ALARM_EN*n*) register (value of n)	Offset	IRQENx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	0FCh	31	0
1	100h	63	32
2	104h	95	64

NOTE

These registers can be written only when the FCCU is in CONFIG state.

Address: 0h base + FCh offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IRQENx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_IRQ_ALARM_EN*n* field descriptions

Field	Description
0–31 IRQENx	IRQ alarm enable For the mapping of the IRQENx fields among registers, see the earlier table in this register description. 0 Alarm is disabled for error source x. 1 Alarm is enabled for error source x.

3.10.28 NMI Enable (FCCU_NMI_EN*n*)

These registers enable the NMI.

Table 3-12 shows FCCU NMI enable register channels.

Table 3-12. FCCU NMI enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
10Ch	FCCU_NMI_EN0	NMIENE[31:0]
110h	FCCU_NMI_EN1	NMIENE[63:32]
114h	FCCU_NMI_EN2	NMIENE[95:64]

NOTE

These registers can be written only when the FCCU is in CONFIG state.

Address: 0h base + 10Ch offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NMIENx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NMI_EN*n* field descriptions

Field	Description
0–31 NMIENx	NMI enable See Table 3-12 for register offset to channel number relationship. 0 NMI is disabled for error (NCF) source x. 1 NMI is enabled for error (NCF) source x.

3.10.29 Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN*n*)

Writable only when FCCU is in Configuration state. Applies only when the EOUT signals are active (FCCU_SET_AFTER_RESET). When FCCU is configured for Bistable fault-output mode (FOM), controls whether fault-output (EOUT) signaling is enabled for the associated noncritical fault channel (x). (For other fault-output modes, fault-output signaling is always enabled, regardless of the value of this field.)

When FCCU is configured for a fault-output mode other than Bistable, controls whether FCCU asserts the FIF signal when a fault occurs on the associated noncritical fault channel (x). (When FCCU is configured for Bistable fault-output mode, FCCU always asserts the FIF signal when a fault occurs on the channel, regardless of the value of this field.)

Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN*n*) register (value of n)	Offset	EOUTENx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	11Ch	31	0
1	120h	63	32
2	124h	95	64

Address: 0h base + 11Ch offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EOUTENx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_EOUT_SIG_EN*n* field descriptions

Field	Description
0–31 EOUTENx	Bistable Fault-Output (EOUT) Mode Signaling Enable x

FCCU_EOUT_SIG_EN*n* field descriptions (continued)

Field	Description
	Writable only when FCCU is in Configuration state. Applies only when the EOUT signals are active (FCCU_SET_AFTER_RESET). When FCCU is configured for Bistable fault-output mode (FOM), controls whether fault-output (EOUT) signaling is enabled for the associated noncritical fault channel (x). (For other fault-output modes, fault-output signaling is always enabled, regardless of the value of this field.) When FCCU is configured for a fault-output mode other than Bistable, controls whether FCCU asserts the FIF signal when a fault occurs on the associated noncritical fault channel (x). (When FCCU is configured for Bistable fault-output mode, FCCU always asserts the FIF signal when a fault occurs on the channel, regardless of the value of this field.) **NOTE:** When FCCU is configured for a fault-output mode other than Bistable, you must set this field to enabled to ensure that FCCU asserts the FIF signal when a fault occurs on the associated noncritical fault channel (x) so the FOSU module doesn't mistakenly generate a destructive chip reset. For the mapping of the EOUTENx fields among registers, see the earlier table in this register description. 0 Disabled 1 Enabled

3.10.30 Configuration registers

3.10.30.1 Definition

Configuration registers are registers that:

- Let you configure FCCU's Alarm-state timer interval, fault channels, and fault-output (EOUT) signals.
- You can write to only when the configuration is not locked (see Lock and unlock the configuration) and FCCU is in Configuration state (see Put FCCU in Configuration or Normal state).
- Save the values you write to them while FCCU is in Configuration state only after you manually put FCCU in Normal state. If FCCU automatically leaves Configuration state and enters Normal state because the configuration-timer interval (TO) expires (called a Configuration-state timeout), FCCU changes the value of the FCCU_CFG register to its Configuration-state-timeout value and the value of each of the other configuration registers to its reset value. For information on the Configuration-state timeout value, see FCCU_CFG register bit value sources (N and C) by event.

3.10.30.2 The configuration registers

The following registers are the configuration registers, listed in offset order from lowest to highest:

- Configuration (FCCU_CFG)
- Noncritical Fault Configuration (FCCU_NCF_CFG*n*)
- Noncritical Fault State Configuration (FCCU_NCFS_CFG*n*)
- Noncritical Fault Enable (FCCU_NCF_E*n*)
- Noncritical Fault Timeout Enable (FCCU_NCF_TOE*n*)
- Noncritical Fault Timeout (FCCU_NCF_TO)
- Delta T (FCCU_DELTA_T)
- IRQ Alarm Enable (FCCU_IRQ_ALARM_EN*n*)
- NMI Enable (FCCU_NMI_EN*n*)
- Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN*n*)

3.10.31 FCCU_CFG register bit value sources (N and C) by event

Certain events affect the bit values of the Configuration (FCCU_CFG) register; when one of these events occurs, the bit value is either unaffected, or FCCU changes it to one of the following:

- N: a chip-specific (or, for some chips, user-specified) value provided by the bit's associated NVMCFG signal
- C: a chip-specific value provided by the bit's associated constant inside the module

For each bit in the Configuration (FCCU_CFG) register, this table shows the associated NVMCFG signal, if any, and indicates the source of the bit value for the events that affect it. For specific values of *N* and *C* for each bit, see the chip-specific FCCU information.

Bit	NVMCFG signal	FCCU reset	Configuration-state timeout	OP31 operation
0	21	N	C	N
1	20	N	C	N
2	19	N	C	N
3	18	N	C	N
4	17	N	C	N
5	16	N	C	N
6	15	N	C	N
7	—	C	C	(Unaffected)
8	—	C	C	(Unaffected)
9	—	C	C	(Unaffected)
10	—	C	C	(Unaffected)
11	—	C	C	(Unaffected)
12	—	C	C	(Unaffected)

Table continues on the next page...

Bit	NVMCFG signal	FCCU reset	Configuration-state timeout	OP31 operation
13	—	C	C	(Unaffected)
14	—	C	C	(Unaffected)
15	—	C	C	(Unaffected)
16	14	N	C	N
17	13	N	C	N
18	12	N	C	N
19	—	C	C	(Unaffected)
20	11	N	C	N
21	10	N	C	N
22	9	N	C	N
23	8	N	C	N
24	7	N	C	N
25	6	N	C	N
26	5	N	C	N
27	4	N	C	N
28	3	N	C	N
29	2	N	C	N
30	1	N	C	N
31	0	N	C	N

3.11 FCCU Output Supervision Unit

The FOSU provides a supervision of the primary fault notification path by analyzing FCCU behavior for correctness. It waits for any reaction of the FCCU in a fixed time window after a fault is signaled.

The intention of the FOSU is to provide a secondary fault reaction path in most cases when the FCCU fails but not to needlessly propagate a fault which is already handled by the FCCU in a full chip reset. Only a failed primary fault reaction (that is, FCCU's failure) is a reason for the secondary reaction to take over (and generate a destructive reset request).

There is a 'do nothing' input coming from the FCCU that indicates that the FCCU is programmed for no reaction for ALL FAULTS. It is a "static" input in the sense that it does not change after FCCU configuration. The FOSU masks the incoming faults with the 'do nothing' control from the FCCU, meaning that a fault is not captured by the FOSU if the 'do nothing' signal is asserted, (i.e., a disabled fault). There is no minimum pulse

width requirement on the fault indication other than what is required by the technology, which is the same as that of the FCCU. FOSU does not monitor FCCU for the case of faults occurring during CONFIG state.

The FOSU contains a timer with a duration of FOSU_COUNT, driven by CLKSAFE. The timer is initialized and started on any captured, enabled fault. While the timer is running, any subsequent captured fault will neither restart nor reinitialize the timer. The timer is stopped when the FCCU shows any of the following reactions (the FOSU does not check whether the reaction is the configured one for the faults which occurred):

- Reset: Long or short functional reset
- IRQ: NMI or Alarm
- Error out triggered (by FCCU or by SW)

When the timer is stopped, the fault capture logic is cleared in order to ensure that the timer is not restarted due to faults still 'stuck' in the capture logic. The timer will then be restarted by the next new failure indication. When the timer expires, the FOSU's failure indicator output is asserted after it ensures that the fault is enabled and the static "fccu program to do nothing" signal is deasserted. This is because FCCU uses settings after it exits CONFIG state, even if fault captured before the exit.

The FOSU's failure indicator output is connected to one of the MC_RGM's 'destructive' reset inputs, so its assertion will cause a reset sequence to be initiated starting at PHASE0. The FOSU module is reset with the same reset as is used by the FCCU, which is asserted on power-on, 'destructive', and external resets. When this reset is asserted, the FOSU's capture logic is cleared, its timer is kept stopped and in a non-expired state, and its failure indicator output is deasserted.

NOTE

FOSU is triggered on assertion of enabled fault. In case the triggering fault is disabled, FOSU times-out without reaction.

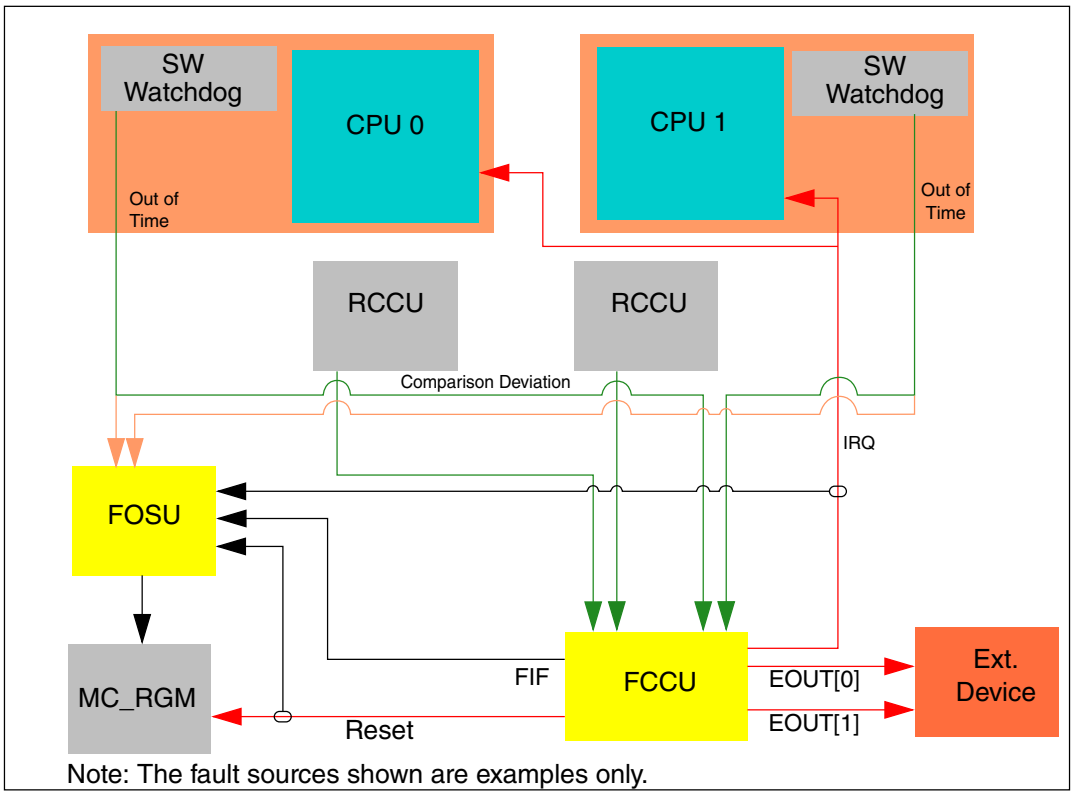


Figure 3-17. FOSU connections to the FCCU and MC_RGM

It is important to remember that there will be no FCCU reaction to a fault while the FCCU is in the CONFIG state. For this reason, the FCCU should not be kept in CONFIG for longer than the FOSU_COUNT duration. Otherwise, there is a risk that an incoming error report causing the FOSU to mistakenly see the FCCU as having failed, and then resets the MCU.

NOTE

FCCU and FOSU timeout counter settings: The FCCU counter value should always be programmed less than the FOSU counter value, FOSU_COUNT, or destructive resets may be generated by an FOSU time-out. This ensures that the FOSU reacts only when the FCCU fails to react to any particular fault.

3.12 Use cases and limitations

3.12.1 Configuration guidelines

Follow these guidelines to configure FCCU:

- If you want FCCU to react to a fault on a noncritical fault channel:
  - Enable the channel (NCFEx).
  - Enable at least one type of Fault-state reaction for the channel: chip reset (NCFSCx), fault-output (EOUT) signaling (EOUTENx), or nonmaskable interrupt (NMIENx).
  - If you enable chip reset as the type of Fault-state reaction for the channel (NCFSCx), enable either Alarm state (NCFTOEx) or at least one other type of Fault-state reaction for the channel: fault-output (EOUT) signaling (EOUTENx) or nonmaskable interrupt (NMIENx).
  - If you enable Alarm state for the channel (NCFTOEx), enable the Alarm-state reaction (IRQENx).
  - If you enable Alarm state for the channel (NCFTOEx), make sure the Alarm-state timer interval (TO) is less than the FOSU module's timer interval; otherwise, FOSU generates a chip reset every time a fault occurs on the channel. The FOSU timer interval (FOSU_COUNT) is chip-specific. See the chip-specific FCCU information.

NOTE

If you enable a noncritical fault channel but disable all reactions for that channel, FCCU changes state if necessary but doesn't perform any reaction because reactions are disabled. If you enable reactions for a noncritical fault channel but disable that channel, and FCCU is in Normal state when a fault occurs on the channel, FCCU doesn't enter Alarm or Fault state and therefore doesn't perform any reaction.

3.12.2 Recommendations to configure FCCU

1. After a power on, 'destructive', or long external reset, where both system and FCCU are reset, the following steps could be followed to configure FCCU:
   a. Check and clear any pending fault status
   b. Verify FCCU is in NORMAL state, else repeat step(a) above
   c. Configure FCCU
2. After a short external or any 'functional' reset of the system, arising out of a reset request from FCCU or other sources, the following steps could be followed to reconfigure FCCU:
   a. If active, wait for the Error out T_min to expire
   b. Check and clear fault status
   c. Error pin moves to "non faulty" state, once fault status is cleared and T_min expires

d. Verify FCCU is in NORMAL state, else repeat step(a) above
e. Read and verify value in FCCU_NCF_Ex
f. Reconfigure FCCU, if necessary

Appendix A
Release Notes for MPC5746RRMAD

A.1 Enhanced Time Processing Unit (eTPU) changes

A.1.1 Chip-specific eTPU2 Plus information changes

- In Register reset values, changed the following:
  - "eTPU System Configuration Registers_MCR" to eTPU_SCR_MCR.
  - "eTPU System Configuration Registers_ECR_ENG1" to eTPU_SCR_ECR_ENG1.
  - "eTPU System Configuration Registers_ECR_ENG2" to eTPU_SCR_ECR_ENG2.
  - "Time Base Registers_TBCR_ENG1" to eTPU_TBR_TBCR_ENG1.
  - "Time Base Registers_TBCR_ENG2" to eTPU_TBR_TBCR_ENG2.
  - "Time Base Registers_STAC_ENG1" to eTPU_TBR_STAC_ENG1.
  - "Time Base Registers_STAC_ENG2" to eTPU_TBR_STAC_ENG2.

A.1.2 eTPU module changes

- Changed the reset value of eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1) and eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2) (was 0000_0000h, is 2000_0000h).

- Editorial changes.
- In Table 2-3, changed the footnote to offsets 0x8000–0xBFFF and 0xC000–0xFFFF (was "Actual sizes of SCM and SDM are MCU-dependent (see Configuration section)", is "Actual sizes of SCM and SDM are MCU-dependent (see the chip-specific eTPU information)").
- In Table 2-4 :
  - Changed the footnote for offsets 0x8000-0xBFFF, 0xC000-0xFFFF, and 0x10000–0x1FFFF (was "The actual SDM size is MCU-dependent (see Configuration section)", is "The actual SDM size is MCU-dependent (see the chip-specific eTPU information). SDM is allocated from the starting value of the given range of offsets.").
  - Changed the footnote for offset 0x10000-1FFFF (was "The actual SCM size is MCU-dependent (see Configuration section). When the size is not the maximum, the unused SCM address range returns the value of the register ETPUSCMOFFDATAR.", is "The actual SCM size is MCU-dependent (see the chip-specific eTPU information). SCM is allocated from the starting value of the given range of offsets. When the size is not the maximum, the unused SCM address range returns the value of the register ETPUSCMOFFDATAR.")
- Changed "Configuration information" to "the chip-specific eTPU information" in:
  - The SCMSIZE field description of eTPU Module Configuration Register (eTPU_SCR_MCR)
  - The note and field descriptions of eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1)
  - The note and field descriptions of eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2)
- In eTPU Module Configuration Register (eTPU_SCR_MCR), revised the SCMSIZE reset value (is "See the chip-specific eTPU information for this field's reset value").

- Editorial changes (including the addition of cross-references for improved usability).
- Throughout the chapter, revised register names so that they are always consistent with the official register names as specified in these tables:
  - "eTPU_SCR memory map" in System Configuration memory map and registers
  - "eTPU_TBR memory map" in Timebase memory map and registers
  - "eTPU_ER memory map" in Engine memory map and registers
  - "eTPU_LMR memory map" in Latency Monitor memory map and registers
  - "eTPU_MESR memory map" in Memory Error Support memory map and registers
  - "eTPU_GCR memory map" in Global Channel memory map and registers
  - "eTPU_CCCR memory map" in Channel Configuration Control memory map and registers
  - "eTPU_SSIR memory map" in Shared Subsystem Integration memory map and registers

  So, for example, the chapter now consistently uses:
  - eTPU_MESR_CERDR instead of ETPUCERD
  - eTPU_GCR_CDTRER_ENG1 and eTPU_GCR_CDTRER_ENG2 instead of ETPUCDTRER
  - eTPU_TBR_STAC_ENG1 and eTPU_TBR_STAC_ENG2 instead of ETPUREDCR
- In eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1), revised the CIE field description.
- In eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1), in the DTRE field description, changed "This bit is mirrored from eTPU Channel Data Transfer Request Enable Register for Engine 2 (eTPU_GCR_CDTRER_ENG2)" to "This bit is mirrored from eTPU Channel Data Transfer Request Enable Register for Engine 1 (eTPU_GCR_CDTRER_ENG1)."
- In eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2), in the T4DIS field description, added "For more details, see Channel timing modes."
- In Table 2-34, deleted the footnote "See the eTPU integration guide...".
- In Latency Monitor, in the last bulleted list, changed "after each latency measurement, the value measured is compared with the MAXLAT value in the register ETPULMnMR; and if MAXLAT is not zero and the new measured value is greater." to "After each latency measurement, the value measured is compared with the LATENCY value in eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM*n*MR_ENG1) and eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM*n*MR_ENG2). If the new measured value is greater, it replaces the old value in eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM*n*MR_ENG1) and eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM*n*MR_ENG2)."
- In Using Worst-Case Latency estimates to evaluate performance, changed "Refer to Freescale Programming Note" to "See NXP Programming Note".
- In Execution and channel timing, deleted "For timings of eTPU block internal interface signals, consult the eTPU Integration Guide. For timings internal to the eTPU, refer to the eTPU Creation Guide."

A.2 Fault Collection and Control Unit (FCCU) changes

A.2.1 Chip-specific FCCU information changes

- In the FCCU chip-specific register reset values table, changed the reset value of FCCU_CFG to "See FCCU_CFG register bit value sources (N and C) by event."
- Added the FCCU_CFG register event bit values by source (N and C) section/table.
- In the FCCU failure inputs from other modules section, added the following sentence to the second "Note" in this section: "FCCU Fault Channels 22, 23, and 24 may be asserted after reset."
- In Table 3-3, added the following note in the Description column of NCF[17], NCF[19], and NCF[21]: "This input must be mapped only to reset, and must not be mapped to an alarm."

A.2.2 FCCU module changes

• Made the following changes to the Configuration (FCCU_CFG) register description:
  • Added the FOP field, which was previously marked as reserved.
  • Added the FOPE field, which was previously marked as reserved.

• In the Configuration (FCCU_CFG) register description, at the end of the FOP field description, added a Note: *Writing to FCCU_CFG sets FOP to 0.*
• In the Mode Controller Status (FCCU_MCS) register description, changed the footnote for the register reset value from *The MC_ME module changes the value of the register immediately after the chip leaves RESET mode.* to *The reset value is chip-specific. See the chip-specific FCCU information. (The MC_ME module changes the value of the register immediately after the chip leaves RESET mode.)*.
• In the NA Freeze Status (FCCU_N2AF_STATUS) register description, in the NAFS field description, deleted this note: *This field can be read and written but writing 1 has no effect.* • In the AF Freeze Status (FCCU_A2FF_STATUS) register description, changed the access types of the AF_SRC and AFFS fields from read-write to read-only. In the AF_SRC and AFFS field descriptions, deleted this note: *This field can be read and written but writing 1 has no effect.* • In the NF Freeze Status (FCCU_N2FF_STATUS) register description, changed the access types of the NF_SRC and NFFS fields from read-write to read-only. In the NF_SRC and NFFS field descriptions, deleted this note: *This field can be read and written but writing 1 has no effect.* • In the FA Freeze Status (FCCU_F2A_STATUS) register description, in the FAFS field description, deleted this note: *This field can be read and written but writing 1 has no effect.*
• In the Noncritical Fault State Configuration (FCCU_NCFS_CFG*n*) and NMI Enable (FCCU_NMI_EN*n*) register descriptions, removed the following inaccurate note: *Do not configure the same channel for both a RESET reaction in the FCCU_NCFS_CFGn register and an NMI reaction in the FCCU_NMI_ENn register at the same time.*
• In Use cases and limitations, rewrote Configuration guidelines (formerly titled "Misconfigurations") to improve the accuracy and clarity of the material by putting it in terms of guidelines you must follow. Added guidelines about what you must do if you enable chip reset as the type of Fault-state reaction for a noncritical fault channel and if you enable Alarm state for the channel.
• In Fault recovery, added the word *noncritical* before the word *FAULT*.
• In FCCU Output Supervision Unit, deleted *Also, the case of a continuously incoming disabled fault being enabled later, is not monitored*. Also deleted *All intermediate faults are masked (not monitored) during this timeout cycle*.
• In the PhysicErrorPin field description, added the spelled-out name of the field, Fault Output (EOUT) States. Replaced *PhysicErrorPin : Physical Error Pin's state. PhysicErrorPin[0] corresponds to error pin 0 and PhysicErrorPin[1] corresponds to error pin 1. For each of these, the bit values have the following meanings: 0: Error Pin is at logical 0; 1: Error Pin is at logical 1* with *Applies only after successfully running an OP3 operation (OPR) and only when the EOUT signals are active (FCCU_SET_AFTER_RESET). Indicates the states that FCCU is driving on the EOUT signals.* Deleted *During Fault state, a static or toggling value is observed based on selected protocol.* Replaced the list of two values and meanings with a list of four values and meanings.
• In the Mode Controller Status (FCCU_MCS) register description, changed the general description from *and whether FCCU was in Fault state when FCCU captured each mode* to *and whether FCCU was in Fault state and NMIOUT was asserted (as the result of a noncritical fault) when FCCU captured each mode*. In the FS0 field description (and similarly in the FS1, FS2, and FS3 field descriptions), changed *Indicates whether FCCU was in Fault state when FCCU captured MCS0* to *Indicates whether FCCU was in Fault state and NMIOUT was asserted (as the result of a noncritical fault) when FCCU captured MCS0*; changed the meaning for the value 0 from *A state other than Fault state* to *Not in Fault state or NMIOUT not asserted*; changed the meaning for the value 1 from *Fault state* to *In Fault state and NMIOUT asserted*.
• In the Permanent Configuration Lock (FCCU_PERMNT_LOCK) register description, changed the register name from *Permanent Lock* to *Permanent Configuration Lock*; in the general description, changed *The FCCU_PERMNT_LOCK register is used for locking configuration permanently by writing FFh. This register is available only in supervisor mode. The permanent lock can only be removed by applying a reset (not necessarily power on reset). This register is write only.* to *Writable only by code running in Supervisor mode. Permanently locks the configuration, which prevents FCCU from entering Configuration state until FCCU is reset.*; changed the access type of the reserved field from Read-Only Zero to Reserved, Unimplemented. In the PERMNTKEY field description, changed *Transition Locking Key*

value to *Permanent Configuration Lock—Writable only by code running in Supervisor mode. Permanently locks the configuration, which prevents FCCU from entering Configuration state until FCCU is reset.*; changed the meaning for the value FFh from *Configuration Locked* to *Lock.*; changed the meaning for the value *Any other value* from *No Change* to *Do nothing.* • In the Transient Configuration Lock (FCCU_TRANS_LOCK) register description, changed the register name from *Transient Lock* to *Transient Configuration Lock*; in the general description, changed *The FCCU_TRANS register is used for unlocking configuration by writing BCh. This register is available only in supervisor mode. This register is write only.* to *Writable only by code running in Supervisor mode. Temporarily locks and unlocks the configuration. Locking the configuration prevents FCCU from entering Configuration state.*; changed the access type of the reserved field from Read-Only Zero to Reserved, Unimplemented. In the TRANSKEY field description, changed *Transition Unlocking Key value* to *Transient Configuration Lock—Writable only by code running in Supervisor mode. Temporarily locks and unlocks the configuration. Locking the configuration prevents FCCU from entering Configuration state.*; changed the meaning for the value BCh from *Configuration Unlocked* to *Unlock.*; changed the meaning for the value *Any other value* from *FCCU gets transiently locked again.* to *Lock.*
• In the Configuration (FCCU_CFG) register description, in the FCCU_SET_CLEAR field description, added the field name, Fault-Output (EOUT) Control; rewrote the function description and value meanings.
• In the FCCU_SET_AFTER_RESET field description, added the spelled-out name of the field, Fault-Output (EOUT) Activate; rewrote the general description; rewrote the meanings for both values.

• In the Configuration (FCCU_CFG) register description, rewrote the RCCE1 andRCCE0 field descriptions, including their spelled-out names, for clarity.
• In the Delta T (FCCU_DELTA_T) register description, in the DELTA_T field description, added the spelled-out name of the field, Bistable Minimum Fault-Output (EOUT) Faulty Interval; added the following sentence: *Applies only to Bistable fault-output mode (FOM)*; changed *EOUT fault-indication timer* to *fault-output (EOUT) timer*; moved the information that is after the equation to EOUT interface because it's general and not only applicable to DELTA_T.
• Inserted the following new sections in EOUT interface prior to Dual-rail protocol :
  • Introduction. Moved the first paragraph from the previous EOUT interface section into this section. In the moved information, replaced *The FCCU provides up to two bidirectional signals (EOUT[1:0]) as a failure indication to the external world.* with *You use the EOUT[1:0] signals to indicate FCCU's condition to off-chip logic.*
  • The FCCU conditions. Added new content.
  • How the fault-output (EOUT) timer works. Moved general information from the DELTA_T field description to this section. In the moved information, changed *EOUT fault-indication timer* to *fault-output (EOUT) timer*. In the first paragraph of the moved information, added the following new list item to the beginning of the list: *The fault-output (EOUT) signals are not programmed to be always low (FCCU_SET_CLEAR)*. In the third paragraph of the moved information, changed the first list item from *T_min has expired* to *If the EOUT signals are in Bistable fault-output mode (FOM), T_min has expired*. In the second paragraph, deleted *or not applicable*.
  • Prepare the EOUT signals to indicate FCCU's condition. Added new content.
  • More about the EOUT interface. In one of the notes, changed *after software has enabled the EOUT signals (FCCU_CFG[FCCU_SET_AFTER_RESET]* to *when the EOUT signals are active (FCCU_SET_AFTER_RESET)*.
• Revised the Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN*n*) register description for clarity.

• In Signals, rewrote the description of the EOUT[1:0] signal function for accuracy and clarity. • In the figure in Block diagram, changed the double-headed arrow between *EOUT1 intf* and *EOUT[1]* to a single-headed arrow that indicates that EOUT[1] is an output only.
• In the figure in FCCU Output Supervision Unit, removed the line that connects an FCCU fault output to a FOSU input; added a line that connects the FCCU FIF output to a FOSU input. • In Signals, added a signal description for the FIF signal.
• In the Status (FCCU_STAT) register description, rewrote the ESTAT and STATUS field descriptions for clarity.
• In the IO Control (FCCU_EINOUT) register description, for the EIN1 and EIN0 fields, changed the reset value to indeterminate (x) and rewrote the field descriptions for clarity. • In Signals, added a table row for the EIN[1:0] signals.

• Added the following sections to clarify how to perform certain procedures:
  • Put FCCU in Configuration or Normal state
  • Run operations
  • Lock and unlock the configuration

• In Acronyms and abbreviations, added a row for *NVM*.
In Introduction :

• Replaced the list item that begins with *The FCCU's configuration is lockable* with *Lockable configuration*. • Added the list item *Nonvolatile memory interface (NVMCFG signals) for configuration loading*.
• In Main features, added the list item *Nonvolatile memory interface (NVMCFG signals) for configuration loading*.
• Added FCCU_CFG register bit value sources (N and C) by event.
• Added Configuration registers.
• In Signals, added table rows for the NVMCFG[21:0] and SSCMDONE signals.
• In the Configuration-State Timer Interval (FCCU_CFG_TO) register description, changed its name to *Configuration-State Timer Interval*, and rewrote the description of the register's function and the description of the TO field's function.

• In the Control (FCCU_CTRL) register description:
  • Rewrote the description of the register's function.
  • Rewrote the description of the OPR field.

• Added Nonvolatile memory interface.
• In Introduction, changed *They show operational state only on software request* to *They show operational state only after configuration by software*.
• In More about the EOUT interface, deleted this paragraph: *These resets influence the state of the failure indication pins…Other resets should not influence the state of the failure indication pins*.
• In the Delta T (FCCU_DELTA_T) register description, rewrote the note in the DELTA_T field description to clarify that the variance of the DELTA_T durations is chip-specific.
• In More about the EOUT interface, changed *NA (not applicable) appears in the following table* to *NA (not applicable) appears in the following table*.
• In the Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN*n*) register description, rewrote the register's function description and the EOUTENx field's function description.
• In Signals, revised the function description for the FIF signal.
• In NMI/WKPU interface, changed *another mode* to *another mode change*.

• Throughout the chapter, changed *non-critical* to *noncritical*.
• In each of the following register descriptions, changed the format of the table in the register function summary and reworded the sentence in the field function summary that refers to the table:
  • Noncritical Fault Configuration (FCCU_NCF_CFG*n*)
  • Noncritical Fault Status (FCCU_NCF_S*n*)
  • Noncritical Fault Enable (FCCU_NCF_E*n*)
  • Noncritical Fault Timeout Enable (FCCU_NCF_TOE*n*)
  • IRQ Alarm Enable (FCCU_IRQ_ALARM_EN*n*)
  • Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN*n*) (no sentence rewording)

• In Introduction, changed *One of the error out pins is high* to *One of the fault-output (EOUT) signals is high*.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, C-5, CodeTest, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and µVision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

©
2016
NXP B.V.

Document Number MPC5746RRMAD
Revision 2, 10/2016

MPC5746R Reference Manual

Document Number: MPC5746RRM
Rev. 6, 03/2016

Contents

Section number **Title** **Page**

Chapter 1
Preface

1.1 Overview....91

1.2 Audience....91

1.3 Content....91

1.4 Document organization....92

1.5 Register description conventions....92

1.6 Special terms....93

1.7 Acronyms and abbreviations....93

1.8 References....95

Chapter 2
Introduction

2.1 Introduction....97

2.2 Target applications....97

2.3 Block diagrams....97

2.4 Overall architecture....100

2.5 Core features....100

2.6 Memory hierarchy....100

2.7 Features summary....101

2.8 Feature list....102

2.9 Packaging....103

2.10 Software debug and calibration....104

Chapter 3
Embedded memories

3.1 Overview....105

3.2 SRAM....105

3.2.1 System SRAM....105

3.2.2 Standby SRAM Regulator....106

3.2.3 Overlay SRAM....107
3.2.4 Processor core local SRAM....108
3.3 Flash memory....109
3.4 Flash memory controller....110
3.5 Flash memory array....110
3.5.1 Features....110
3.5.2 Flash organization....111
3.5.3 UTEST memory space....112
3.6 End-to-end Error Correction Code (e2eECC)....112

Chapter 4
Signal Multiplexing and Signal Description

4.1 Generic pins/balls....117
4.1.1 System pin functions....117

Chapter 5
Memory Map

5.1 Introduction....119

Chapter 6
Device Configuration Format (DCF) Records

6.1 Overview....121
6.2 DCF records in boot sequence....122
6.3 DCF records function....122
6.4 DCF records location....122
6.5 DCF record structure....123
6.6 DCF records sequence....124
6.7 DCF clients....126
6.7.1 Miscellaneous DCF clients....136

Chapter 7
e200z4d Core Complex Overview

7.1 Introduction....149
7.2 Overview of the e200z425n3 and e200z424 cores....149

7.3 Features....150
7.4 Microarchitecture summary....151
7.4.1 Instruction unit features....153
7.4.2 Integer unit features....153
7.4.3 Load/Store unit features....154
7.4.4 MPU features....154
7.4.5 Cache features....154
7.4.6 Local memory features....155
7.4.7 e200z425n3 system bus features....155
7.4.8 e200z424 system features....155

Chapter 8
Core description

8.1 Overview of the e200z425n3 core....157
8.2 Register model....158
8.3 Dual-issue operation....162
8.4 Instruction timing....163
8.5 Reservation instructions and cache interactions....174
8.6 Signal Processing Extension/Embedded Floating-point Status and Control Register (SPEFSCR)....175
8.7 Cache....177
8.7.1 Cache overview....177
8.7.2 Cache organization....178
8.7.3 L1 Cache Control and Status Register 0 (L1CSR0)....178
8.7.4 L1 Cache Control and Status Register 1 (L1CSR1)....179
8.7.5 L1 Cache Configuration Register 0 (L1CFG0)....181
8.7.6 L1 Cache Configuration Register 1 (L1CFG1)....182
8.7.7 Cache Invalidate by Set and Way....183
8.7.8 Cache EDC/ECC Parity Protection....184
8.7.9 Cache Error Injection....186

8.8 Exceptions....187
8.8.1 Exception Syndrome Register (ESR)....188
8.8.2 Machine State Register (MSR)....189
8.8.3 Machine Check Syndrome Register (MCSR)....190
8.8.4 Interrupt Vector Prefix Registers (IVPR)....194
8.8.5 Interrupt Definitions....194
8.9 Memory Protection Unit (MPU)....205
8.9.1 MPU Overview....205
8.9.2 Software Interface and MPU Instructions....205
8.9.3 MPU Read Entry Instruction (mpure)....205
8.9.4 MPU Write Entry Instruction (mpuwe)....206
8.9.5 MPU Synchronize Instruction (mpusync)....206
8.9.6 MMU/MPU Configuration Register (MMUCFG)....207
8.9.7 MPU0 Configuration Register (MPU0CFG)....208
8.9.8 MPU0 Control and Status Register 0 (MPU0CSR0)....209
8.9.9 MPU Assist Registers (MAS)....211
8.9.10 MAS Registers Summary....217
8.10 Local memories....217
8.10.1 Local Instruction and Data Memory Overview....217
8.10.2 Local memory control and configuration....218
8.11 End-to-End ECC Support....233
8.11.1 End-to-End ECC control and configuration....233

Chapter 9
System Integration Unit Lite2 (SIUL2)

9.1 System Integration Unit Lite2 (SIUL2) chip-specific module information....237
9.1.1 SIUL2 interrupts....237
9.1.2 Reset values of MIDRn....237
9.2 Introduction....238
9.2.1 Overview....238

9.2.2 Features....239
9.2.3 Register protection....241
9.3 Memory map and register description....241
9.3.1 Memory map....241
9.3.2 SIUL2 MCU ID Register #1 (SIUL2_MIDR1)....275
9.3.3 SIUL2 MCU ID Register #2 (SIUL2_MIDR2)....277
9.3.4 SIUL2 DMA/Interrupt Status Flag Register0 (SIUL2_DISR0)....278
9.3.5 SIUL2 DMA/Interrupt Request Enable Register0 (SIUL2_DIRER0)....280
9.3.6 SIUL2 DMA/Interrupt Request Select Register0 (SIUL2_DIRSR0)....282
9.3.7 SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)....284
9.3.8 SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)....285
9.3.9 SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)....287
9.3.10 SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR*n*)....289
9.3.11 SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)....289
9.3.12 SoC Configuration Register 0 (SIUL2_SCR0*n*)....290
9.3.13 SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR*n*)....291
9.3.14 SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR*n*)....294
9.3.15 General Purpose Register 0 (SIUL2_GPR0)....295
9.3.16 General Purpose Register 1 (SIUL2_GPR1)....297
9.3.17 General Purpose Register 2 (SIUL2_GPR2)....299
9.3.18 SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO*n*)....300
9.3.19 SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI*n*)....301
9.3.20 SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO*n*)....302
9.3.21 SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI*n*)....302
9.3.22 SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO*n*)....303
9.4 Functional description....304
9.4.1 General....304
9.4.2 Pad Control....304
9.4.3 General Purpose Input and Output Pads (GPIO)....305

9.4.4 External Interrupts/DMA Requests (REQ Pins).....306

Chapter 10
Crossbar Switch (AXBS)

10.1 Crossbar switch configuration.....311
10.1.1 Computational shell crossbar switch configuration.....311
10.1.2 Peripheral Shell Crossbar Switch Configuration.....315
10.2 Introduction.....317
10.2.1 Features.....317
10.3 Memory Map / Register Definition.....317
10.3.1 Priority Registers Slave (XBAR_PRS*n*).....319
10.3.2 Control Register (XBAR_CRS*n*).....321
10.3.3 Master General Purpose Control Register (XBAR_MGPCR*n*).....323
10.4 Functional Description.....324
10.4.1 General operation.....324
10.4.2 Register coherency.....325
10.4.3 Arbitration.....325
10.5 Initialization/application information.....328

Chapter 11
Crossbar Integrity Checker (XBIC)

11.1 Overview.....329
11.2 Features.....329
11.3 Block diagram.....329
11.4 External signal description.....330
11.5 Memory map and register definition.....330
11.5.1 XBIC Module Control Register (XBIC_MCR).....331
11.5.2 XBIC Error Injection Register (XBIC_EIR).....333
11.5.3 XBIC Error Status Register (XBIC_ESR).....333
11.5.4 XBIC Error Address Register (XBIC_EAR).....336
11.6 Functional description.....337

Chapter 12
Peripheral Bridge (AIPS-Lite)

12.1 Peripheral Bridge chip-specific module information....339
12.1.1 Number of peripheral bridges....339
12.1.2 Memory maps....339
12.1.3 MPRA registers....339
12.1.4 PACR registers....340
12.2 Introduction....340
12.2.1 Features....340
12.2.2 General operation....340
12.3 Memory map/register definition....341
12.3.1 Master Privilege Register A (AIPS_MPRA)....343
12.3.2 Peripheral Access Control Register (AIPS_PACR*n*)....346
12.3.3 Off-Platform Peripheral Access Control Register (AIPS_OPACR*n*)....351
12.4 Functional description....357
12.4.1 Access support....357

Chapter 13
System Memory Protection Unit (SMPU)

13.1 System Memory Protection Unit (SMPU) chip-specific module information....359
13.1.1 SMPUx configuration....359
13.2 Overview....360
13.2.1 Block diagram....360
13.2.2 Features....361
13.3 Memory map/register definition....362
13.3.1 Control/Error Status Register 0 (SMPU_CESR0)....365
13.3.2 Control/Error Status Register 1 (SMPU_CESR1)....366
13.3.3 Error Address Register, Bus Master n (SMPU_EAR*n*)....367
13.3.4 Error Detail Register, Bus Master n (SMPU_EDR*n*)....367
13.3.5 Region Descriptor n, Word 0 (SMPU_RGD*n*_WORD0)....368

13.3.6 Region Descriptor n, Word 1 (SMPU_RGD*n*_WORD1)....369
13.3.7 Region Descriptor n, Word 2 (SMPU_RGD*n*_WORD2)....369
13.3.8 Region Descriptor n, Word 3 (SMPU_RGD*n*_WORD3)....371
13.4 Functional description....372
13.4.1 Access evaluation macro....372
13.4.2 Final evaluation and error terminations....373
13.5 Initialization information....374
13.6 Application information....375

Chapter 14
Intelligent AHB Gasket (IAHBG)

14.1 IAHBG chip-specific module information....377
14.1.1 IAHBG implemented operation modes....377
14.1.2 IAHBG clock selection ratios....377
14.1.3 IAHBG features enabled by Platform Configuration Module (PCM)....377
14.2 Introduction....378
14.3 Timing modes....378
14.3.1 1:1....378
14.3.2 2:1....378
14.3.3 1:2....379

Chapter 15
Platform Configuration Module (PCM)

15.1 PCM memory map....381
15.1.1 FEC Burst Optimization Master Control Register (PCM_FBOMCR)....382
15.1.2 Bus Bridge Configuration Register 1 (PCM_IAHB_BE1)....383
15.1.3 Bus Bridge Configuration Register 2 (PCM_IAHB_BE2)....385
15.1.4 IPS Clock Gating Module Enable 0 Register (PCM_IPS_CGM_EN0)....386

Chapter 16
Semaphores2 (SEMA42)

16.1 Introduction....389
16.1.1 Features....389

16.2 Memory map/register definition....391
16.2.1 Gate Register (SEMA42_GATE*n*)....391
16.2.2 Reset Gate Write (SEMA42_RSTGT_W)....392
16.2.3 Reset Gate Read (SEMA42_RSTGT_R)....394
16.3 Functional description....395

Chapter 17
Interrupt Controller (INTC)

17.1 Interrupt Controller (INTC) chip-specific module information....397
17.1.1 INTC configurable parameters....397
17.1.2 Interrupt sources....397
17.2 Introduction....410
17.3 Block diagram....411
17.4 Features....412
17.5 Modes of operation....413
17.5.1 Software vector mode....414
17.5.2 Hardware vector mode....414
17.6 Memory map and register definition....415
17.6.1 INTC Block Configuration Register (INTC_BCR)....444
17.6.2 INTC Master Protection Register (INTC_MPROT)....446
17.6.3 INTC Current Priority Register for Processor 0 (INTC_CPR0)....447
17.6.4 INTC Current Priority Register for Processor 1 (INTC_CPR1)....448
17.6.5 INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)....449
17.6.6 INTC Interrupt Acknowledge Register for Processor 1 (INTC_IACKR1)....450
17.6.7 INTC End Of Interrupt Register for Processor 0 (INTC_EOIR0)....451
17.6.8 INTC End Of Interrupt Register for Processor 1 (INTC_EOIR1)....451
17.6.9 INTC Software Set/Clear Interrupt Register (INTC_SSCIR*n*)....452
17.6.10 INTC Priority Select Register (INTC_PSR*n*)....454
17.6.11 INTC Monitor Mode Register 0 (INTC_MMRC0)....456
17.6.12 INTC HIPRI Register (INTC_HIPRI*n*C0)....457

17.6.13 INTC LAT Register (INTC_LAT*n*C0)....457
17.6.14 INTC Timer Register (INTC_TIMER*n*C0)....458
17.6.15 INTC Monitor Mode Register 1 (INTC_MMRC1)....458
17.6.16 INTC HIPRI Register (INTC_HIPRI*n*C1)....459
17.6.17 INTC LAT Register (INTC_LAT*n*C1)....459
17.6.18 INTC Timer Register (INTC_TIMER*n*C1)....460
17.7 Functional description....460
17.7.1 Interrupt request sources....460
17.7.2 Priority management....462
17.7.3 Handshaking with processor....464
17.7.4 Interrupt Controller Monitor (INTCM)....467
17.8 Initialization/application information....468
17.8.1 Initialization flow....468
17.8.2 Interrupt exception handler....469
17.8.3 ISR, RTOS, and task hierarchy....470
17.8.4 Order of execution....471
17.8.5 Priority ceiling protocol....472
17.8.6 Selecting priorities according to request rates and deadlines....475
17.8.7 Software-settable interrupt requests....476
17.8.8 Lowering priority within an ISR....477
17.8.9 Negating an interrupt request outside of its ISR....478
17.8.10 Examining LIFO contents....478
17.9 Interrupt sources....479

Chapter 18
Enhanced Direct Memory Access (eDMA)

18.1 Introduction....481
18.2 Overview....482
18.3 Features....482
18.4 Memory map/register definition....488

18.5 Memory Map and Registers....491
18.5.1 Control Register (DMA_CR)....520
18.5.2 Error Status Register (DMA_ES)....523
18.5.3 Enable Request Register High (DMA_ERQH)....526
18.5.4 Enable Request Register Low (DMA_ERQL)....530
18.5.5 Enable Error Interrupt Register High (DMA_EEIH)....533
18.5.6 Enable Error Interrupt Register Low (DMA_EEIL)....537
18.5.7 Set Enable Request Register (DMA_SERQ)....540
18.5.8 Clear Enable Request Register (DMA_CERQ)....541
18.5.9 Set Enable Error Interrupt Register (DMA_SEEI)....542
18.5.10 Clear Enable Error Interrupt Register (DMA_CEEI)....543
18.5.11 Clear Interrupt Request Register (DMA_CINT)....544
18.5.12 Clear Error Register (DMA_CERR)....545
18.5.13 Set START Bit Register (DMA_SSRT)....546
18.5.14 Clear DONE Status Bit Register (DMA_CDNE)....547
18.5.15 Interrupt Request Register High (DMA_INTH)....547
18.5.16 Interrupt Request Register Low (DMA_INTL)....551
18.5.17 Enable Error Interrupt Register High (DMA_ERRH)....555
18.5.18 Enable Error Interrupt RegisterLow (DMA_ERRL)....559
18.5.19 Hardware Request Status Register High (DMA_HRSH)....563
18.5.20 Hardware Request Status Register Low (DMA_HRSL)....566
18.5.21 Channel n Priority Register (DMA_DCHPRI*n*)....569
18.5.22 Channel n Master ID Register (DMA_DCHMID*n*)....571
18.5.23 TCD Source Address (DMA_TCD*n*_WORD_0)....572
18.5.24 TCD Transfer Attributes (DMA_TCD*n*_WORD_1)....573
18.5.25 TCD Number of bytes to be transferred (DMA_TCD*n*_WORD_2_NBYTES)....574
18.5.26 TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD*n*_WORD_2_MLOOP)575
18.5.27 TCD Last Source Address Adjustment (DMA_TCD*n*_WORD_3)....576
18.5.28 TCD Destination Address (DMA_TCD*n*_WORD_4)....576

18.5.29 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_WORD_5)....577
18.5.30 TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD*n*_WORD_6)....578
18.5.31 TCD Control and Status (DMA_TCD*n*_WORD_7)....579
18.6 Functional description....582
18.6.1 eDMA microarchitecture....582
18.6.2 eDMA basic data flow....583
18.6.3 eDMA performance....587
18.7 Initialization/application information....591
18.7.1 eDMA initialization....591
18.7.2 eDMA programming errors....592
18.7.3 eDMA arbitration mode considerations....593
18.7.4 Transfer....595
18.7.5 TCD status....598
18.7.6 Channel linking....600
18.7.7 Dynamic programming....601

Chapter 19
Direct Memory Access Multiplexer (DMACHMUX)

19.1 DMACHMUX chip-specific module information....607
19.1.1 DMA channel assignment....607
19.1.2 DMA channel multiplexing block diagram....607
19.1.3 DMA request mapping....608
19.1.4 DMAMUX Memory map....611
19.2 Introduction....611
19.2.1 Overview....611
19.2.2 Features....612
19.2.3 Modes of operation....612
19.3 External signal description....613

19.4 Memory map/register definition....613
19.4.1 Channel Configuration register (DMAMUX_CHCFG*n*)....614
19.5 Functional description....615
19.5.1 DMA channels with periodic triggering capability....615
19.5.2 DMA channels with no triggering capability....617
19.5.3 Always-enabled DMA sources....618
19.6 Initialization/application information....619
19.6.1 Reset....619
19.6.2 Enabling and configuring sources....619

Chapter 20
Wakeup Unit (WKPU)

20.1 Introduction....623
20.1.1 Features....624
20.2 External Signal Description....624
20.3 WKPU memory map and registers....624
20.3.1 NMI Status Flag Register (WKPU_NSR)....625
20.3.2 NMI Configuration Register (WKPU_NCR)....627
20.4 Functional Description....629
20.4.1 Non-Maskable Interrupts....629
20.4.2 System Wakeup....629
20.4.3 Glitch Filter....630
20.5 Initialization Information....630
20.5.1 Glitch Filter and Pad Configuration....630
20.5.2 Non-Maskable Interrupts....631

Chapter 21
Error Injection Module (EIM)

21.1 Error Injection Module (EIM) chip-specific module information....633
21.2 Introduction....633
21.2.1 Overview....633

21.2.2 Features....634
21.3 Memory map and register definition....635
21.3.1 Error Injection Module Configuration Register (EIM_EIMCR)....636
21.3.2 Error Injection Channel Enable register (EIM_EICHEN)....637
21.3.3 Error Injection Channel Descriptor, Word0 (EIM_EICHD*n*_WORD0)....638
21.3.4 Error Injection Channel Descriptor, Word1 (EIM_EICHD*n*_WORD1)....639
21.3.5 Error Injection Channel Descriptor, Word2 (EIM_EICHD*n*_WORD2)....639
21.4 Functional description....640

Chapter 22
Clocking

22.1 Introduction....641
22.2 Clock generation....642
22.2.1 MC_CGM registers....644
22.3 System clock frequency limitations....645
22.3.1 JTAG frequencies....646
22.4 Default clock configuration....647
22.5 Clock sources....647
22.5.1 PLL....647
22.5.2 External oscillator (XOSC)....652
22.5.3 Internal 16 MHz RC Oscillator (IRCOSC)....653
22.6 Peripheral Clocks....654
22.6.1 LFAST clocking....655
22.6.2 Ethernet clocking....656
22.6.3 Sigma-Delta ADC clocking....657
22.6.4 LINFlex clocking....658
22.7 Clock monitoring....659
22.7.1 CMU configuration....659
22.7.2 External oscillator (XOSC) monitor....662
22.7.3 Internal RC oscillator (IRCOSC) monitor....662

22.8 Loss of system clock behavior....662
22.8.1 Loss of PLL or XOSC....663
22.8.2 Loss of IRCOSC....663
22.8.3 PLL loss of lock interrupt....663
22.9 Progressive clock switching....664

Chapter 23
Dual PLL Digital Interface (PLLDIG)

23.1 Introduction....665
23.2 Block Diagram....665
23.3 Features....665
23.4 Modes of operation....666
23.4.1 Bypass mode with reference, both PLLs disabled....666
23.4.2 Normal mode with reference, PLL0 or both PLLs enabled....666
23.5 Memory map and register definition....667
23.5.1 PLLDIG PLL0 Control Register (PLLDIG_PLL0CR)....668
23.5.2 PLLDIG PLL0 Status Register (PLLDIG_PLL0SR)....670
23.5.3 PLLDIG PLL0 Divider Register (PLLDIG_PLL0DV)....672
23.5.4 PLLDIG PLL1 Control Register (PLLDIG_PLL1CR)....674
23.5.5 PLLDIG PLL1 Status Register (PLLDIG_PLL1SR)....676
23.5.6 PLLDIG PLL1 Divider Register (PLLDIG_PLL1DV)....677
23.5.7 PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)....679
23.5.8 PLLDIG PLL1 Fractional Divide Register (PLLDIG_PLL1FD)....681
23.6 Register classification for safety requirements....682
23.7 Functional description....682
23.7.1 Input clock frequency....682
23.7.2 Clock configuration....682
23.7.3 Frequency modulation....684
23.8 Maximum lock time....686
23.9 Initialization information....686

Chapter 24
Clock Monitor Unit (CMU)

24.1 Introduction....687
24.1.1 Main features....688
24.2 Block diagram....688
24.3 Signals....688
24.4 Register description and memory map....689
24.4.1 CMU Control Status Register (CMU_CSR)....690
24.4.2 CMU Frequency Display Register (CMU_FDR)....691
24.4.3 CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)....692
24.4.4 CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)....692
24.4.5 CMU Interrupt Status Register (CMU_ISR)....693
24.4.6 CMU Measurement Duration Register (CMU_MDR)....695
24.5 Functional description....695
24.5.1 Frequency meter....695
24.5.2 CLKMN0_RMT supervisor....696
24.5.3 CLKMN1 supervisor....696

Chapter 25
Clock Generation Module (MC_CGM)

25.1 Introduction....699
25.1.1 Overview....699
25.1.2 Features....700
25.2 External Signal Description....701
25.3 MC_CGM Registers....701
25.3.1 PCS Switch Duration (MC_CGM_PCS_SDUR)....704
25.3.2 PCS Divider Change for external crystal oscillator (MC_CGM_PCS_DIVC1)....704
25.3.3 PCS Divider End for external crystal oscillator (MC_CGM_PCS_DIVE1)....705
25.3.4 PCS Divider Start for external crystal oscillator (MC_CGM_PCS_DIVS1)....705
25.3.5 PCS Divider Change for PLL0 PHI (MC_CGM_PCS_DIVC2)....706

25.3.6 PCS Divider End for PLL0 PHI (MC_CGM_PCS_DIVE2)....707
25.3.7 PCS Divider Start for PLL0 PHI (MC_CGM_PCS_DIVS2)....707
25.3.8 PCS Divider Change for PLL1 PHI (MC_CGM_PCS_DIVC4)....708
25.3.9 PCS Divider End for PLL1 PHI (MC_CGM_PCS_DIVE4)....708
25.3.10 PCS Divider Start for PLL1 PHI (MC_CGM_PCS_DIVS4)....709
25.3.11 System Clock Divider Ratio Change Register (MC_CGM_SC_DIV_RC)....710
25.3.12 Divider Update Type (MC_CGM_DIV_UPD_TYPE)....711
25.3.13 Divider Update Trigger (MC_CGM_DIV_UPD_TRIG)....712
25.3.14 Divider Update Status (MC_CGM_DIV_UPD_STAT)....713
25.3.15 System Clock Select Status (MC_CGM_SC_SS)....715
25.3.16 System Clock Divider Configuration 0 (MC_CGM_SC_DC0)....716
25.3.17 System Clock Divider Configuration 1 (MC_CGM_SC_DC1)....717
25.3.18 System Clock Divider Configuration 2 (MC_CGM_SC_DC2)....717
25.3.19 Aux Clock 0 Select Control (MC_CGM_AC0_SC)....718
25.3.20 Aux Clock 0 Select Status (MC_CGM_AC0_SS)....719
25.3.21 Aux Clock 0 Divider Configuration 0 (MC_CGM_AC0_DC0)....720
25.3.22 Aux Clock 0 Divider Configuration 1 (MC_CGM_AC0_DC1)....721
25.3.23 Aux Clock 0 Divider Configuration 2 (MC_CGM_AC0_DC2)....721
25.3.24 Aux Clock 0 Divider Configuration 3 (MC_CGM_AC0_DC3)....722
25.3.25 Aux Clock 0 Divider Configuration 4 (MC_CGM_AC0_DC4)....723
25.3.26 Aux Clock 1 Select Control (MC_CGM_AC1_SC)....724
25.3.27 Aux Clock 1 Select Status (MC_CGM_AC1_SS)....725
25.3.28 Aux Clock 1 Divider Configuration 0 (MC_CGM_AC1_DC0)....726
25.3.29 Aux Clock 2 Select Control (MC_CGM_AC2_SC)....726
25.3.30 Aux Clock 2 Select Status (MC_CGM_AC2_SS)....727
25.3.31 Aux Clock 2 Divider Configuration 0 (MC_CGM_AC2_DC0)....728
25.3.32 Aux Clock 3 Select Control (MC_CGM_AC3_SC)....729
25.3.33 Aux Clock 3 Select Status (MC_CGM_AC3_SS)....730
25.3.34 Aux Clock 4 Select Control (MC_CGM_AC4_SC)....731

25.3.35 Aux Clock 4 Select Status (MC_CGM_AC4_SS)....732
25.3.36 Aux Clock 5 Select Control (MC_CGM_AC5_SC)....732
25.3.37 Aux Clock 5 Select Status (MC_CGM_AC5_SS)....733
25.3.38 Aux Clock 5 Divider Configuration 0 (MC_CGM_AC5_DC0)....734
25.3.39 Aux Clock 5 Divider Configuration 1 (MC_CGM_AC5_DC1)....735
25.3.40 Aux Clock 6 Select Control (MC_CGM_AC6_SC)....736
25.3.41 Aux Clock 6 Select Status (MC_CGM_AC6_SS)....737
25.3.42 Aux Clock 6 Divider Configuration 0 (MC_CGM_AC6_DC0)....737
25.3.43 Aux Clock 8 Select Control (MC_CGM_AC8_SC)....738
25.3.44 Aux Clock 8 Select Status (MC_CGM_AC8_SS)....739
25.3.45 Aux Clock 8 Divider Configuration 0 (MC_CGM_AC8_DC0)....740
25.3.46 Aux Clock 9 Select Control (MC_CGM_AC9_SC)....741
25.3.47 Aux Clock 9 Select Status (MC_CGM_AC9_SS)....742
25.3.48 Aux Clock 9 Divider Configuration 0 (MC_CGM_AC9_DC0)....743
25.3.49 Aux Clock 10 Select Control (MC_CGM_AC10_SC)....743
25.3.50 Aux Clock 10 Select Status (MC_CGM_AC10_SS)....744
25.3.51 Aux Clock 10 Divider Configuration 0 (MC_CGM_AC10_DC0)....745
25.4 Functional Description....746
25.4.1 System Clock Generation....746
25.4.2 Auxiliary Clock Generation....751
25.4.3 Dividers Functional Description....758

Chapter 26
OSC Digital Interface (XOSC)

26.1 Introduction....769
26.2 Functional description....769
26.2.1 Oscillator power-down control and status....770
26.2.2 Oscillator startup delay....770
26.2.3 Oscillator clock available interrupt....770
26.2.4 Oscillator bypass mode....771

26.3 Memory map and register definition....771
26.3.1 XOSC Control Register (XOSC_CTL)....772

Chapter 27
IRCOSC Digital Interface

27.1 Introduction....775
27.2 Functional description....775
27.3 Memory map and register definition....776
27.3.1 IRCOSC Control Register (IRCOSC_CTL)....776
27.3.2 Frequency trimming calculation....777

Chapter 28
RAM Controller (PRAMC)

28.1 Introduction....779
28.2 SRAM controller memory map and register definition....780
28.2.1 Platform RAM Configuration Register 1 (PRAMC_PRCR1)....781
28.3 Functional description....783
28.3.1 Read and Write operations....783
28.3.2 Writes....784
28.4 Initialization/application information....786
28.5 Reliability considerations....786
28.5.1 Hsiao ECC algorithm....786

Chapter 29
Flash memory controller

29.1 Flash memory controller (PFLASH) chip-specific module information....791
29.1.1 PFCR1[RWSC]....791
29.2 Introduction....791
29.3 Features....792
29.4 Block diagrams....792
29.5 Flash memory controller memory map....793
29.5.1 Platform Flash Configuration Register 1 (PFLASH_PFCR1)....798
29.5.2 Platform Flash Configuration Register 2 (PFLASH_PFCR2)....802

29.5.3 Platform Flash Configuration Register 3 (PFLASH_PFCR3)....805
29.5.4 Platform Flash Access Protection Register (PFLASH_PFAPR)....807
29.5.5 Platform Flash Remap Control Register (PFLASH_PFCRCR)....810
29.5.6 Platform Flash Remap Descriptor Enable Register (PFLASH_PFCRDE)....812
29.5.7 Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD*n*_Word0)....817
29.5.8 Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD*n*_Word1)....818
29.5.9 Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD*n*_Word2)....819
29.6 Functional description....824
29.6.1 Basic interface protocol....825
29.6.2 Access protections....825
29.6.3 Access pipelining....827
29.6.4 Line read buffers and prefetch operation....827
29.6.5 Instruction/Data prefetch triggering....828
29.6.6 Per-Master prefetch triggering....829
29.6.7 Buffer allocation....829
29.6.8 PFlash calibration remap support....829
29.6.9 Reliability considerations....836
29.6.10 ECC on data flash accesses....840
29.6.11 Array integrity considerations....841

Chapter 30
Embedded Flash Memory (c55fmc)

30.1 Embedded Flash Memory (c55fmc) chip-specific module information....843
30.2 C55FMC_LOCKn register bit mapping....843
30.3 C55FMC_SELn register bit mapping....843
30.4 Embedded Flash Memory (c55fmc) register reset values....843
30.5 Introduction....844
30.5.1 Overview....845
30.5.2 Features....846
30.5.3 Modes of operation....846

30.6 UTest NVM block....847
30.7 Test mode disable seal....847
30.8 Memory map and register definition....848
30.8.1 Module Configuration Register (C55FMC_MCR)....850
30.8.2 Extended Module Configuration Register (C55FMC_MCRE)....855
30.8.3 Lock 0 register (C55FMC_LOCK0)....858
30.8.4 Lock 1 register (C55FMC_LOCK1)....860
30.8.5 Lock 2 register (C55FMC_LOCK2)....860
30.8.6 Lock 3 register (C55FMC_LOCK3)....861
30.8.7 Select 0 register (C55FMC_SEL0)....862
30.8.8 Select 1 register (C55FMC_SEL1)....863
30.8.9 Select 2 register (C55FMC_SEL2)....864
30.8.10 Select 3 register (C55FMC_SEL3)....865
30.8.11 Address register (C55FMC_ADR)....866
30.8.12 UTest 0 register (C55FMC_UT0)....868
30.8.13 UMISR register (C55FMC_UM*n*)....871
30.8.14 UMISR register (C55FMC_UM9)....872
30.8.15 Over-Program Protection 0 register (C55FMC_OPP0)....873
30.8.16 Over-Program Protection 1 register (C55FMC_OPP1)....874
30.8.17 Over-Program Protection 2 register (C55FMC_OPP2)....875
30.8.18 Over-Program Protection 3 register (C55FMC_OPP3)....875
30.8.19 Test Mode Disable Password Check register (C55FMC_TMD)....876
30.9 Functional Description....877
30.9.1 User Mode....878
30.9.2 UTest mode....886
30.10 Initialization information....890

Chapter 31
Decorated Storage Memory Controller (DSMC)

31.1 Decorated Storage Memory Controller (DSMC) chip-specific module information....893

31.2 Introduction....894
31.3 Decorated stores: st[b,h,w]d{cb}x rS,rA,rB....896
31.3.1 Bit Field Insert (BFINS) into an 8, 16 or 32-bit memory container....897
31.4 DSMC instantiations....902

Chapter 32
Flash Memory Programming and Configuration

32.1 Introduction....903
32.2 Selection of flash memory blocks for erase....905
32.3 Non-secure write protection....907
32.4 Security configuration planning....909
32.4.1 Creating password groups....909
32.5 Secure write protection....910
32.5.1 Implementing secure write protection....911
32.5.2 Overriding secure write protection....915
32.6 Secure read protection....917
32.6.1 Implementing secure read protection....918
32.6.2 Overriding secure read protection....922
32.7 Debug port enable/disable....923
32.8 Tamper detection....923
32.8.1 Implementing tamper detection....924
32.8.2 Creating the tamper detect diary....924
32.8.3 Assigning blocks to Tamper Detection Regions (TDRs)....926
32.8.4 Overriding tamper detection....926
32.9 Implementing OTP....927

Chapter 33
Analog Peripheral Modules

33.1 Analog sub-system overview....929
33.1.1 Block diagram....929

Chapter 34
Sigma Delta Analog-to-Digital Converter (SDADC) Digital Interface

34.1 Sigma Delta Analog-to-Digital Converter (SDADC) digital interface chip-specific information....931
34.1.1 Introduction....931
34.1.2 Sigma-Delta ADC features....931
34.1.3 Sigma-Delta ADC block diagram....932
34.1.4 Sigma-Delta ADC external signal description....933
34.1.5 Sigma-Delta ADC analog input multiplexing....933
34.1.6 Sigma-Delta digital interface....934
34.1.7 Sigma-Delta ADC clock architecture....934
34.1.8 Sigma-Delta ADC starting and triggering....935
34.1.9 Sigma-Delta ADC input biasing....937
34.1.10 Sigma-Delta ADC conversion result processing....937
34.1.11 Sigma-Delta interface to ETPU module....938
34.1.12 Sigma-Delta interface to eMIOS module....939
34.1.13 Sigma-Delta interface to decimation filter modules....940
34.1.14 Sigma-Delta interface to reaction channels....940
34.1.15 Sigma-Delta Interface to the Cross Triggering Unit (CTU)....940
34.2 Sigma Delta Analog-to-Digital Converter (SDADC) digital interface reference....941
34.2.1 Introduction....941
34.2.2 Overview....941
34.2.3 Features....941
34.2.4 Modes of operation....942
34.2.5 External signal description....942

34.2.6 Memory map and register definition....943
34.2.7 Functional description....958
34.2.8 Initialization information....964

Chapter 35
Analog-to-Digital Converter (ADC)

35.1 Analog-to-Digital Converter (ADC) chip-specific information....967
35.1.1 Introduction....967
35.1.2 SAR ADC block diagram....967
35.1.3 SAR ADC external signal description....968
35.1.4 SARADC feature comparison....969
35.1.5 SARADC channel map....969
35.1.6 SARADC control registers....973
35.1.7 SAR ADC trigger inputs....977
35.1.8 SAR ADC externally muxed inputs....977
35.1.9 SAR ADC conversion result processing....979
35.2 Analog-to-Digital Converter (ADC) reference....980
35.2.1 Introduction....980
35.2.2 Overview....980
35.2.3 Features....983
35.2.4 Memory Map/Register Definition....985
35.2.5 Functional description....1102
35.2.6 Clock frequency....1132

Chapter 36
Decimation Filter (DECFILTER)

36.1 Decimation Filter chip-specific module information....1135
36.2 Introduction....1135
36.2.1 Overview....1135
36.2.2 Features....1137

36.2.3 Modes of operation....1137
36.3 External signal description....1139
36.3.1 Integrator enable signal....1139
36.3.2 Integrator halt signal....1139
36.3.3 Integrator reset signal....1139
36.3.4 Integrator output request signal....1139
36.4 Memory map and register definition....1140
36.4.1 Module Configuration Register (DECFILTER_MCR)....1141
36.4.2 Module Status Register (DECFILTER_MSR)....1145
36.4.3 Module Extended Config Register (DECFILTER_MXCR)....1149
36.4.4 Module Extended Status Register (DECFILTER_MXSR)....1152
36.4.5 Interface Input Buffer (DECFILTER_IB)....1155
36.4.6 Interface Output Buffer (DECFILTER_OB)....1156
36.4.7 Filter Coefficient (DECFILTER_COEF*n*)....1157
36.4.8 Filter TAP (DECFILTER_TAP*n*)....1157
36.4.9 Final Integer Value Register (DECFILTER_FINTVAL)....1158
36.4.10 Final Integer Count Register (DECFILTER_FINTCNT)....1159
36.4.11 Current Integer Value Register (DECFILTER_CINTVAL)....1159
36.4.12 Current Integer Count Register (DECFILTER_CINTCNT)....1160
36.5 Functional Description....1160
36.5.1 Overview....1160
36.5.2 Data Interface sub-block description....1160
36.5.3 Input Buffer Description....1161
36.5.4 Output Buffer Description....1162
36.5.5 Bypass Configuration Description....1163
36.5.6 IIR and FIR Filter....1163
36.5.7 Filter Prefill Control Description....1167
36.5.8 Flush Command Description....1168
36.5.9 Soft-Reset Command Description....1169

36.5.10 Interrupts Requests Description....1170
36.5.11 DMA Requests Description....1172
36.5.12 Freeze Mode Description....1173
36.5.13 Integrator....1173
36.5.14 Cascade mode description....1178
36.6 Initialization Information....1182
36.6.1 Initialization Procedure....1182
36.7 Filter Example Simulation....1183
36.7.1 Coefficients Calculation....1183
36.7.2 Input Data Calculation....1184
36.7.3 Filter Results....1185

Chapter 37
Temperature Sensor (TSENS)

37.1 Introduction....1187
37.2 Functional Description....1187
37.2.1 Temperature threshold detection (digital output generation)....1188
37.2.2 Linear temperature sensor (analog output generation)....1189
37.3 Temperature formula....1189
37.4 Calculating device temperature....1190

Chapter 38
Timers Overview

38.1 Timers Overview....1191

Chapter 39
System Timer Module (STM)

39.1 System Timer Module (STM) configuration....1193
39.1.1 STM chip-specific module information....1193
39.2 Introduction....1193
39.2.1 Overview....1193
39.2.2 Features....1194
39.2.3 Modes of operation....1194

39.3 External signal description....1194
39.4 Memory map and registers....1194
39.4.1 STM Control Register (STM_CR)....1195
39.4.2 STM Count Register (STM_CNT)....1196
39.4.3 STM Channel Control Register (STM_CCR*n*)....1197
39.4.4 STM Channel Interrupt Register (STM_CIR*n*)....1197
39.4.5 STM Channel Compare Register (STM_CMP*n*)....1198
39.5 Functional description....1198

Chapter 40
Software Watchdog Timer (SWT)

40.1 Software Watchdog Timer (SWT) chip-specific module information....1201
40.1.1 Implementation....1201
40.1.2 Configuration....1201
40.1.3 SWT_TO register minimum value....1202
40.1.4 SWT time-out events: configuring reset and interrupt requests....1202
40.2 Introduction....1203
40.2.1 Overview....1203
40.2.2 Features....1203
40.2.3 Modes of operation....1203
40.3 External signal description....1204
40.4 Memory Map and Registers....1204
40.4.1 SWT Control Register (SWT_CR)....1205
40.4.2 SWT Interrupt Register (SWT_IR)....1208
40.4.3 SWT Time-out Register (SWT_TO)....1208
40.4.4 SWT Window Register (SWT_WN)....1209
40.4.5 SWT Service Register (SWT_SR)....1209
40.4.6 SWT Counter Output Register (SWT_CO)....1210
40.4.7 SWT Service Key Register (SWT_SK)....1211

40.5 Functional description....1211
40.5.1 Introduction....1211
40.5.2 Configuration locking....1212
40.5.3 Unlock sequence....1213
40.5.4 Servicing operations....1213
40.5.5 Time-out....1215
40.5.6 Initialization....1215

Chapter 41
Periodic Interrupt Timer (PIT-RTI)

41.1 Periodic Interrupt Timer (PIT-RTI) chip-specific module information....1217
41.1.1 Overview....1217
41.1.2 Configuration....1217
41.1.3 General-use PIT....1218
41.2 Introduction....1218
41.2.1 Block diagram....1218
41.2.2 Features....1219
41.3 Modes of operation....1220
41.4 Signal description....1220
41.5 Memory map/register description....1220
41.5.1 PIT Module Control Register (PIT_MCR)....1222
41.5.2 PIT Upper Lifetime Timer Register (PIT_LTMR64H)....1223
41.5.3 PIT Lower Lifetime Timer Register (PIT_LTMR64L)....1224
41.5.4 Timer Load Value Register (PIT_RTI_LDVAL)....1224
41.5.5 Current Timer Value Register (PIT_RTI_CVAL)....1225
41.5.6 Timer Control Register (PIT_RTI_TCTRL)....1225
41.5.7 Timer Flag Register (PIT_RTI_TFLG)....1226
41.5.8 Timer Load Value Register (PIT_LDVAL*n*)....1227
41.5.9 Current Timer Value Register (PIT_CVAL*n*)....1227
41.5.10 Timer Control Register (PIT_TCTRL*n*)....1228

41.5.11 Timer Flag Register (PIT_TFLG*n*)....1229
41.6 Functional description....1229
41.6.1 General operation....1229
41.6.2 Interrupts....1231
41.6.3 Chained timers....1231
41.7 Initialization and application information....1232
41.8 Example configuration for chained timers....1233
41.9 Example configuration for the lifetime timer....1234

Chapter 42
Enhanced Modular IO Subsystem (eMIOS200)

42.1 Enhanced Modular IO Subsystem (eMIOS253) chip-specific module information....1235
42.1.1 Configuration....1235
42.1.2 Global Timebase Enabling....1235
42.1.3 Clocking....1236
42.1.4 STAC bus....1236
42.1.5 Output Disable Inputs....1236
42.2 eMIOS200 Introduction....1236
42.2.1 Overview....1236
42.2.2 Features....1238
42.2.3 Modes of operation....1239
42.3 Memory map / register definition....1240
42.3.1 Memory map....1240
42.4 Control Memory Map and Registers....1242
42.4.1 eMIOS Module Configuration Register (eMIOS_MCR)....1242
42.4.2 eMIOS Global FLAG Register (eMIOS_GFLAG)....1244
42.4.3 eMIOS Output Update Disable register (eMIOS_OUDIS)....1244
42.4.4 eMIOS Disable Channel register (eMIOS_UCDIS)....1245
42.5 Unified Channels Memory Map and Registers....1246
42.5.1 eMIOS UC A register n (eMIOS_A*n*)....1248

42.5.2 eMIOS UC B register n (eMIOS_B*n*)....1249
42.5.3 eMIOS CNT register n (eMIOS_CNT*n*)....1250
42.5.4 eMIOS UC Control register n (eMIOS_C*n*)....1251
42.5.5 eMIOS UC Status register n (eMIOS_S*n*)....1257
42.5.6 eMIOS Alternate Address register n (eMIOS_ALTA*n*)....1259
42.6 Functional Description....1259
42.6.1 Unified Channel (UC)....1261
42.6.2 IP Bus Interface Unit (BIU)....1311
42.6.3 STAC Bus Client submodule....1312
42.6.4 Global Clock Prescaler Submodule (GCP)....1312
42.7 Initialization/Application Information....1313
42.7.1 Considerations....1313
42.7.2 Application Information....1314

Chapter 43
Enhanced Time Processing Unit (eTPU)

43.1 Enhanced Time Processing Unit (eTPU) chip-specific module information....1319
43.1.1 Configuration....1319
43.1.2 Device-specific information....1319
43.1.3 Register reset values....1320
43.1.4 Memory....1320
43.1.5 Clocking....1320
43.1.6 Debug....1321
43.1.7 STAC bus....1321
43.1.8 Input connections....1321
43.1.9 Output connections....1322

43.2 Introduction....1322
43.2.1 Overview....1323
43.2.2 Features....1330
43.2.3 Modes of Operation....1337
43.3 Detailed Signal Descriptions....1339
43.3.1 eTPU Channel Output Signals....1339
43.3.2 eTPU Channel Input Signals....1339
43.3.3 Time Base Clock Signal — TCRCLK....1340
43.3.4 eTPU Channel Output Disable Signals....1340
43.4 Memory Map/Register Definition....1341
43.5 Memory Map....1341
43.6 System Configuration memory map and registers....1346
43.6.1 eTPU Module Configuration Register (eTPU_SCR_MCR)....1347
43.6.2 eTPU Coherent Dual-Parameter Controller Register (eTPU_SCR_CDCR)....1351
43.6.3 eTPU MISC Compare Register (eTPU_SCR_MISCCMPR)....1353
43.6.4 eTPU SCM Off-range Data Register (eTPU_SCR_SCMOFFDATAR)....1353
43.6.5 eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1)....1354
43.6.6 eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2)....1358
43.7 Timebase memory map and registers....1362
43.7.1 eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1)....1363
43.7.2 eTPU Engine 1 Time Base 1 (TCR1) Visibility Register (eTPU_TBR_TB1R_ENG1)....1367
43.7.3 eTPU Engine 1 Time Base 2 (TCR2) Visibility Register (eTPU_TBR_TB2R_ENG1)....1367
43.7.4 eTPU Engine 1 STAC Configuration Register (eTPU_TBR_STAC_ENG1)....1368
43.7.5 eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2)....1370
43.7.6 eTPU Engine 2 Time Base 1 (TCR1) Visibility Register (eTPU_TBR_TB1R_ENG2)....1374
43.7.7 eTPU Engine 2 Time Base 2 (TCR2) Visibility Register (eTPU_TBR_TB2R_ENG2)....1374
43.7.8 eTPU Engine 2 STAC Configuration Register (eTPU_TBR_STAC_ENG2)....1375
43.8 Engine memory map and registers....1377
43.8.1 eTPU Watchdog Timer Register for Engine 1 (eTPU_ER_WDTR_ENG1)....1377

43.8.2 eTPU Idle Register for Engine 1 (eTPU_ER_IDLER_ENG1).....1379
43.8.3 eTPU Watchdog Timer Register for Engine 2 (eTPU_ER_WDTR_ENG2).....1380
43.8.4 eTPU Idle Register for Engine 2 (eTPU_ER_IDLER_ENG2).....1381
43.9 Latency Monitor memory map and registers.....1382
43.9.1 eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM*n*CR_ENG1).....1383
43.9.2 eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM*n*MR_ENG1).....1385
43.9.3 eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM*n*CR_ENG2).....1386
43.9.4 eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM*n*MR_ENG2).....1387
43.10 Memory Error Support memory map and registers.....1388
43.10.1 eTPU Memory Error Control Register (eTPU_MESR_MECR).....1389
43.10.2 eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR).....1392
43.10.3 eTPU Data Error Injection Data Pattern Register (eTPU_MESR_DEIDPR).....1393
43.10.4 eTPU Data Error Injection Parity Pattern Register (eTPU_MESR_DEIPPR).....1393
43.10.5 eTPU Data Error Report Address Register (eTPU_MESR_DERAR).....1395
43.10.6 eTPU Data Error Report Data Register (eTPU_MESR_DERDR).....1396
43.10.7 eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR).....1397
43.10.8 eTPU Memory Error Status Register (eTPU_MESR_MESR).....1399
43.10.9 eTPU Code Error Injection Address Register (eTPU_MESR_CEIAR).....1402
43.10.10 eTPU Code Error Injection Data Pattern Register (eTPU_MESR_CEIDPR).....1403
43.10.11 eTPU Code Error Injection Parity Pattern Register (eTPU_MESR_CEIPPR).....1403
43.10.12 eTPU Code Error Report Address Register (eTPU_MESR_CERAR).....1404
43.10.13 eTPU Code Error Report Data Register (eTPU_MESR_CERDR).....1405
43.10.14 eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR).....1406
43.10.15 eTPU Code Error Fix Register (eTPU_MESR_CEFR).....1407
43.11 Global Channel memory map and registers.....1408
43.11.1 eTPU Channel Interrupt Status Register for Engine 1 (eTPU_GCR_CISR_ENG1).....1409
43.11.2 eTPU Channel Interrupt Status Register for Engine 2 (eTPU_GCR_CISR_ENG2).....1409
43.11.3 eTPU Channel Data Transfer Request Status Register for Engine 1 (eTPU_GCR_CDTRSR_ENG1).....1410
43.11.4 eTPU Channel Data Transfer Request Status Register for Engine 2 (eTPU_GCR_CDTRSR_ENG2).....1411

43.11.5 eTPU Channel Interrupt Overflow Status Register for Engine 1 (eTPU_GCR_CIOSR_ENG1).....1412
43.11.6 eTPU Channel Interrupt Overflow Status Register for Engine 2 (eTPU_GCR_CIOSR_ENG2).....1412
43.11.7 eTPU Channel Data Transfer Request Overflow Status Register for Engine 1 (eTPU_GCR_CDTROSR_ENG1).....1413
43.11.8 eTPU Channel Data Transfer Request Overflow Status Register for Engine 2 (eTPU_GCR_CDTROSR_ENG2).....1414
43.11.9 eTPU Channel Interrupt Enable Register for Engine 1 (eTPU_GCR_CIER_ENG1).....1414
43.11.10 eTPU Channel Interrupt Enable Register for Engine 2 (eTPU_GCR_CIER_ENG2).....1415
43.11.11 eTPU Channel Data Transfer Request Enable Register for Engine 1 (eTPU_GCR_CDTRER_ENG1).…1416
43.11.12 eTPU Channel Data Transfer Request Enable Register for Engine 2 (eTPU_GCR_CDTRER_ENG2).…1416
43.11.13 eTPU Watchdog Status Register for Engine 1 (eTPU_GCR_WDSR_ENG1).....1417
43.11.14 eTPU Watchdog Status Register for Engine 2 (eTPU_GCR_WDSR_ENG2).....1417
43.11.15 eTPU Channel Pending Service Status Register for Engine 1 (eTPU_GCR_CPSSR_ENG1).....1418
43.11.16 eTPU Channel Pending Service Status Register for Engine 2 (eTPU_GCR_CPSSR_ENG2).....1419
43.11.17 eTPU Channel Service Status Register for Engine 1 (eTPU_GCR_CSSR_ENG1).....1419
43.11.18 eTPU Channel Service Status Register for Engine 2 (eTPU_GCR_CSSR_ENG2).....1420
43.12 Channel Configuration Control memory map and registers.....1421
43.12.1 eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1).....1431
43.12.2 eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C*n*SCR_ENG1).....1434
43.12.3 eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C*n*HSRR_ENG1).....1436
43.12.4 eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2).....1437
43.12.5 eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C*n*SCR_ENG2).....1440
43.12.6 eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C*n*HSRR_ENG2).....1442
43.13 Shared Subsystem Integration memory map and registers.....1443
43.13.1 eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR*n*).....1444

43.14 Functional Description....1444
43.14.1 Functions and Threads....1445
43.14.2 Host Interface....1460
43.14.3 Scheduler....1468
43.14.4 Parameter Sharing and Coherency....1476
43.14.5 Enhanced Channels....1482
43.14.6 Time Bases....1533
43.14.7 EAC - eTPU Angle Counter....1542
43.14.8 Microengine....1564
43.14.9 Microinstruction Set....1583
43.14.10 Shared Subsystems Support....1617
43.14.11 Development Support Features....1620
43.14.12 Safety Features....1628
43.14.13 Performance Monitoring Features....1637
43.15 Initialization/Application Information....1639
43.15.1 Configuration Sequence....1639
43.15.2 Reset Options....1641
43.15.3 Multiple Parameter Coherency Methods....1642
43.15.4 Programming Hints and Caveats....1643
43.15.5 Estimating Worst Case Latency....1645
43.15.6 Endianness....1660
43.15.7 Memory Error Service Procedures....1660
43.16 Appendices....1663
43.16.1 Microcycle and I/O Timing....1663
43.16.2 Channel Mode Summary....1668
43.16.3 MISC Algorithm....1671

Chapter 44
Reaction Module 2 (REACM2)

44.1 Reaction Module 2 chip-specific module information....1673

44.2 Introduction....1674
44.2.1 Features....1675
44.2.2 Modes of Operation....1675
44.2.3 Block Diagram....1677
44.3 Signal Description....1681
44.3.1 REACM2_RCHn — REACM Channel (n) Output Pin a, b and c....1681
44.4 Memory Map and Register Definition....1682
44.4.1 Module Configuration (REACM2_MCR)....1687
44.4.2 Timer Configuration (REACM2_TCR)....1689
44.4.3 Threshold Router Register (REACM2_THRR)....1690
44.4.4 ADC Sensor Input Register (REACM2_SINR)....1691
44.4.5 Period Generator Configuration Register (REACM2_PCR)....1692
44.4.6 Period Shift Delay Configuration Register (REACM2_PSCR)....1693
44.4.7 Global Error Flag Register (REACM2_GEFR)....1694
44.4.8 Channel n Configuration Register (REACM2_CHCR*n*)....1695
44.4.9 Channel n Status Register (REACM2_CHSR*n*)....1699
44.4.10 Channel n Router Register (REACM2_CHRR*n*)....1702
44.4.11 Shared Timer Bank (REACM2_STBK*n*)....1703
44.4.12 Hold Timer Bank (REACM2_HOTBK*n*)....1703
44.4.13 Threshold Bank (REACM2_THBK*n*)....1704
44.4.14 ADC Result Maximum Limit Check (REACM2_ADCMAX)....1705
44.4.15 Modulation Range Pulse Width (REACM2_RANGE_PWD)....1705
44.4.16 Modulation Minimum Pulse Width (REACM2_MIN_PWD)....1706
44.4.17 Modulation Control Word Bank (REACM2_MWBK*n*)....1707
44.5 Functional Description....1709
44.5.1 Reaction Channel....1710
44.5.2 Modulation Control Words Bank....1712
44.5.3 Shared Timer Bank....1713
44.5.4 Hold Time Bank....1715

44.5.5 Threshold Bank and Comparator....1716
44.5.6 ADC Interface....1717
44.5.7 Prescalers....1719
44.5.8 Period / Delay Generator....1720
44.5.9 Banked Mode Support....1721
44.6 Modulation Modes....1722
44.6.1 Threshold/Threshold Mode....1722
44.6.2 Threshold/Hold-off Mode....1723
44.6.3 Threshold/Hold-on Mode....1724
44.6.4 Threshold low/Period Mode....1725
44.6.5 Threshold high/Period Mode....1726
44.6.6 Limitations on the Modulation Process....1727
44.7 Monitored Modulation....1731
44.7.1 Monitoring the ON Pulse Width....1731
44.7.2 Monitoring the OFF State Width and the ADC Maximum Value....1735
44.8 DMA support....1737
44.9 Reset Overview....1738
44.10 Reaction Module Interrupts....1739
44.10.1 Interrupt sources....1739
44.10.2 Global Interrupt and Channel Interrupt....1739
44.11 Use cases....1740
44.11.1 Advancing Modulation Phase on a Threshold level....1747
44.11.2 Controlling the LOOP function....1748
44.11.3 Banked Mode....1749

Chapter 45
Input Glitch Filter (IGF)

45.1 Input Glitch Filter chip-specific module information....1751
45.2 Introduction....1752
45.3 Filter Cluster....1752

45.4 Block Diagram....1753
45.5 Input Glitch Filter overview....1754
45.5.1 Interaction with other peripherals....1754
45.5.2 Mode of operation....1755
45.6 Memory map....1757
45.6.1 Module Configuration Register (IGF_MCR*n*)....1761
45.6.2 Module Status Register (IGF_MSR*n*)....1764
45.6.3 Prescaler Register (IGF_PRESR*n*)....1765
45.6.4 Rising edge Threshold Register (IGF_RTHR*n*)....1766
45.6.5 Falling edge Threshold Register (IGF_FTHR*n*)....1766
45.7 Functional description....1766
45.7.1 Input double sampling....1767
45.7.2 Filter types....1767
45.7.3 Application....1771
45.7.4 Programming guidelines....1774
45.7.5 On-the-fly filter reconfiguration....1774
45.8 Power saving mode....1775
45.8.1 STOP mode....1775

Chapter 46
Body Cross-triggering Unit (BCTU)

46.1 Body Cross-triggering Unit (BCTU) chip-specific module information....1777
46.1.1 Overview....1777
46.1.2 Functional Description Summary....1781
46.2 Introduction....1781
46.3 BCTU overview....1782
46.4 BCTU Interacting with other peripherals....1783
46.5 BCTU Modes of operation....1784
46.5.1 Configuration Mode....1785
46.5.2 Normal Mode....1785

46.5.3 Low Power Mode....1785
46.6 Memory Map....1786
46.6.1 Module Configuration Register (BCTU_MCR)....1789
46.6.2 Module Status register (BCTU_MSR)....1792
46.6.3 Trigger Configuration Register (BCTU_TRGCFG_*n*)....1796
46.6.4 Write Protection register (BCTU_WRPROT)....1798
46.6.5 Software Trigger register 1 (BCTU_SFTRGR1)....1799
46.6.6 Software Trigger register 1 (BCTU_SFTRGR3)....1802
46.6.7 ADC0 Result Data register (BCTU_ADC0DR)....1806
46.6.8 ADC1 result data register (BCTU_ADC1DR)....1807
46.6.9 ADC2 result data register (BCTU_ADC2DR)....1808
46.6.10 ADC3 result data register (BCTU_ADC3DR)....1809
46.6.11 LIST Size Status register (BCTU_LISTSTAR)....1809
46.6.12 LIST Channel Address Register (BCTU_LISTCHR_*n*)....1810
46.6.13 Conversion configuration register (BCTU_CCR_*n*)....1811
46.7 Functional description....1812
46.7.1 Trigger Configuration....1812
46.7.2 Trigger priority and multiplexing....1813
46.7.3 Conversion LIST....1815
46.7.4 ADC conversion results access....1817
46.7.5 Trigger clearing mechanism....1818
46.7.6 On-the-fly reconfiguration of BCTU registers....1819
46.7.7 Multiple Parallel Conversions....1820
46.8 Interrupts and DMA requests....1821

Chapter 47
CAN (FlexCAN)

47.1 CAN chip-specific module information....1823
47.1.1 FlexCAN modules on this chip....1823
47.1.2 Message buffer size....1823

47.1.3 Message buffer address offsets....1824
47.1.4 Reset value of MDIS bit....1826
47.1.5 FlexCAN Clocking....1826
47.1.6 FlexCAN Interrupts....1827
47.1.7 FlexCAN CLKSRC....1827
47.1.8 FlexCAN RAM to memory map address mapping....1827
47.2 Introduction....1828
47.2.1 Overview....1829
47.2.2 FlexCAN module features....1830
47.2.3 Modes of operation....1832
47.3 FlexCAN signal descriptions....1833
47.3.1 CAN Rx....1833
47.3.2 CAN Tx....1833
47.4 Memory map/register definition....1834
47.4.1 FlexCAN memory mapping....1834
47.4.2 Module Configuration Register (CAN_MCR)....1842
47.4.3 Control 1 register (CAN_CTRL1)....1847
47.4.4 Free Running Timer (CAN_TIMER)....1851
47.4.5 Rx Mailboxes Global Mask Register (CAN_RXMGMASK)....1851
47.4.6 Rx 14 Mask register (CAN_RX14MASK)....1853
47.4.7 Rx 15 Mask register (CAN_RX15MASK)....1853
47.4.8 Error Counter (CAN_ECR)....1854
47.4.9 Error and Status 1 register (CAN_ESR1)....1856
47.4.10 Interrupt Masks 2 register (CAN_IMASK2)....1862
47.4.11 Interrupt Masks 1 register (CAN_IMASK1)....1862
47.4.12 Interrupt Flags 2 register (CAN_IFLAG2)....1863
47.4.13 Interrupt Flags 1 register (CAN_IFLAG1)....1864
47.4.14 Control 2 register (CAN_CTRL2)....1867
47.4.15 Error and Status 2 register (CAN_ESR2)....1870

47.4.16 CRC Register (CAN_CRCR)....1872
47.4.17 Rx FIFO Global Mask register (CAN_RXFGMASK)....1872
47.4.18 Rx FIFO Information Register (CAN_RXFIR)....1873
47.4.19 CAN Bit Timing Register (CAN_CBT)....1874
47.4.20 Interrupt Masks 3 Register (CAN_IMASK3)....1876
47.4.21 Interrupt Flags 3 Register (CAN_IFLAG3)....1877
47.4.22 Rx Individual Mask Registers (CAN_RXIMR*n*)....1877
47.4.23 Memory Error Control Register (CAN_MECR)....1878
47.4.24 Error Injection Address Register (CAN_ERRIAR)....1880
47.4.25 Error Injection Data Pattern Register (CAN_ERRIDPR)....1881
47.4.26 Error Injection Parity Pattern Register (CAN_ERRIPPR)....1881
47.4.27 Error Report Address Register (CAN_RERRAR)....1882
47.4.28 Error Report Data Register (CAN_RERRDR)....1884
47.4.29 Error Report Syndrome Register (CAN_RERRSYNR)....1884
47.4.30 Error Status Register (CAN_ERRSR)....1887
47.4.31 Message buffer structure....1888
47.4.32 Rx FIFO structure....1894
47.5 Functional description....1896
47.5.1 Transmit process....1897
47.5.2 Arbitration process....1898
47.5.3 Receive process....1902
47.5.4 Matching process....1904
47.5.5 Move process....1908
47.5.6 Data coherence....1910
47.5.7 Rx FIFO....1914
47.5.8 CAN protocol related features....1917
47.5.9 Clock domains and restrictions....1924
47.5.10 Modes of operation details....1926
47.5.11 Interrupts....1928

47.5.12 Bus interface....1930
47.5.13 Detection and Correction of Memory Errors....1930
47.6 Initialization/application information....1935
47.6.1 FlexCAN initialization sequence....1935

Chapter 48
De-serial Serial Peripheral Interface (DSPI)

48.1 DSPI chip-specific module information....1937
48.1.1 Number of CTARs....1937
48.1.2 Number of PCS signals....1938
48.1.3 IP Instantiation Parameters....1938
48.2 Introduction....1939
48.2.1 Block Diagram....1939
48.2.2 Features....1940
48.2.3 SPI Configuration....1942
48.2.4 DSI Configuration....1943
48.2.5 CSI Configuration....1943
48.2.6 Modes of Operation....1944
48.3 Module signal descriptions....1945
48.3.1 PCS0/SS — Peripheral Chip Select/Slave Select....1946
48.3.2 PCS1 – PCS3 — Peripheral Chip Selects 1 – 3....1946
48.3.3 PCS4/MTRIG — Peripheral Chip Select 4/Master Trigger....1946
48.3.4 PCS5/PCSS — Peripheral Chip Select 5/Peripheral Chip Select Strobe....1946
48.3.5 SIN — Serial Input....1947
48.3.6 SOUT — Serial Output....1947
48.3.7 SCK — Serial Clock....1947
48.4 Memory Map/Register Definition....1947
48.4.1 Module Configuration Register (DSPI_MCR)....1950
48.4.2 Transfer Count Register (DSPI_TCR)....1953
48.4.3 Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR*n*)....1954

48.4.4 Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTAR*n*_SLAVE)....1959
48.4.5 Status Register (DSPI_SR)....1961
48.4.6 DMA/Interrupt Request Select and Enable Register (DSPI_RSER)....1965
48.4.7 PUSH TX FIFO Register In Master Mode (DSPI_PUSHR)....1968
48.4.8 PUSH TX FIFO Register In Slave Mode (DSPI_PUSHR_SLAVE)....1970
48.4.9 POP RX FIFO Register (DSPI_POPR)....1971
48.4.10 Transmit FIFO Registers (DSPI_TXFR*n*)....1971
48.4.11 Receive FIFO Registers (DSPI_RXFR*n*)....1972
48.4.12 DSI Configuration Register 0 (DSPI_DSICR0)....1973
48.4.13 DSI Serialization Data Register 0 (DSPI_SDR0)....1975
48.4.14 DSI Alternate Serialization Data Register 0 (DSPI_ASDR0)....1976
48.4.15 DSI Transmit Comparison Register 0 (DSPI_COMPR0)....1976
48.4.16 DSI Configuration Register 1 (DSPI_DSICR1)....1977
48.4.17 DSI Serialization Source Select Register 0 (DSPI_SSR0)....1979
48.4.18 DSI Serialization Data Register 1 (DSPI_SDR1)....1979
48.4.19 DSI Alternate Serialization Data Register 1 (DSPI_ASDR1)....1980
48.4.20 DSI Transmit Comparison Register 1 (DSPI_COMPR1)....1981
48.4.21 DSI Deserialization Data Register 1 (DSPI_DDR1)....1981
48.4.22 DSI Serialization Source Select Register 1 (DSPI_SSR1)....1982
48.4.23 DSI Deserialized Data Interrupt Mask Register 1 (DSPI_DIMR1)....1983
48.4.24 DSI Deserialized Data Polarity Interrupt Register 1 (DSPI_DPIR1)....1983
48.4.25 Clock and Transfer Attributes Register Extended (DSPI_CTARE*n*)....1984
48.4.26 Status Register Extended (DSPI_SREX)....1985
48.4.27 Trigger Register (DSPI_TRIG)....1986
48.4.28 Time Slot Length Register (DSPI_TSL)....1987
48.4.29 Time Slot Configuration Register (DSPI_TS_CONF)....1988
48.5 Functional description....1990
48.5.1 Start and Stop of module transfers....1991

48.5.2 Serial Peripheral Interface (SPI) configuration....1992
48.5.3 Deserial Serial Interface (DSI) Configuration....1997
48.5.4 Combined Serial Interface (CSI) Configuration....2002
48.5.5 Module baud rate and clock delay generation....2004
48.5.6 Transfer formats....2007
48.5.7 Continuous Serial Communications Clock....2019
48.5.8 Slave Mode Operation Constraints....2021
48.5.9 Timed Serial Bus (TSB)....2021
48.5.10 Interleaved TSB (ITSB) Mode....2024
48.5.11 Parity Generation and Check....2033
48.5.12 Interrupts/DMA requests....2034
48.5.13 Power saving features....2039
48.6 Initialization/application information....2040
48.6.1 How to manage queues....2040
48.6.2 Switching Master and Slave mode....2041
48.6.3 Initializing Module in Master/Slave Modes....2041
48.6.4 Baud rate settings....2041
48.6.5 Delay settings....2042
48.6.6 Calculation of FIFO pointer addresses....2043

Chapter 49
Zipwire

49.1 Introduction....2047
49.2 Block Diagram....2047
49.3 Architecture....2048
49.4 LFAST Clocking....2049
49.5 Zipwire interconnections....2050

49.6 Zipwire performance....2051
49.6.1 Read performance....2051
49.6.2 Write performance....2053
49.7 LFAST/SIPI configuration....2055

Chapter 50
Serial Interprocessor Interface (SIPI)

50.1 Introduction....2057
50.1.1 Scalability....2057
50.2 Overview....2058
50.3 SIPI block diagram....2060
50.4 Feature description....2060
50.4.1 Main features....2060
50.4.2 Standard features....2061
50.5 SIPI operation from reset....2061
50.6 Functional description....2061
50.6.1 External signals....2061
50.6.2 Frame format....2062
50.7 Transfer types....2068
50.7.1 Read transfer....2068
50.7.2 Register read answer transfer....2069
50.7.3 Register Write transfer....2070
50.7.4 Write Acknowledge transfer....2073
50.7.5 ID request response....2073
50.8 Transfer API and flow charts....2075
50.9 DMA programming sequence....2082
50.10 Modes of operation....2083
50.10.1 Initialization mode....2083
50.10.2 Normal mode....2083
50.10.3 Module Disable (MD)....2084

50.11 Errors....2084
50.11.1 Timeout error....2084
50.11.2 CRC error....2084
50.11.3 Maximum count reached error....2085
50.11.4 Transaction ID error....2085
50.11.5 Acknowledge error....2085
50.12 CRC calculation....2085
50.13 Interrupt logic....2086
50.14 SIPI control and status overview....2087
50.15 Memory map and register definition....2088
50.15.1 SIPI Channel Control Register 0 (SIPI_CCR0)....2091
50.15.2 SIPI Channel Status Register 0 (SIPI_CSR0)....2094
50.15.3 SIPI Channel Interrupt Register 0 (SIPI_CIR0)....2095
50.15.4 SIPI Channel Timeout Register 0 (SIPI_CTOR0)....2096
50.15.5 SIPI Channel CRC Register 0 (SIPI_CCRC0)....2097
50.15.6 SIPI Channel Address Register 0 (SIPI_CAR0)....2097
50.15.7 SIPI Channel Data Register 0 (SIPI_CDR0)....2098
50.15.8 SIPI Channel Control Register 1 (SIPI_CCR1)....2098
50.15.9 SIPI Channel Status Register 1 (SIPI_CSR1)....2101
50.15.10 SIPI Channel Interrupt Register 1 (SIPI_CIR1)....2103
50.15.11 SIPI Channel Timeout Register 1 (SIPI_CTOR1)....2104
50.15.12 SIPI Channel CRC Register 1 (SIPI_CCRC1)....2105
50.15.13 SIPI Channel Address Register 1 (SIPI_CAR1)....2105
50.15.14 SIPI Channel Data Register 1 (SIPI_CDR1)....2106
50.15.15 SIPI Channel Control Register 2 (SIPI_CCR2)....2106
50.15.16 SIPI Channel Status Register 2 (SIPI_CSR2)....2109
50.15.17 SIPI Channel Interrupt Register 2 (SIPI_CIR2)....2111
50.15.18 SIPI Channel Timeout Register 2 (SIPI_CTOR2)....2112
50.15.19 SIPI Channel CRC Register 2 (SIPI_CCRC2)....2113

50.15.20 SIPI Channel Address Register 2 (SIPI_CAR2)....2113
50.15.21 SIPI Channel Data Register 2 (SIPI_CDR2_*n*)....2114
50.15.22 SIPI Channel Control Register 3 (SIPI_CCR3)....2114
50.15.23 SIPI Channel Status Register 3 (SIPI_CSR3)....2118
50.15.24 SIPI Channel Interrupt Register 3 (SIPI_CIR3)....2119
50.15.25 SIPI Channel Timeout Register 3 (SIPI_CTOR3)....2120
50.15.26 SIPI Channel CRC Register 3 (SIPI_CCRC3)....2121
50.15.27 SIPI Channel Address Register 3 (SIPI_CAR3)....2121
50.15.28 SIPI Channel Data Register 3 (SIPI_CDR3)....2122
50.15.29 SIPI Module Configuration Register (SIPI_MCR)....2123
50.15.30 SIPI Status Register (SIPI_SR)....2126
50.15.31 SIPI Max Count Register (SIPI_MAXCR)....2128
50.15.32 SIPI Address Reload Register (SIPI_ARR)....2128
50.15.33 SIPI Address Count Register (SIPI_ACR)....2129
50.15.34 SIPI Error Register (SIPI_ERR)....2130

Chapter 51
LVDS Fast Asynchronous Serial Transmission (LFAST) – Interprocessor Communications

51.1 Introduction....2133
51.2 Block diagram....2133
51.3 External signals....2134
51.3.1 LFAST operating data rates....2134
51.4 LFAST frame structure....2134
51.5 Features....2137
51.6 Memory map and register definition....2138
51.6.1 LFAST Mode Configuration Register (LFAST_MCR)....2140
51.6.2 LFAST Speed Control Register (LFAST_SCR)....2142
51.6.3 LFAST Correlator Control Register (LFAST_COCR)....2143
51.6.4 LFAST Test Mode Control Register (LFAST_TMCR)....2145
51.6.5 LFAST Auto Loopback Control Register (LFAST_ALCR)....2146

51.6.6 LFAST Rate Change Delay Control Register (LFAST_RCDCR)....2147
51.6.7 LFAST Wakeup Delay Control Register (LFAST_SLCR)....2147
51.6.8 LFAST ICLC Control Register (LFAST_ICR)....2149
51.6.9 LFAST Ping Control Register (LFAST_PICR)....2150
51.6.10 LFAST Rx FIFO CTS Control Register (LFAST_RFCR)....2150
51.6.11 LFAST Tx Interrupt Enable Register (LFAST_TIER)....2151
51.6.12 LFAST Rx Interrupt Enable Register (LFAST_RIER)....2152
51.6.13 LFAST Rx ICLC Interrupt Enable Register (LFAST_RIIER)....2154
51.6.14 LFAST PLL Control Register (LFAST_PLLCR)....2156
51.6.15 LFAST LVDS Control Register (LFAST_LCR)....2158
51.6.16 LFAST Unsolicited Tx Control Register (LFAST_UNSTCR)....2161
51.6.17 LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR*n*)....2161
51.6.18 LFAST Global Status Register (LFAST_GSR)....2162
51.6.19 LFAST Ping Status Register (LFAST_PISR)....2163
51.6.20 LFAST Data Frame Status Register (LFAST_DFSR)....2164
51.6.21 LFAST Tx Interrupt Status Register (LFAST_TISR)....2165
51.6.22 LFAST Rx Interrupt Status Register (LFAST_RISR)....2166
51.6.23 LFAST Rx ICLC Interrupt Status Register (LFAST_RIISR)....2168
51.6.24 LFAST PLL and LVDS Status Register (LFAST_PLLLSR)....2170
51.6.25 LFAST Unsolicited Rx Status Register (LFAST_UNSRSR)....2171
51.6.26 LFAST Unsolicited Rx Data Register (LFAST_UNSRDR*n*)....2172
51.7 Functional description....2172
51.7.1 Startup procedure....2172
51.7.2 Line Receiver....2176
51.7.3 Transmit Controller....2185
51.7.4 CTS mode support....2190
51.7.5 Frames supported....2191

51.7.6 Frame flow....2192
51.7.7 Test and Debug Support....2199
51.7.8 Interrupts....2208
51.8 Packet memory....2211
51.9 Resets....2212
51.10 Clocks....2213
51.10.1 Clocking strategy....2213
51.10.2 Slow speed clock....2214
51.10.3 Rx Controller Clocks....2217
51.10.4 Clocking Module Requirements for High Speed Phases....2217
51.10.5 Clock module requirements for low speed phases....2218
51.10.6 Tx Controller Clocks....2219

Chapter 52
Fast Ethernet Controller (FEC)

52.1 Fast Ethernet Controller (FEC) chip-specific module information....2221
52.1.1 FEC interface selection....2221
52.2 Introduction....2221
52.2.1 Block diagram....2221
52.2.2 Features....2223
52.2.3 Clocking requirements....2224
52.3 Modes of operation....2225
52.3.1 Full and half duplex operation....2225
52.3.2 Interface options....2225
52.3.3 Address recognition options....2227
52.3.4 Internal loopback....2227
52.4 External signal description....2227
52.5 Memory map and register definition....2229
52.5.1 Interrupt Event Register (FEC_EIR)....2233
52.5.2 Interrupt Mask Register (FEC_EIMR)....2235

52.5.3 Receive Descriptor Active Register (FEC_RDAR)....2237
52.5.4 Transmit Descriptor Active Register (FEC_TDAR)....2238
52.5.5 Ethernet Control Register (FEC_ECR)....2239
52.5.6 MII Management Frame Register (FEC_MMFR)....2240
52.5.7 MII Speed Control Register (FEC_MSCR)....2241
52.5.8 MIB Control Register (FEC_MIBC)....2242
52.5.9 Receive Control Register (FEC_RCR)....2244
52.5.10 Transmit Control Register (FEC_TCR)....2246
52.5.11 Physical Address Low Register (FEC_PALR)....2247
52.5.12 Physical Address High Register and Type Field (FEC_PAUR)....2248
52.5.13 Opcode/Pause Duration (FEC_OPD)....2248
52.5.14 Descriptor Individual Upper Address Register (FEC_IAUR)....2249
52.5.15 Descriptor Individual Lower Address Register (FEC_IALR)....2249
52.5.16 Descriptor Group Upper Address Register (FEC_GAUR)....2250
52.5.17 Descriptor Group Lower Address Register (FEC_GALR)....2250
52.5.18 Transmit FIFO Watermark (FEC_TFWR)....2251
52.5.19 FIFO Receive Bound Register (FEC_FRBR)....2252
52.5.20 FIFO Receive Start Register (FEC_FRSR)....2252
52.5.21 Receive Descriptor Ring Start Register (FEC_ERDSR)....2253
52.5.22 Transmit Buffer Descriptor Ring Start Register (FEC_ETDSR)....2254
52.5.23 Receive Buffer Size Register (FEC_EMRBR)....2254
52.5.24 Count of frames not counted correctly (FEC_RMON_T_DROP)....2255
52.5.25 RMON Tx packet count (FEC_RMON_T_PACKETS)....2256
52.5.26 RMON Tx broadcast packets (FEC_RMON_T_BC_PKT)....2256
52.5.27 RMON Tx multicast packets (FEC_RMON_T_MC_PKT)....2257
52.5.28 RMON Tx packets with CRC/align error (FEC_RMON_T_CRC_ALIGN)....2257
52.5.29 RMON Tx packets < 64 bytes, good CRC (FEC_RMON_T_UNDERSIZE)....2258
52.5.30 RMON Tx packets > MAX_FL bytes, good CRC (FEC_RMON_T_OVERSIZE)....2258
52.5.31 RMON Tx packets < 64 bytes, bad CRC (FEC_RMON_T_FRAG)....2259

52.5.32 RMON Tx packets > MAX_FL bytes, bad CRC (FEC_RMON_T_JAB).....2259
52.5.33 RMON Tx collision count (FEC_RMON_T_COL).....2260
52.5.34 RMON Tx 64 byte packets (FEC_RMON_T_P64).....2260
52.5.35 RMON Tx 65 to 127 byte packets (FEC_RMON_T_P65TO127).....2261
52.5.36 RMON Tx 128 to 255 byte packets (FEC_RMON_T_P128TO255).....2261
52.5.37 RMON Tx 256 to 511 byte packets (FEC_RMON_T_P256TO511).....2262
52.5.38 RMON Tx 512 to 1023 byte packets (FEC_RMON_T_P512TO1023).....2262
52.5.39 RMON Tx 1024 to 2047 byte packets (FEC_RMON_T_P1024TO2047).....2263
52.5.40 RMON Tx packets with > 2048 bytes (FEC_RMON_T_P_GTE2048).....2263
52.5.41 RMON Tx Octets (FEC_RMON_T_OCTETS).....2264
52.5.42 Count of transmitted frames not counted correctly (FEC_IEEE_T_DROP).....2264
52.5.43 Frames transmitted OK (FEC_IEEE_T_FRAME_OK).....2265
52.5.44 Frames transmitted with single collision (FEC_IEEE_T_1COL).....2265
52.5.45 Frames transmitted with multiple collisions (FEC_IEEE_T_MCOL).....2266
52.5.46 Frames transmitted after deferral delay (FEC_IEEE_T_DEF).....2266
52.5.47 Frames transmitted with late collision (FEC_IEEE_T_LCOL).....2267
52.5.48 Frames transmitted with excessive collisions (FEC_IEEE_T_EXCOL).....2267
52.5.49 Frames transmitted with Tx FIFO underrun (FEC_IEEE_T_MACERR).....2268
52.5.50 Frames transmitted with carrier sense error (FEC_IEEE_T_CSERR).....2268
52.5.51 Frames transmitted with SQE error (FEC_IEEE_T_SQE).....2269
52.5.52 Flow control pause frames transmitted (FEC_IEEE_T_FDXFC).....2269
52.5.53 Octet count for frames transmitted without error (FEC_IEEE_T_OCTETS_OK).....2270
52.5.54 Count of received frames not counted correctly (FEC_RMON_R_DROP).....2270
52.5.55 RMON Rx packet count (FEC_RMON_R_PACKETS).....2271
52.5.56 RMON Rx broadcast packets (FEC_RMON_R_BC_PKT).....2271
52.5.57 RMON Rx multicast packets (FEC_RMON_R_MC_PKT).....2272
52.5.58 RMON Rx packets with CRC/Align error (FEC_RMON_R_CRC_ALIGN).....2272
52.5.59 RMON Rx packets < 64 bytes, good CRC (FEC_RMON_R_UNDERSIZE).....2273
52.5.60 RMON Rx packets > MAX_FL bytes, good CRC (FEC_RMON_R_OVERSIZE).....2273

52.5.61 RMON Rx packets < 64 bytes, bad CRC (FEC_RMON_R_FRAG)....2274
52.5.62 RMON Rx packets > MAX_FL bytes, bad CRC (FEC_RMON_R_JAB)....2274
52.5.63 Reserved (FEC_RMON_R_RESVD_0)....2275
52.5.64 RMON Rx 64 byte packets (FEC_RMON_R_P64)....2275
52.5.65 RMON Rx 65 to 127 byte packets (FEC_RMON_R_P65TO127)....2276
52.5.66 RMON Rx 128 to 255 byte packets (FEC_RMON_R_P128TO255)....2276
52.5.67 RMON Rx 256 to 511 byte packets (FEC_RMON_R_P256TO511)....2277
52.5.68 RMON Rx 512 to 1023 byte packets (FEC_RMON_R_P512TO1023)....2277
52.5.69 RMON Rx 1024 to 2047 byte packets (FEC_RMON_R_P1024TO2047)....2278
52.5.70 RMON Rx packets with > 2048 bytes (FEC_RMON_R_P_GTE2048)....2278
52.5.71 RMON Rx octets (FEC_RMON_R_OCTETS)....2279
52.5.72 Count of received frames not counted correctly (FEC_IEEE_R_DROP)....2279
52.5.73 Frames received OK (FEC_IEEE_R_FRAME_OK)....2280
52.5.74 Frames received with CRC error (FEC_IEEE_R_CRC)....2280
52.5.75 Frames received with alignment error (FEC_IEEE_R_ALIGN)....2281
52.5.76 Receive FIFO overflow count (FEC_IEEE_R_MACERR)....2281
52.5.77 Flow control pause frames received (FEC_IEEE_R_FDXFC)....2282
52.5.78 Octet count for frames received without error (FEC_IEEE_R_OCTETS_OK)....2282
52.5.79 Using the RMON and IEEE registers....2282
52.6 Functional description....2283
52.6.1 MII data frame....2283
52.6.2 MII management frame structure....2284
52.6.3 Buffer descriptors....2285
52.6.4 Initialization sequence....2291
52.6.5 User initialization (Prior to Setting ECR[ETHER_EN])....2291
52.6.6 Microcontroller initialization....2292
52.6.7 User initialization (after setting ECR[ETHER_EN])....2293
52.6.8 Network interface options....2293

52.6.9 FEC frame transmission....2294
52.6.10 FEC frame reception....2296
52.6.11 Ethernet address recognition....2297
52.6.12 Hash algorithm....2300
52.6.13 Full duplex flow control....2303
52.6.14 Inter-Packet Gap (IPG) time....2304
52.6.15 Collision managing....2304
52.6.16 MII internal and external loopback....2304
52.6.17 RMII loopback....2305
52.6.18 RMII echo....2305
52.6.19 Ethernet error-managing procedure....2305

Chapter 53
SENT Receiver (SRX)

53.1 SENT chip-specific modules information....2309
53.1.1 SRX_GBL_CTRL[NIB_LEN_VAR_LIMIT] setting....2309
53.2 Introduction....2310
53.2.1 Features....2310
53.2.2 Modes of operation....2311
53.2.3 Block diagram....2314
53.2.4 Design overview....2316
53.3 External signal description....2317
53.3.1 SENT_RX [(CH-1):0]....2317
53.4 Memory map and register definition....2318
53.4.1 Global Control Register (SRX_GBL_CTRL)....2322
53.4.2 Channel Enable Register (SRX_CHNL_EN)....2323
53.4.3 Global Status Register (SRX_GBL_STATUS)....2325
53.4.4 Fast Message Ready Status Register (SRX_FMSG_RDY)....2326
53.4.5 Slow Serial Message Ready Status Register (SRX_SMSG_RDY)....2327
53.4.6 Data Control Register 1 (SRX_DATA_CTRL1)....2328

53.4.7 Fast Message DMA Control Register (SRX_FDMA_CTRL)....2330
53.4.8 Slow Serial Message DMA Control Register (SRX_SDMA_CTRL)....2330
53.4.9 Fast Message Ready Interrupt Control Register (SRX_FRDY_IE)....2331
53.4.10 Slow Serial Message Ready Interrupt Enable Register (SRX_SRDY_IE)....2332
53.4.11 DMA Fast Message Data Read Register (SRX_DMA_FMSG_DATA)....2332
53.4.12 DMA Fast Message CRC Read Register (SRX_DMA_FMSG_CRC)....2333
53.4.13 DMA Fast Message Time Stamp Read Register (SRX_DMA_FMSG_TS)....2334
53.4.14 DMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3)....2334
53.4.15 DMA Slow Serial Message Bit2 Read Register (SRX_DMA_SMSG_BIT2)....2335
53.4.16 DMA Slow Serial Message Time Stamp Read Register (SRX_DMA_SMSG_TS)....2336
53.4.17 Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH*n*_CLK_CTRL)....2337
53.4.18 Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH*n*_STATUS)....2339
53.4.19 Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH*n*_CONFIG)....2342
53.4.20 Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH*n*_FMSG_DATA)....2345
53.4.21 Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH*n*_FMSG_CRC)....2346
53.4.22 Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH*n*_FMSG_TS)....2346
53.4.23 Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH*n*_SMSG_BIT3)....2347
53.4.24 Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH*n*_SMSG_BIT2)....2348
53.4.25 Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH*n*_SMSG_TS)....2349
53.5 Functional description....2349
53.5.1 Initialization sequence....2349
53.5.2 DMA read logic....2350
53.5.3 Message reading via interrupts....2355
53.5.4 Overflow behavior....2357
53.5.5 Adjustment for variation in sensor (Tx) clock....2358
53.5.6 Input programmable filter....2359
53.5.7 Receiver diagnostics....2360
53.5.8 Time stamp logic....2365
53.5.9 Bus Idle Diagnostic....2366

53.6 Clocks and resets....2367
53.6.1 Clocking strategy....2367
53.6.2 System bus clock requirements....2368
53.6.3 High frequency receiver clock (protocol clock) requirements....2369

Chapter 54
LINFlexD

54.1 LinFlexD chip-specific module information....2371
54.2 Introduction....2372
54.2.1 Glossary and acronyms....2372
54.2.2 References....2372
54.3 Main features....2374
54.3.1 LIN mode features....2374
54.3.2 UART mode features....2375
54.4 Functional description....2376
54.4.1 LIN protocol....2376
54.4.2 LINFlexD features....2378
54.4.3 Timer....2398
54.4.4 UART mode....2398
54.4.5 DMA interface....2404
54.5 Memory map and register description....2423
54.5.1 LIN Control Register 1 (LINFlexD_LINCR1)....2425
54.5.2 LIN Interrupt enable register (LINFlexD_LINIER)....2428
54.5.3 LIN Status Register (LINFlexD_LINSR)....2430
54.5.4 LIN Error Status Register (LINFlexD_LINESR)....2434
54.5.5 UART Mode Control Register (LINFlexD_UARTCR)....2435
54.5.6 UART Mode Status Register (LINFlexD_UARTSR)....2441
54.5.7 LIN Time-Out Control Status Register (LINFlexD_LINTCSR)....2444
54.5.8 LIN Output Compare Register (LINFlexD_LINOCR)....2445
54.5.9 LIN Time-Out Control Register (LINFlexD_LINTOCR)....2446

54.5.10 LIN Fractional Baud Rate Register (LINFlexD_LINFBRR)....2446
54.5.11 LIN Integer Baud Rate Register (LINFlexD_LINIBRR)....2447
54.5.12 LIN Checksum Field Register (LINFlexD_LINCFR)....2448
54.5.13 LIN Control Register 2 (LINFlexD_LINCR2)....2449
54.5.14 Buffer Identifier Register (LINFlexD_BIDR)....2451
54.5.15 Buffer Data Register Least Significant (LINFlexD_BDRL)....2452
54.5.16 Buffer Data Register Most Significant (LINFlexD_BDRM)....2453
54.5.17 Identifier Filter Enable Register (LINFlexD_IFER)....2453
54.5.18 Identifier Filter Match Index (LINFlexD_IFMI)....2454
54.5.19 Identifier Filter Mode Register (LINFlexD_IFMR)....2454
54.5.20 Identifier Filter Control Register (LINFlexD_IFCR*n*)....2455
54.5.21 Global Control Register (LINFlexD_GCR)....2456
54.5.22 UART Preset Timeout Register (LINFlexD_UARTPTO)....2458
54.5.23 UART Current Timeout Register (LINFlexD_UARTCTO)....2459
54.5.24 DMA Tx Enable Register (LINFlexD_DMATXE)....2460
54.5.25 DMA Rx Enable Register (LINFlexD_DMARXE)....2460
54.6 Programming considerations....2461
54.6.1 Master node....2461
54.6.2 Slave node....2463
54.6.3 Extended frames....2468
54.6.4 Timeout....2469
54.6.5 UART mode....2470
54.6.6 Interrupts....2470
54.6.7 LINFlexD Clock Tolerance....2472

Chapter 55
Reset and Boot

55.1 Introduction....2475
55.2 Flash memory boot overview....2475
55.3 Device Configuration Format (DCF) records....2476

55.4 Reset sequence....2477
55.4.1 RSTOUT functionality....2477
55.4.2 POWERUP Phase: power stabilization....2478
55.4.3 PHASE0 Phase: analog supply initial configuration....2478
55.4.4 PHASE1[DEST] Phase: temporization and monitoring setup....2478
55.4.5 PHASE2[DEST] Phase: flash initial configuration....2479
55.4.6 PHASE3[DEST] Phase: device configuration....2479
55.4.7 IDLE[DEST] Phase: self-test execution....2480
55.4.8 PHASE1[FUNC] Phase: temporization and monitoring setup....2480
55.4.9 PHASE2[FUNC] Phase: flash initial configuration....2480
55.4.10 PHASE3[FUNC] Phase: device configuration monitoring....2481
55.4.11 IDLE[FUNC] Phase....2481
55.5 Module status during reset sequence....2482

Chapter 56
Reset Generation Module (MC_RGM)

56.1 Introduction....2485
56.1.1 Overview....2485
56.1.2 Features....2486
56.1.3 Reset Sources....2487
56.2 External Signal Description....2488
56.3 MC_RGM Registers....2489
56.3.1 ‘Destructive’ Event Status (MC_RGM_DES)....2490
56.3.2 ‘Destructive’ Event Reset Disable (MC_RGM_DERD)....2492
56.3.3 ‘Destructive’ Event Alternate Request (MC_RGM_DEAR)....2494
56.3.4 ‘Destructive’ Bidirectional Reset Enable (MC_RGM_DBRE)....2496
56.3.5 ‘Functional’ Event Status (MC_RGM_FES)....2498
56.3.6 ‘Functional’ Event Reset Disable (MC_RGM_FERD)....2500
56.3.7 ‘Functional’ Event Alternate Request (MC_RGM_FEAR)....2501
56.3.8 ‘Functional’ Bidirectional Reset Enable (MC_RGM_FBRE)....2502

56.3.9 'Functional' Event Short Sequence (MC_RGM_FESS).....2504
56.3.10 Functional Reset Threshold (MC_RGM_FRET).....2506
56.3.11 'Destructive' Reset Threshold (MC_RGM_DRET).....2507
56.4 Functional Description.....2508
56.4.1 Reset State Machine.....2508
56.4.2 'Destructive' Resets.....2511
56.4.3 External Reset.....2511
56.4.4 'Functional' Resets.....2512
56.4.5 Alternate Event Generation.....2513
56.4.6 'Functional' Reset Escalation.....2514
56.4.7 'Destructive' Reset Escalation.....2515

Chapter 57
Boot Assist Flash (BAF)

57.1 BAF chip-specific module information.....2517
57.1.1 Search boot header and boot options.....2517
57.1.2 LinFLEXD_1 and FlexCAN_1 pin configuration.....2518
57.1.3 BAF memory map.....2519
57.1.4 Reset values on BAF exit.....2520
57.1.5 Serial boot password behavior.....2520
57.2 Introduction.....2520
57.3 BAF image header.....2521
57.3.1 BAF image version.....2521
57.4 Functional description.....2521
57.4.1 Boot modes.....2521
57.4.2 Device initialization.....2522
57.4.3 Flow of control.....2522
57.4.4 Optionally remap calibration RAM.....2523
57.4.5 BAF SAFE mode event handling.....2524
57.4.6 BAF NMI Event Handling.....2525

57.4.7 Initialization of BAF DCF clients....2526
57.4.8 Optionally perform serial boot....2535
57.4.9 Serial boot configuration....2540
57.5 Resources....2543

Chapter 58
System Status and Configuration Module (SSCM)

58.1 System Status and Configuration Module (SSCM) chip-specific module information....2545
58.2 Introduction....2546
58.2.1 Overview....2546
58.2.2 Glossary....2547
58.2.3 Features....2547
58.2.4 Modes of operation....2548
58.3 External signal description....2548
58.4 Memory map and register definition....2548
58.4.1 SSCM System Status (SSCM_STATUS)....2549
58.4.2 SSCM System Memory and ID Register (SSCM_MEMCONFIG)....2551
58.4.3 Processor Start Address Register (SSCM_PSA)....2551
58.4.4 Life Cycle Status Register (SSCM_LCSTAT)....2552
58.5 Functional description....2553
58.5.1 CERS-DCF Mechanism....2553
58.5.2 ECC error monitoring....2554
58.5.3 Boot mode functionality....2554
58.5.4 BAF configuration....2554
58.5.5 Life Cycle....2554
58.6 Initialization and application information....2556
58.6.1 Reset....2556
58.7 Additional safety measures....2556
58.7.1 Spurious reset protection....2556

Chapter 59
Power Management

59.1 Overview....2559
59.1.1 Power Management Controller (PMC)....2560
59.1.2 Supply Concept....2560
59.1.3 Power Modes....2561
59.1.4 External Supply....2562
59.1.5 Voltage Monitoring....2562
59.1.6 PMC Power-on Reset....2564
59.1.7 SoC Power-on Reset....2565
59.1.8 Power-up Sequence....2565
59.1.9 ADC Interface....2567
59.1.10 Safety Measures....2569

Chapter 60
Power Management Controller digital interface (PMC_dig)

60.1 PMC_dig chip-specific module information....2571
60.1.1 PMC registers whose reset values can be defined by DCF clients....2571
60.1.2 PMC registers with chip-specific reset values....2571
60.1.3 Self test bit definition....2572
60.2 PMC introduction....2572
60.2.1 Features....2573
60.3 Memory Map and Registers....2573
60.3.1 Temperature Sensor Interrupt Enable Register (PMC_TS_IER)....2575
60.3.2 Temperature Event Status register (PMC_ESR_TD)....2577
60.3.3 Temperature Reset Event Enable register (PMC_REE_TD)....2579
60.3.4 Temperature Reset Event Selection register (PMC_RES_TD)....2580
60.3.5 Temperature detector configuration register (PMC_CTL_TD)....2582
60.3.6 Temperature Fault Injection Register (PMC_TS_FIR)....2583
60.3.7 Voltage Detect User Mode Test Register (PMC_SELF_TEST_UM_VD_REG)....2584

60.3.8 FCCU Fault Injection Register (PMC_FIR)....2586
60.3.9 LVD HVD Interrupt Enable Control Register (PMC_LVD_HVD_INTR_ENB)....2587
60.3.10 LVD HVD Event Status Register (PMC_LVD_HVD_EVENT_STATUS)....2590
60.3.11 Reset Enable Control Register (PMC_REE)....2592
60.3.12 Reset Type (Destructive / Functional) Selection Register (PMC_RES)....2595
60.3.13 ADC Channel Select Register (PMC_ADC_CS)....2599
60.4 Functional description....2600
60.4.1 Analog PMC interface....2600
60.4.2 Temperature sensor interface logic....2600
60.4.3 POR RGM phase gates....2600
60.4.4 Flash interface to DCF....2601
60.4.5 FCCU interface....2601
60.4.6 LVD-HVD self test mechanism....2602

Chapter 61
Power Control Unit (MC_PCU)

61.1 Introduction....2607
61.2 Memory map and register definition....2607
61.2.1 Power Domain Status Register (MC_PCU_PSTAT)....2608
61.3 Functional description....2608

Chapter 62
Mode Entry Module (MC_ME)

62.1 Introduction....2609
62.1.1 Overview....2609
62.1.2 Features....2610
62.1.3 Modes of Operation....2611
62.2 External Signal Description....2613
62.3 MC_ME Registers....2613
62.3.1 Global Status (MC_ME_GS)....2618
62.3.2 Mode Control (MC_ME_MCTL)....2620

62.3.3 Mode Enable (MC_ME_ME)....2622
62.3.4 Interrupt Status (MC_ME_IS)....2624
62.3.5 Interrupt Mask (MC_ME_IM)....2625
62.3.6 Invalid Mode Transition Status (MC_ME_IMTS)....2627
62.3.7 Debug Mode Transition Status (MC_ME_DMTS)....2628
62.3.8 RESET Mode Configuration (MC_ME_RESET_MC)....2633
62.3.9 TEST Mode Configuration (MC_ME_TEST_MC)....2635
62.3.10 SAFE Mode Configuration (MC_ME_SAFE_MC)....2636
62.3.11 DRUN Mode Configuration (MC_ME_DRUN_MC)....2638
62.3.12 RUN0 Mode Configuration (MC_ME_RUN0_MC)....2640
62.3.13 RUN1 Mode Configuration (MC_ME_RUN1_MC)....2641
62.3.14 RUN2 Mode Configuration (MC_ME_RUN2_MC)....2643
62.3.15 RUN3 Mode Configuration (MC_ME_RUN3_MC)....2645
62.3.16 HALT0 Mode Configuration (MC_ME_HALT0_MC)....2646
62.3.17 STOP0 Mode Configuration (MC_ME_STOP0_MC)....2648
62.3.18 Peripheral Status 0 (MC_ME_PS0)....2650
62.3.19 Peripheral Status 1 (MC_ME_PS1)....2652
62.3.20 Peripheral Status 2 (MC_ME_PS2)....2654
62.3.21 Peripheral Status 3 (MC_ME_PS3)....2656
62.3.22 Peripheral Status 4 (MC_ME_PS4)....2658
62.3.23 Peripheral Status 5 (MC_ME_PS5)....2659
62.3.24 Peripheral Status 6 (MC_ME_PS6)....2661
62.3.25 Peripheral Status 7 (MC_ME_PS7)....2663
62.3.26 Run Peripheral Configuration 0 (MC_ME_RUN_PC*n*)....2665
62.3.27 Low-Power Peripheral Configuration 0 (MC_ME_LP_PC*n*)....2666
62.3.28 LFAST_0 Control (MC_ME_PCTL9)....2667
62.3.29 SIPI_0 Control (MC_ME_PCTL11)....2668
62.3.30 DECFILT_1 Control (MC_ME_PCTL28)....2669
62.3.31 DECFILT_0 Control (MC_ME_PCTL29)....2669

62.3.32 PIT_RTI_0 Control (MC_ME_PCTL30)....2670
62.3.33 PIT_RTI_1 Control (MC_ME_PCTL31)....2670
62.3.34 DMA_CH_MUX_0-2 Control (MC_ME_PCTL36)....2671
62.3.35 CRC_0 Control (MC_ME_PCTL38)....2671
62.3.36 JTAGM Control (MC_ME_PCTL45)....2672
62.3.37 ADCSD_2 Control (MC_ME_PCTL59)....2672
62.3.38 ADCSD_0 Control (MC_ME_PCTL60)....2673
62.3.39 ADCSAR_2 Control (MC_ME_PCTL69)....2673
62.3.40 ADCSAR_0 Control (MC_ME_PCTL71)....2674
62.3.41 FLEXCAN_2 Control (MC_ME_PCTL81)....2674
62.3.42 FLEXCAN_0 Control (MC_ME_PCTL83)....2675
62.3.43 LINFLEX_M0 Control (MC_ME_PCTL85)....2675
62.3.44 LINFLEX_2 Control (MC_ME_PCTL91)....2676
62.3.45 LINFLEX_0 Control (MC_ME_PCTL92)....2677
62.3.46 DSPI_M0 Control (MC_ME_PCTL96)....2677
62.3.47 DSPI_4 Control (MC_ME_PCTL97)....2678
62.3.48 DSPI_2 Control (MC_ME_PCTL98)....2678
62.3.49 DSPI_0 Control (MC_ME_PCTL99)....2679
62.3.50 EMIOS Control (MC_ME_PCTL102)....2680
62.3.51 SENT_0 Control (MC_ME_PCTL104)....2680
62.3.52 DMA_CH_MUX_3 Control (MC_ME_PCTL143)....2681
62.3.53 CRC_1 Control (MC_ME_PCTL166)....2681
62.3.54 ADCSD_1 Control (MC_ME_PCTL188)....2682
62.3.55 ADCSAR_3 Control (MC_ME_PCTL196)....2682
62.3.56 IGF Control (MC_ME_PCTL197)....2683
62.3.57 ADCSAR_1 Control (MC_ME_PCTL198)....2683
62.3.58 ETPU_SCRAM Control (MC_ME_PCTL202)....2684
62.3.59 ETPU_SCRAM Control (MC_ME_PCTL203)....2685
62.3.60 ETPU_SPRAM Control (MC_ME_PCTL204)....2685

62.3.61 ETPU_SPRAM Control (MC_ME_PCTL205)....2686
62.3.62 REACM Control (MC_ME_PCTL206)....2686
62.3.63 ETPU Control (MC_ME_PCTL207)....2687
62.3.64 FLEXCAN_3 Control (MC_ME_PCTL208)....2688
62.3.65 FLEXCAN_1 Control (MC_ME_PCTL210)....2688
62.3.66 LINFLEX_M1 Control (MC_ME_PCTL213)....2689
62.3.67 LINFLEX_3 Control (MC_ME_PCTL219)....2689
62.3.68 LINFLEX_1 Control (MC_ME_PCTL220)....2690
62.3.69 DSPI_M1 Control (MC_ME_PCTL225)....2690
62.3.70 DSPI_3 Control (MC_ME_PCTL226)....2691
62.3.71 DSPI_1 Control (MC_ME_PCTL227)....2691
62.3.72 EMIOS_1 Control (MC_ME_PCTL230)....2692
62.3.73 SENT_1 Control (MC_ME_PCTL232)....2692
62.3.74 BCTU Control (MC_ME_PCTL251)....2693
62.3.75 Core Status (MC_ME_CS)....2693
62.3.76 CORE1 Control (MC_ME_CCTL0)....2695
62.3.77 CORE0 Control (MC_ME_CCTL1)....2696
62.3.78 CORE2 Control (MC_ME_CCTL2)....2697
62.3.79 CORE1 Address (MC_ME_CADDR0)....2699
62.3.80 CORE0 Address (MC_ME_CADDR1)....2700
62.3.81 CORE2 Address (MC_ME_CADDR2)....2700
62.4 Functional Description....2701
62.4.1 Mode Transition Request....2701
62.4.2 Modes Details....2703
62.4.3 Mode Transition Process....2708
62.4.4 Software Considerations for Preventing the Blocking of Mode Transitions....2717
62.4.5 Protection of Mode Configuration Registers....2720
62.4.6 Mode Transition Interrupts....2721
62.4.7 Peripheral Clock Gating....2724

62.4.8 Initialization / Application Information....2725

Chapter 63
Calibration and Debug

63.1 Introduction....2727
63.2 Core debug support....2727
63.3 Run control and memory access....2728
63.3.1 Debug and Calibration Interface (DCI)....2729
63.3.2 JTAG Data Communication (JDC)....2734
63.3.3 Sequence Processing Unit (SPU)....2734
63.4 Calibration interface....2751
63.4.1 JTAG Master (JTAGM)....2751
63.4.2 Development Trigger Semaphore (DTS)....2752
63.5 Flash calibration support....2752
63.6 Nexus Trace Aurora interface....2753
63.6.1 Nexus Aurora overview....2755
63.6.2 Nexus Aurora Router (NAR)....2756
63.6.3 Nexus Aurora Link (NAL)....2759
63.6.4 Nexus Aurora PHY (NAP)....2759
63.7 Nexus clients....2761
63.7.1 Enhanced Time Processing Unit (eTPU)....2761
63.7.2 e200z4 Nexus 3....2761
63.7.3 Nexus Crossbar Multi-master Client (NXMC)....2762

Chapter 64
Core Debug Support

64.1 Overview....2765
64.1.1 Software Debug Facilities....2765
64.1.2 Additional Debug Facilities....2766
64.1.3 Hardware Debug Facilities....2767

64.1.4 Sharing Debug Resources by Software/Hardware....2768
64.2 Software Debug Events and Exceptions....2769
64.2.1 Instruction Address Compare Event....2771
64.2.2 Data Address Compare Event....2771
64.2.3 Linked Instruction Address Compare and Data Address Compare Events....2781
64.2.4 Trap Debug Event....2782
64.2.5 Branch Taken Debug Event....2782
64.2.6 Instruction Complete Debug Event....2783
64.2.7 Interrupt Taken Debug Event....2783
64.2.8 Critical Interrupt Taken Debug Event....2784
64.2.9 Return Debug Event....2784
64.2.10 Critical Return Debug Event....2784
64.2.11 External debug event....2785
64.2.12 Unconditional debug event....2785
64.2.13 Performance Monitor Interrupt debug event....2785
64.3 Debug registers....2786
64.3.1 Debug Address and Value Registers....2786
64.3.2 Debug Control and Status registers....2787
64.3.3 External Debug Resource Allocation Control Register (EDBRAC0)....2812
64.3.4 Debug Event Select Register (DEVENT)....2819
64.3.5 Debug Data Acquisition Message Register (DDAM)....2821
64.4 Using Debug Resources for Stack Limit Checking....2822
64.5 External Debug Support....2824
64.5.1 External Debug Registers....2826
64.5.2 OnCE Introduction....2832
64.5.3 JTAG/OnCE Pins....2835

64.5.4 OnCE Internal Interface Signals....2835
64.5.5 OnCE Interface Signals....2836
64.5.6 OnCE Controller and serial interface....2839
64.5.7 Access to Debug Resources....2847
64.5.8 Methods of Entering Debug Mode....2849
64.5.9 CPU Status and Control Scan Chain Register (CPUSCR)....2850
64.5.10 Reserved Registers (Reserved)....2857
64.6 Watchpoint Support....2857
64.7 MPU Operation During Debug....2860
64.8 Cache Array Access During Debug....2861
64.9 Basic Steps for Enabling, Using, and Exiting External Debug Mode....2861

Chapter 65
Debug and Calibration Interface (DCI)

65.1 Introduction....2865
65.1.1 Features....2865
65.1.2 Overview....2865
65.1.3 Operating modes....2868
65.2 External signal description....2875
65.3 Register description....2875
65.3.1 DCI control register (DCI_CR)....2875
65.3.2 DCI EVTx pin multiplexing control register (DCI_PINCR)....2877

Chapter 66
JTAG Controller (JTAGC)

66.1 Introduction....2879
66.1.1 Block diagram....2879
66.1.2 Features....2880
66.1.3 Modes of operation....2880
66.2 External signal description....2881
66.2.1 TCK—Test clock input....2882

66.2.2 TDI—Test data input....2882
66.2.3 TDO—Test data output....2882
66.2.4 TMS—Test mode select....2882
66.2.5 JCOMP—JTAG compliancy....2882
66.3 Register description....2883
66.3.1 Instruction register....2883
66.3.2 Bypass register....2883
66.3.3 Device identification register....2883
66.3.4 JTAG_PASSWORD register....2884
66.3.5 Boundary scan register....2885
66.4 Functional description....2885
66.4.1 JTAGC reset configuration....2885
66.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port....2885
66.4.3 TAP controller state machine....2886
66.4.4 JTAGC block instructions....2888
66.4.5 Boundary scan....2891
66.5 Initialization/Application information....2892

Chapter 67
IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)

67.1 References....2893
67.2 Abbreviations....2893
67.3 Introduction....2894
67.3.1 Types of operation....2895
67.3.2 Deployment by class....2895
67.3.3 1149.7 TAP signals....2896
67.3.4 TAP.7 architecture....2897
67.3.5 Protocols....2898
67.4 Operating models....2900

67.5 CJTAG implementation summary....2900
67.5.1 T0 functions....2900
67.5.2 T1 functions....2901
67.5.3 T2 functions....2901
67.5.4 T3 functions....2901
67.5.5 T4 functions....2901
67.6 Ancillary services....2901
67.6.1 Overview....2901
67.6.2 Resets....2902
67.6.3 Start-up Options....2905
67.6.4 RSU operation....2906
67.6.5 TAPC State Machine....2908
67.7 EPU (Extended Protocol Unit) Operation....2909
67.7.1 EPU Operation....2909
67.7.2 EPU Registers....2912
67.7.3 EPU Commands....2924
67.7.4 EPU operating states....2932
67.7.5 System and EPU Paths....2933
67.8 APU (Advanced Protocol Unit) Operation....2933
67.8.1 Overview....2933
67.8.2 Operation....2936
67.8.3 Escape sequences....2938
67.8.4 Signal behaviors....2938
67.8.5 APU Functions....2939
67.8.6 Configuration Change Packets (CP)....2945
67.8.7 Scan Packet....2946
67.8.8 SP Format....2946
67.9 Functional Description....2953
67.9.1 Switching from Standard Protocol to Advanced Protocol....2953

Chapter 68
JTAG Data Communication (JDC)

68.1 Introduction....2955

68.2 Overview....2955

68.3 Memory mapped registers....2956

68.3.1 Module Configuration Register (JDC_MCR)....2957

68.3.2 Module Status Register (JDC_MSR)....2958

68.3.3 JTAG Output Data Register (JDC_JOUT_IPS)....2959

68.3.4 JTAG Input Data Register (JDC_JIN_IPS)....2960

68.4 Non-Memory mapped register definition....2960

68.4.1 JTAG output data register (JOUT)....2960

68.4.2 JTAG input data register (JIN)....2961

68.5 Functional description....2961

68.5.1 JTAG register access....2962

Chapter 69
Sequence Processing Unit (SPU)

69.1 Sequence Processing Unit (SPU) chip-specific module information....2963

69.2 Introduction....2963

69.2.1 SPU block diagram....2963

69.2.2 Overview....2964

69.2.3 Features....2965

69.3 SPU actions....2968

69.4 SPU enable/disable mode....2969

69.5 SPU module interface....2969

69.6 Register definition....2970

69.6.1 Level1 input Mux configurations registers....2974

69.6.2 Level2 input Mux configurations registers....2981

69.6.3 Sequence unit registers....2988

69.6.4 SLU status (SS) register....2995

69.6.5 SPU enable (SE) register....2997
69.6.6 Sample and hold mechanism....2998
69.6.7 SLU action unit registers....2999
69.6.8 Counters control n (CCTRLn) registers....3006
69.6.9 Counters compare n (CCMPn) registers....3007
69.6.10 Status registers....3008
69.7 Functional description....3010
69.7.1 Input Mux unit....3010
69.7.2 Processor exception vector encoding....3012
69.7.3 State logic unit (SLU)....3013
69.7.4 Sequence formation....3014
69.7.5 Performance counters/timers unit....3017
69.7.6 Action unit....3020
69.7.7 Optional trace group....3020
69.7.8 CKSRC and CKDATA format....3020

Chapter 70
JTAG Master (JTAGM)

70.1 Introduction....3023
70.1.1 Overview....3023
70.1.2 Feature description....3024
70.2 Functional description....3025
70.2.1 JTAGM Ready signal....3025
70.2.2 JTAGM to DCI serial interface....3026
70.2.3 Software interface....3026
70.3 Modes of operation....3026
70.4 Memory mapped registers....3027
70.4.1 Module Configuration Register (JTAGM_MCR)....3027
70.4.2 Status Register (JTAGM_SR)....3029
70.4.3 Data Out Register 0 (JTAGM_DOR0)....3030

70.4.4 Data Out Register 1 (JTAGM_DOR1)....3031
70.4.5 Data Out Register 2 (JTAGM_DOR2)....3031
70.4.6 Data Out Register 3 (JTAGM_DOR3)....3032
70.4.7 Data Input Register 0 (JTAGM_DIR0)....3032
70.4.8 Data Input Register 1 (JTAGM_DIR1)....3033

Chapter 71
Development Trigger Semaphore (DTS)

71.1 Introduction....3035
71.2 Overview....3035
71.3 DTS device connections....3037
71.3.1 DTS register access....3038
71.4 Memory mapped registers....3039
71.4.1 Output Enable Register (DTS_ENABLE)....3039
71.4.2 Startup Register (DTS_STARTUP)....3040
71.4.3 Semaphore Register (DTS_SEMAPHORE)....3041
71.4.4 Semaphore Extension (DTS_SEMAPHORE_B)....3042
71.5 Example application....3042

Chapter 72
Nexus Aurora Router (NAR)

72.1 Nexus Aurora Router (NAR) chip-specific information....3045
72.1.1 NAR DID value....3045
72.2 Introduction....3045
72.3 Overview....3047
72.4 Features....3047
72.5 Register definition....3048
72.5.1 NAR control register (NAR_CR)....3048
72.5.2 NAR status register (NAR_ST)....3050

72.5.3 Message filtering registers....3052
72.5.4 Trace memory bus configuration registers....3055
72.5.5 Suppress mode triggers registers (NAR_STCR)....3058
72.5.6 Client suppress status register (NAR_CSSR)....3059
72.5.7 Receive queue client disable register (NAR_CDR)....3060
72.5.8 Trace Memory fill level and partition configuration register (NAR_AHFPAR)....3062
72.6 Functional description....3063
72.6.1 Reset/Startup....3063
72.6.2 Operation modes....3064
72.6.3 Data reception and arbitration....3066
72.6.4 Suppress mode....3069
72.6.5 Stall detection....3070
72.6.6 Message translation and formatting....3071
72.6.7 Trace memory bus output port usage....3071
72.6.8 Internal message generation....3073
72.6.9 Timestamp function....3077
72.6.10 Overlay and internal trace memory full status generation....3078
72.6.11 Configuration/debug interface JTAG....3079
72.6.12 Trace memory bus controller....3082
72.6.13 Legacy client interface controller....3084
72.7 Application information....3086

Chapter 73
Nexus Module

73.1 Introduction....3089
73.1.1 General Description....3089
73.1.2 Terms and Definitions....3089
73.1.3 Feature List....3090
73.1.4 Functional Block Diagram....3092

73.2 Enabling Nexus 3 Operation....3093
73.2.1 Interaction with Low Power Modes....3094
73.3 TCODEs supported....3095
73.4 Nexus 3 Programmer's Model....3100
73.4.1 Client Select Control (CSC) register....3102
73.4.2 Port Configuration Register (PCR) - reference only....3102
73.4.3 Nexus Development Control Register 1 (DC1)....3103
73.4.4 Nexus Development Control Registers 2 & 3 (DC2, DC3)....3105
73.4.5 Nexus Development Control Register 4 (DC4)....3111
73.4.6 Development Status Register (DS)....3112
73.4.7 Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers....3113
73.4.8 Nexus Watchpoint Mask (WMSK) register....3122
73.4.9 Nexus Overrun Control Register (OVCR)....3124
73.4.10 Data Trace Control Register (DTC) register....3125
73.4.11 Data Trace Start Address Registers (DTSA1-4)....3128
73.4.12 Data Trace End Address (DTEA1-4) registers....3128
73.4.13 Read/Write Access Control/Status (RWCS) register....3129
73.4.14 Read/Write Access Data (RWD) register....3131
73.4.15 Read/Write Access Address (RWA)....3133
73.5 Nexus 3 Register Access via JTAG/OnCE....3133
73.6 Nexus 3 Register Access via Software....3134
73.7 Nexus Message Fields....3135
73.7.1 TCODE Field....3135
73.7.2 Source ID Field (SRC)....3135
73.7.3 Relative Address Field (U-ADDR)....3135
73.7.4 Full Address Field (F-ADDR)....3136
73.8 Nexus Message Queues....3137
73.8.1 Message Queue Overrun....3137
73.8.2 CPU Stall....3137

73.8.3 Message Suppression....3138
73.8.4 Nexus Message Priority....3138
73.8.5 Data Acquisition Message Priority Loss Response....3139
73.8.6 Ownership Trace Message Priority Loss Response....3140
73.8.7 Program Trace Message Priority Loss Response....3140
73.8.8 Program Trace Sync Message Priority Loss Response....3140
73.8.9 Data Trace Message Priority Loss Response....3140
73.9 Debug Status messages....3141
73.10 Error messages....3141
73.11 Ownership trace....3142
73.11.1 Overview....3142
73.11.2 Ownership Trace Messaging (OTM)....3142
73.12 Program trace....3144
73.12.1 Branch Trace Messaging types....3144
73.12.2 BTM Message For mats....3146
73.12.3 Program Trace Message Fields....3147
73.12.4 Resource Full Messages....3149
73.12.5 Program Correlation Messages....3150
73.12.6 Program Trace Overflow Error messages....3151
73.12.7 Program Trace Synchronization messages....3151
73.12.8 Program Trace Direct/Indirect Branch with Sync messages....3153
73.12.9 Enabling Program Trace....3154
73.12.10 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration)....3155
73.13 Data Trace....3156
73.13.1 Data Trace Messaging (DTM)....3156
73.13.2 DTM Message formats....3157
73.13.3 DTM Operation....3160
73.13.4 Data Trace Timing Diagrams (8 MDO / 2 MSEO configuration)....3161

73.14 Data Acquisition messaging....3162
73.14.1 Data Acquisition ID Tag field....3162
73.14.2 Data Acquisition Data field....3163
73.14.3 Data Acquisition Trace event....3163
73.15 Watchpoint Trace messaging....3164
73.15.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration)....3165
73.16 External Hardware Trigger Controls....3166
73.17 Nexus 3 Read/Write access to memory-mapped resources....3167
73.17.1 Single write access....3168
73.17.2 Block write access....3169
73.17.3 Single read access....3170
73.17.4 Block read access....3171
73.17.5 Error handling....3172
73.17.6 Read/Write access error message....3173
73.18 Nexus 3 pin interface....3173
73.18.1 Pins implemented....3173
73.18.2 Pin protocol....3176
73.19 Rules for output messages....3178
73.20 Auxiliary port arbitration....3179
73.21 Examples....3179
73.22 Electrical characteristics....3182
73.23 IEEE 1149.1 (JTAG) RD/WR sequences....3182
73.23.1 JTAG sequence for accessing internal Nexus registers....3183
73.23.2 JTAG sequence for read access of memory-mapped resources....3183
73.23.3 JTAG sequence for write access of memory-mapped resources....3183

Chapter 74
Nexus Crossbar Multi-Master Client (NXMC)

74.1 NXMC chip-specific module information....3185
74.1.1 Generating In Circuit Trace Messages (ICTM)....3185

74.2 Introduction....3186
74.2.1 Features....3186
74.3 External signal description....3187
74.4 Supported messages and TCODEs....3188
74.5 Register description....3191
74.5.1 Development control register 1 (DC1)....3192
74.5.2 Development control register 2 (DC2)....3193
74.5.3 Watchpoint trigger register (WT)....3194
74.5.4 Data trace control register (DTC)....3195
74.5.5 Data trace start address registers 1 and 2 (DTSA1 and DTSA2)....3196
74.5.6 Data trace end address registers 1 and 2 (DTEA1 and DTEA2)....3196
74.5.7 Breakpoint/watchpoint control register 1 (BWC1)....3197
74.5.8 Breakpoint/watchpoint control register 2 (BWC2)....3198
74.5.9 Breakpoint/watchpoint address registers 1 and 2 (BWA1 and BWA2)....3199
74.5.10 Unimplemented registers....3199
74.6 Programming considerations (RESET)....3200
74.6.1 IEEE 1149.1 (JTAG) Test Access Port....3200
74.6.2 Enabling NXMC operation....3200
74.6.3 Enabling NXMC TAP controller....3200
74.6.4 NXMC register access via JTAG....3200
74.7 Functional description....3201
74.7.1 Data trace....3202
74.7.2 Watchpoint support....3207
74.7.3 Peripheral interface....3209
74.7.4 Timestamping feature....3210

Chapter 75
Functional Safety

75.1 Introduction....3213
75.2 Safety overview....3213

75.3 Module categorization....3213
75.4 System implementation....3214
75.4.1 General concept....3214
75.4.2 Dual-core lockstep....3222
75.4.3 Common Cause Failure measures....3225
75.4.4 ECC....3225

Chapter 76
Cyclic Redundancy Check (CRC) Unit

76.1 Introduction....3229
76.2 Main features....3229
76.2.1 Standard features....3229
76.3 Block diagram....3230
76.4 External signal description....3230
76.4.1 Peripheral bus interface....3230
76.5 CRC memory map and registers....3231
76.5.1 Configuration Register (CRC_CFG)....3232
76.5.2 Input Register (CRC_INP)....3233
76.5.3 Current Status Register (CRC_CSTAT)....3234
76.5.4 Output Register (CRC_OUTP)....3234
76.6 Functional description....3235
76.7 Use cases....3237
76.7.1 Programming example....3237
76.7.2 Register programming....3238

Chapter 77
Memory Error Management Unit (MEMU)

77.1 Memory Error Management Unit (MEMU) chip-specific module information....3241
77.1.1 MEMU system connections....3241
77.1.2 MEMU error sources....3241
77.1.3 Reporting table implementation....3242

77.1.4 Concurrent overflow register (OFLW) implementation....3242
77.1.5 Uncorrectable error in flash memory can cause uncorrectable error in SRAM....3244
77.2 Introduction....3244
77.3 Features....3246
77.4 Block diagram....3247
77.5 Design overview....3247
77.6 External signal description....3251
77.7 Memory map and register definition....3252
77.7.1 Overview....3252
77.7.2 Control register (MEMU_CTRL)....3256
77.7.3 Error flag register (MEMU_ERR_FLAG)....3256
77.7.4 Debug register (MEMU_DEBUG)....3260
77.7.5 System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS*n*)....3262
77.7.6 System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR*n*)..3263
77.7.7 System RAM uncorrectable error reporting table status register (MEMU_SYS_RAM_UNCERR_STS). 3263
77.7.8 System RAM uncorrectable error reporting table address register (MEMU_SYS_RAM_UNCERR_ADDR)....3264
77.7.9 System RAM concurrent overflow register (MEMU_SYS_RAM_OFLW*n*)....3264
77.7.10 Peripheral RAM correctable error reporting table status register (MEMU_PERIPH_RAM_CERR_STS*n*)....3265
77.7.11 Peripheral RAM correctable error reporting table address register (MEMU_PERIPH_RAM_CERR_ADDR*n*)....3266
77.7.12 Peripheral RAM uncorrectable error reporting table status register (MEMU_PERIPH_RAM_UNCERR_STS)....3266
77.7.13 Peripheral RAM uncorrectable error reporting table address register (MEMU_PERIPH_RAM_UNCERR_ADDR)....3267
77.7.14 Peripheral RAM concurrent overflow register (MEMU_PERIPH_RAM_OFLW*n*)....3267
77.7.15 Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS*n*)....3268
77.7.16 Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR*n*)....3269
77.7.17 Flash memory uncorrectable error reporting table status register (MEMU_FLASH_UNCERR_STS)....3269

77.7.18 Flash memory uncorrectable error reporting table address register (MEMU_FLASH_UNCERR_ADDR)....3270
77.7.19 Flash memory concurrent overflow register (MEMU_FLASH_OFLW*n*)....3270
77.8 Functional description....3271
77.8.1 Initializing MEMU....3271
77.8.2 Reading the reporting table....3272
77.8.3 Handling overflows (Multiple error reporting)....3272

Chapter 78
Fault Collection and Control Unit (FCCU)

78.1 FCCU chip-specific module information....3275
78.1.1 FCCU fault recovery after NMI reaction....3275
78.1.2 FCCU chip-specific register reset values....3275
78.1.3 Available FCCU output signals....3276
78.1.4 Error signal flow diagram....3276
78.1.5 FCCU failure inputs from other modules....3278
78.1.6 Lockstep Status....3287
78.1.7 FOSU Count Value....3287
78.1.8 FCCU_CFG[FCCU_SET_AFTER_RESET]....3287
78.2 Introduction....3287
78.2.1 Acronyms and abbreviations....3289
78.3 Main features....3289
78.4 Block diagram....3290
78.5 Signal description....3291
78.5.1 Signals....3291
78.6 Functional description....3293
78.6.1 Definitions....3293
78.6.2 FSM description....3293
78.6.3 Self-checking capabilities....3296
78.6.4 Fault priority scheme and nesting....3297

78.6.5 Fault recovery....3298
78.6.6 NMI/WKPU interface....3300
78.6.7 STCU interface....3301
78.6.8 EOUT interface....3302
78.6.9 Error signal flow....3307
78.7 Register descriptions....3308
78.7.1 Control (FCCU_CTRL)....3310
78.7.2 Control Key (FCCU_CTRLK)....3313
78.7.3 Configuration (FCCU_CFG)....3314
78.7.4 Noncritical Fault Configuration (FCCU_NCF_CFG*n*)....3317
78.7.5 Noncritical Fault State Configuration (FCCU_NCFS_CFG*n*)....3318
78.7.6 Noncritical Fault Status (FCCU_NCF_S*n*)....3319
78.7.7 Noncritical Fault Key (FCCU_NCFK)....3321
78.7.8 Noncritical Fault Enable (FCCU_NCF_E*n*)....3321
78.7.9 Noncritical Fault Timeout Enable (FCCU_NCF_TOE*n*)....3323
78.7.10 Noncritical Fault Timeout (FCCU_NCF_TO)....3324
78.7.11 Configuration-State Timeout (FCCU_CFG_TO)....3324
78.7.12 IO Control (FCCU_EINOUT)....3325
78.7.13 Status (FCCU_STAT)....3327
78.7.14 NA Freeze Status (FCCU_N2AF_STATUS)....3329
78.7.15 AF Freeze Status (FCCU_A2FF_STATUS)....3330
78.7.16 NF Freeze Status (FCCU_N2FF_STATUS)....3331
78.7.17 FA Freeze Status (FCCU_F2A_STATUS)....3332
78.7.18 SC Freeze Status (FCCU_SCFS)....3333
78.7.19 Noncritical Fault Fake (FCCU_NCFF)....3335
78.7.20 IRQ Status (FCCU_IRQ_STAT)....3335
78.7.21 IRQ Enable (FCCU_IRQ_EN)....3337
78.7.22 XTMR (FCCU_XTMR)....3338
78.7.23 Mode Controller Status (FCCU_MCS)....3339

78.7.24 Transient Lock (FCCU_TRANS_LOCK)....3341
78.7.25 Permanent Lock (FCCU_PERMNT_LOCK)....3341
78.7.26 Delta T (FCCU_DELTA_T)....3342
78.7.27 IRQ Alarm Enable (FCCU_IRQ_ALARM_EN*n*)....3343
78.7.28 NMI Enable (FCCU_NMI_EN*n*)....3344
78.7.29 Fault-Output (EOUT) Signaling Enable (FCCU_EOUT_SIG_EN*n*)....3345
78.8 FCCU Output Supervision Unit....3346
78.9 Use cases and limitations....3348
78.9.1 Misconfigurations....3348
78.9.2 Recommendations to configure FCCU....3348

Chapter 79
Self-Test Control Unit (STCU2)

79.1 STCU2 chip-specific module information....3351
79.1.1 STCU2 register reset values....3351
79.1.2 Chip-specific STCU2 registers....3352
79.1.3 MBIST Partitions....3354
79.1.4 MBIST Test Cycles....3355
79.1.5 Running Offline MBIST....3356
79.1.6 Off-line LBIST....3357
79.1.7 On-line LBIST....3358
79.1.8 LBIST Partitions....3358
79.1.9 LBIST Configuration....3361
79.1.10 AUTOLOCK_VALUE for register write access via STCU2_SKC....3362
79.1.11 Wait time for writing to the online registers....3362
79.1.12 On-Line reset generation (only MBIST)....3362
79.1.13 On-Line reset generation (LBIST enabled)....3362
79.2 Introduction....3363
79.3 Main features....3363
79.4 Block diagram....3365

79.5 IPS bus interface....3366
79.6 Functional description....3368
79.6.1 FSM description....3368
79.6.2 Reset management....3368
79.6.3 Built-in self-test scheduling....3368
79.6.4 ABORT management....3369
79.6.5 PLL interface....3370
79.6.6 Interrupt interface....3370
79.6.7 FCCU interface....3371
79.6.8 Watchdogs....3371
79.6.9 SSCM interface....3372
79.7 Register description....3373
79.7.1 STCU2 Run Register (STCU2_RUN)....3378
79.7.2 STCU2 Run Software Register (STCU2_RUNSW)....3379
79.7.3 STCU2 SK Code Register (STCU2_SKC)....3381
79.7.4 STCU2 Configuration Register (STCU2_CFG)....3382
79.7.5 STCU2 PLL Configuration Register (STCU2_PLL_CFG)....3385
79.7.6 STCU2 Watchdog Register Granularity (STCU2_WDG)....3386
79.7.7 STCU2 Interrupt Flag Register (STCU2_INT_FLG)....3388
79.7.8 STCU2 Error Register (STCU2_ERR_STAT)....3389
79.7.9 STCU2 Error FM Register (STCU2_ERR_FM)....3392
79.7.10 STCU2 Off-Line LBIST Status Register (STCU2_LBS)....3393
79.7.11 STCU2 Off-Line LBIST End Flag Register (STCU2_LBE)....3394
79.7.12 STCU2 On-Line LBIST Status Register (STCU2_LBSSW)....3395
79.7.13 STCU2 On-Line LBIST End Flag Register (STCU2_LBESW)....3397
79.7.14 STCU2 On-Line LBIST Reset Management (STCU2_LBRMSW)....3398
79.7.15 STCU2 LBIST Unrecoverable FM Register (STCU2_LBUFM)....3399
79.7.16 STCU2 Off-Line MBIST Status Low Register (STCU2_MBSL)....3400
79.7.17 STCU2 Off-Line MBIST Status Medium Register (STCU2_MBSM)....3405

79.7.18 STCU2 Off-Line MBIST Status High Register (STCU2_MBSH).....3408
79.7.19 STCU2 Off-Line MBIST End Flag Low Register (STCU2_MBEL).....3409
79.7.20 STCU2 Off-Line MBIST End Flag Medium Register (STCU2_MBEM).....3412
79.7.21 STCU2 Off-Line MBIST End Flag High Register (STCU2_MBEH).....3414
79.7.22 STCU2 On-Line MBIST Status Low Register (STCU2_MBSLSW).....3415
79.7.23 STCU2 On-Line MBIST Status Medium Register (STCU2_MBSMSW).....3419
79.7.24 STCU2 On-Line MBIST Status High Register (STCU2_MBSHSW).....3421
79.7.25 STCU2 On-Line MBIST End Flag Low Register (STCU2_MBELSW).....3421
79.7.26 STCU2 On-Line MBIST End Flag Medium Register (STCU2_MBEMSW).....3425
79.7.27 STCU2 On-Line MBIST End Flag High Register (STCU2_MBEHSW).....3427
79.7.28 STCU2 MBIST Unrecoverable FM Low Register (STCU2_MBUFML).....3428
79.7.29 STCU2 MBIST Unrecoverable FM Medium Register (STCU2_MBUFMM).....3431
79.7.30 STCU2 MBIST Unrecoverable FM High Register (STCU2_MBUFMH).....3434
79.7.31 STCU2 LBIST Control Register (STCU2_LB_CTRL*n*).....3434
79.7.32 STCU2 LBIST PC Stop Register (STCU2_LB_PCS*n*).....3437
79.7.33 STCU2 LBIST PRPG Low Register (STCU2_LB_PRPGL*n*).....3438
79.7.34 STCU2 LBIST PRPG High Register (STCU2_LB_PRPGH*n*).....3439
79.7.35 STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL*n*).....3439
79.7.36 STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH*n*).....3440
79.7.37 STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL*n*).....3441
79.7.38 STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH*n*).....3441
79.7.39 STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW*n*).....3442
79.7.40 STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW*n*).....3443
79.7.41 STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW*n*).....3444
79.7.42 STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW*n*).....3444
79.7.43 STCU2 MBIST Control Register (STCU2_MB_CTRL*n*).....3445
79.8 Use cases and limitations.....3446
79.8.1 Offline self-test sequence.....3446
79.8.2 Online self-test sequence.....3448

79.8.3 Bypass USER Mode....3450
79.8.4 Design implementation information....3451

Chapter 80
Register Protection (REG_PROT)

80.1 Register protection (REG_PROT) chip-specific module information....3453
80.1.1 SIUL2 protected registers....3453
80.1.2 CMU protected registers....3455
80.1.3 PMC_DIG protected registers....3455
80.1.4 PLL_DIG protected registers....3455
80.1.5 OSC_DIG protected registers....3456
80.1.6 MEMU protected registers....3456
80.1.7 MC_ME protected registers....3456
80.1.8 MC_CGM protected registers....3457
80.1.9 MC_RGM protected registers....3458
80.1.10 SSCM protected registers....3458
80.1.11 JTAGM protected registers....3458
80.2 Overview....3459
80.3 Features....3459
80.4 Modes of operation....3460
80.5 External signal description....3460
80.6 Memory map and register definition....3460
80.6.1 Memory map....3462
80.6.2 Register descriptions....3462
80.7 Memory map and registers....3463
80.7.1 Soft Lock Bit Register *n* (REG_PROT_SLBRn)....3463
80.7.2 Global Configuration Register (REG_PROT_GCR)....3465
80.8 Functional description....3466
80.8.1 General....3466

80.8.2 Change lock settings....3466
80.8.3 Access errors....3470

Chapter 81
Security

81.1 Overview....3473
81.1.1 Security Concepts....3473
81.1.2 Security Feature....3474
81.2 Secure System Configuration....3474
81.2.1 Overview....3474
81.2.2 Security-related configuration....3474
81.2.3 Device Configuration Format (DCF) records....3475
81.2.4 Security-related data protection....3475
81.3 External access protection....3477
81.3.1 Overview....3477
81.3.2 Debug access protection....3477
81.3.3 JTAGM and Debug Over CAN....3480
81.3.4 Serial bootloader protection....3481
81.3.5 BAR execution protection....3481
81.4 Flash Memory Protection....3482
81.4.1 Overview....3482
81.4.2 Flash memory program and erase protection....3482
81.4.3 Block Read Protection....3484
81.4.4 Test Feature Protection....3485
81.4.5 Flash overlay mechanism restrictions....3487
81.5 End of Life Protection....3487
81.5.1 Overview....3488
81.5.2 BAF code in Failure Analysis....3488
81.5.3 Protection of Flash Test Mode activation....3488

Chapter 82
Password and Device Security Module (PASS)

82.1 Password and Device Security Module (PASS) chip-specific module information....3489

82.2 Introduction....3490

82.2.1 Overview....3490

82.2.2 Features....3491

82.2.3 Modes of operation....3491

82.3 External signal description....3491

82.4 Memory map and register definition....3492

82.4.1 Life Cycle Status Register (PASS_LCSTAT)....3494

82.4.2 Challenge Selector Register (PASS_CHSEL)....3495

82.4.3 Challenge Status Register (PASS_CSTAT)....3495

82.4.4 Challenge Input Register (PASS_CIN*n*)....3496

82.4.5 Password Group *n* - Lock 0 Status Register (PASS_LOCK0_PG*n*)....3497

82.4.6 Password Group *n* - Lock 1 Status Register (PASS_LOCK1_PG*n*)....3498

82.4.7 Password Group *n* - Lock 2 Status Register (PASS_LOCK2_PG*n*)....3499

82.4.8 Password Group *n* - Lock 3 Status Register (PASS_LOCK3_PG*n*)....3501

82.5 DCF clients....3504

82.5.1 Censorship....3504

82.5.2 Production Disable....3505

82.6 Functional description....3506

82.6.1 Censorship....3506

82.6.2 Debug Interface Access....3507

82.6.3 Flash Memory Read Protection....3507

82.6.4 Production Disable....3508

82.7 Initialization/application information....3509

82.7.1 Reset....3509

82.7.2 Setting lock bits in a password group....3509

Chapter 83
Tamper Detection Module (TDM)

83.1 Tamper Detection Module (TDM) chip-specific module information....3511

83.1.1 TDRn/Lockm OTPENm register bit map....3511

83.1.2 OTPEN0 value....3511

83.2 Overview....3512

83.3 Features....3513

83.4 External signal description....3513

83.5 DCF clients....3514

83.5.1 Diary Base Address (DBA) DCF client....3514

83.5.2 Tamper Region Override (TO) DCF client....3515

83.5.3 Software Tamper Region Override Disable (STO_DIS) DCF client....3516

83.5.4 One Time Programmable Enable (OTPEN0) DCF client....3517

83.5.5 One Time Programmable Enable (OTPEN1) DCF client....3518

83.5.6 One Time Programmable Enable (OTPEN2) DCF client....3518

83.5.7 One Time Programmable Enable (OTPEN3) DCF client....3519

83.5.8 Tamper Region Assignment DCF client (TDRn_LOCK0)....3520

83.5.9 Tamper Region Assignment DCF client (TDRn_LOCK1)....3521

83.5.10 Tamper Region Assignment DCF client (TDRn_LOCK2)....3522

83.5.11 Tamper Region Assignment DCF client (TDRn_LOCK3)....3523

83.6 Memory Map/Register Definition....3524

83.6.1 TDR Status Register (TDM_TDRSR)....3525

83.6.2 Last Flash Programmed Address Register (TDM_LFPAR)....3527

83.6.3 Diary Base Address (TDM_DBA)....3528

83.6.4 Software Tamper Override Key Region (TDM_STO_KEY*n*)....3528

83.7 Functional description....3529

83.7.1 Flash erase counter and tamper detection....3529

83.7.2 Assigning blocks to Tamper Detect Regions (TDR)....3530

83.7.3 Diary....3531

83.7.4 Specifying the diary base address....3532
83.7.5 TDR lock status....3533
83.7.6 TDR override....3533
83.7.7 One time programmable (OTP)....3534

Chapter 1
Preface

1.1 Overview

The primary objective of this document is to define the functionality of the MPC5746R microcontroller for use by software and hardware developers.

The information in this book is subject to change without notice, as described in the disclaimers on the back cover. As with any technical documentation, it is the reader's responsibility to be sure he or she is using the most recent version of the documentation.

To locate any published errata or updates for this document, visit the Freescale Web site at http://www.freescale.com.

1.2 Audience

This manual is intended for system software and hardware developers and applications programmers who want to develop products with the MPC5746R device. It is assumed that the reader understands operating systems, microprocessor system design, basic principles of software and hardware, and basic details of the Power Architecture.

1.3 Content

Please note the following about this version of this document, and use it with a complete understanding of its limitations.

- This is a preliminary draft.
- Its content is subject to change without notice.

1.4 Document organization

This document includes two major sets of chapters:

- Chapters in the first set, SoC Specific Chapters, describe the device as a whole or device-specific information.
- Chapters in the second set describe the functionality of the individual modules on the device. These chapters are organized into the following groups:
  - System modules
  - Clocking
  - Memories and Memory Interfaces
  - Analog modules
  - Timers
  - Communication Interface Modules
  - Reset and Boot Modules
  - Power Management Modules
  - Calibration and Debug Modules
  - Safety Modules
  - Security modules

Within a module chapter, module information that is specific to the chip appears in the Configuration section at the beginning of the chapter.

1.5 Register description conventions

The register information for this chip is presented in:

- Memory maps containing:
  - An offset from the module's base address
  - The name and acronym/abbreviation of each register
  - The page number on which each register is described
- Register figures
- Field-description tables
- Associated text

The register figures show the field structure using the conventions in the following figure.

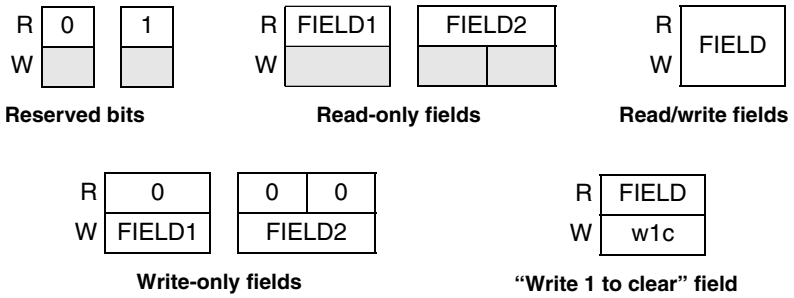


Figure 1-1. Register figure conventions

The numbering of register bits and fields on this chip is as follows:

- Register bit numbers, shown at the top of each figure, use the standard Power Architecture bit ordering (0, 1, 2, ...) where bit 0 is the most significant bit (MSB).
- Multi-bit fields within a register use conventional bit ordering (..., 2, 1, 0) where bit 0 is the least significant bit (LSB).

1.6 Special terms

The following terms have special meanings:

Term	Meaning
asserted	Refers to the state of a signal as follows: • An active-high signal is asserted when high (1). • An active-low signal is asserted when low (0).
deasserted	Refers to the state of a signal as follows: • An active-high signal is deasserted when low (0). • An active-low signal is deasserted when high (1). In some cases, deasserted signals are described as *negated*.
reserved	Refers to a memory space, register, field, or programming setting. Writes to a reserved location can result in unpredictable functionality or behavior. • Do not modify the default value of a reserved programming setting, such as the reset value of a reserved register field. • Consider undefined locations in memory to be reserved.
w1c	Write 1 to clear: Refers to a register bitfield that must be written as 1 to be "cleared."

1.7 Acronyms and abbreviations

Term	Meaning
ATX	Analog Test Module
AUTOSAR	Automotive Open System Architecture

Table continues on the next page...

Term	Meaning
BAF	Boot Assist Flash
CAN	Controller Area Network
CGM	Clock Generation Module
CMU	Clock Monitoring Unit
CRC	Cyclic Redundancy Check
DMA	Direct Memory Access
DMAMUX	Direct Memory Access Controller Multiplexer
DSPI	Deserial Serial Peripheral interface
DTS	Development Tool Semaphore
FCCU	Fault Control and Collection Unit
FEC	Fast Ethernet controller
GPIO	General-purpose I/O
IEEE	Institute for Electrical and Electronics Engineers
INTC	Interrupt Controller
JDC	JTAG Data Communication
JEDEC	Joint Electron Device Engineering Council
JTAG	Joint Test Action Group
Zipwire (LFAST/SIPI)	LVDS Fast Asynchronous Serial Transmission
LINFlex	Local Interconnect Network controller
ME	Mode Entry Module (MC_ME)
MEMU	Memory Error Management Unit
Mux	Multiplex
OSC	Oscillator
PASS	Password and Device Security Module
PCM	Platform Configuration Module
PCU	Power Control Unit
PFLASH	Flash controller
PIT	Periodic Interrupt Timer
PMC	Power Management Controller
PRAM	RAM controller
RCOSC	RC Oscillator
RGM	Reset Generation Module
RTL	Register transfer language
Rx	Receive
SD	ADC Sigma Delta Analog to Digital Converter
SAR	ADC Successive Approximation Register Analog to Digital Converter
SEMA4	Semaphore
SENT	Single Edge Nibble Transmission
SIUL	System Integration Unit Lite
SMPU	System Memory Protection Unit

Table continues on the next page...

Term	Meaning
SoC	System on Chip
SSCM	System Status and Configuration Module
STAC	Shared Time and Angle Count Bus
STCU	Self-Test Control Unit
STM	System Timer module
SWT	Software Watchdog Timer
TSENS	Temperature Sensor
Tx	Transmit
UART	Universal asynchronous/synchronous receiver transmitter
WKPU	Wakeup Unit
XBAR	Crossbar

1.8 References

In addition to this reference manual, the following documents provide additional information on the operation of the MPC5746R:

- IEEE-ISTO 5001-2003 Standard for a Global Embedded Processor Interface (Nexus)
- IEEE 1149.1-2001 standard - IEEE Standard Test Access Port and Boundary-Scan Architecture

Chapter 2
Introduction

2.1 Introduction

The MPC5746R microcontroller is a member of a family of powertrain devices based on the eTPU timer. The device builds on the legacy of the family, while introducing new features coupled with higher throughput to provide substantial reduction of cost per feature and significant performance improvement.

2.2 Target applications

The MPC5746R is targeted for mid-range automotive powertrain controller applications.

Many of the target applications are considered to be functionally safe and are designed to achieve ISO26262 ASIL-D compliance.

2.3 Block diagrams

The figures below are the top-level block diagrams. They show the general interconnection of functional modules through the crossbar switch and provide an indication of the modules that connect to external pins.

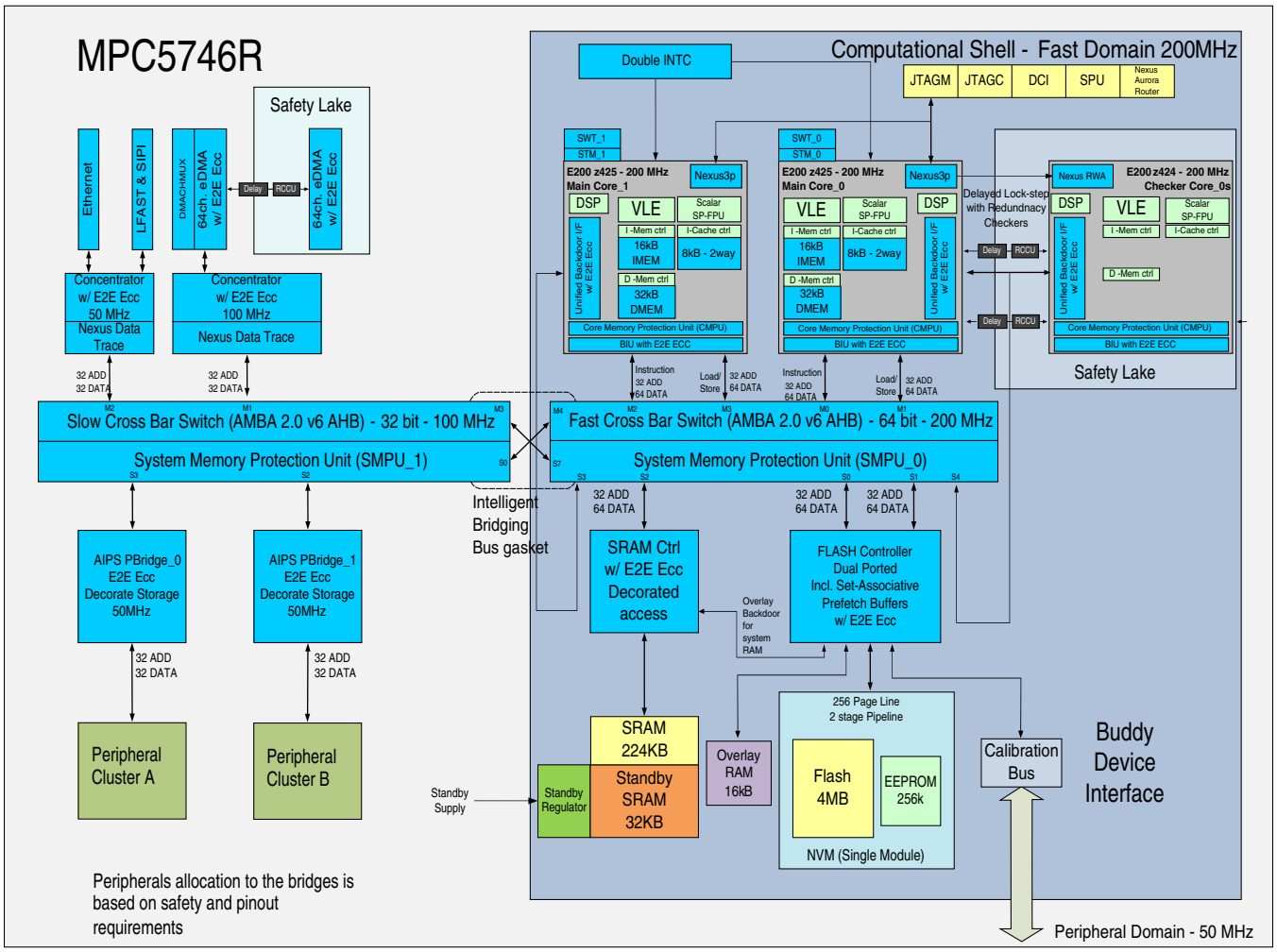


Figure 2-1. Core block diagram

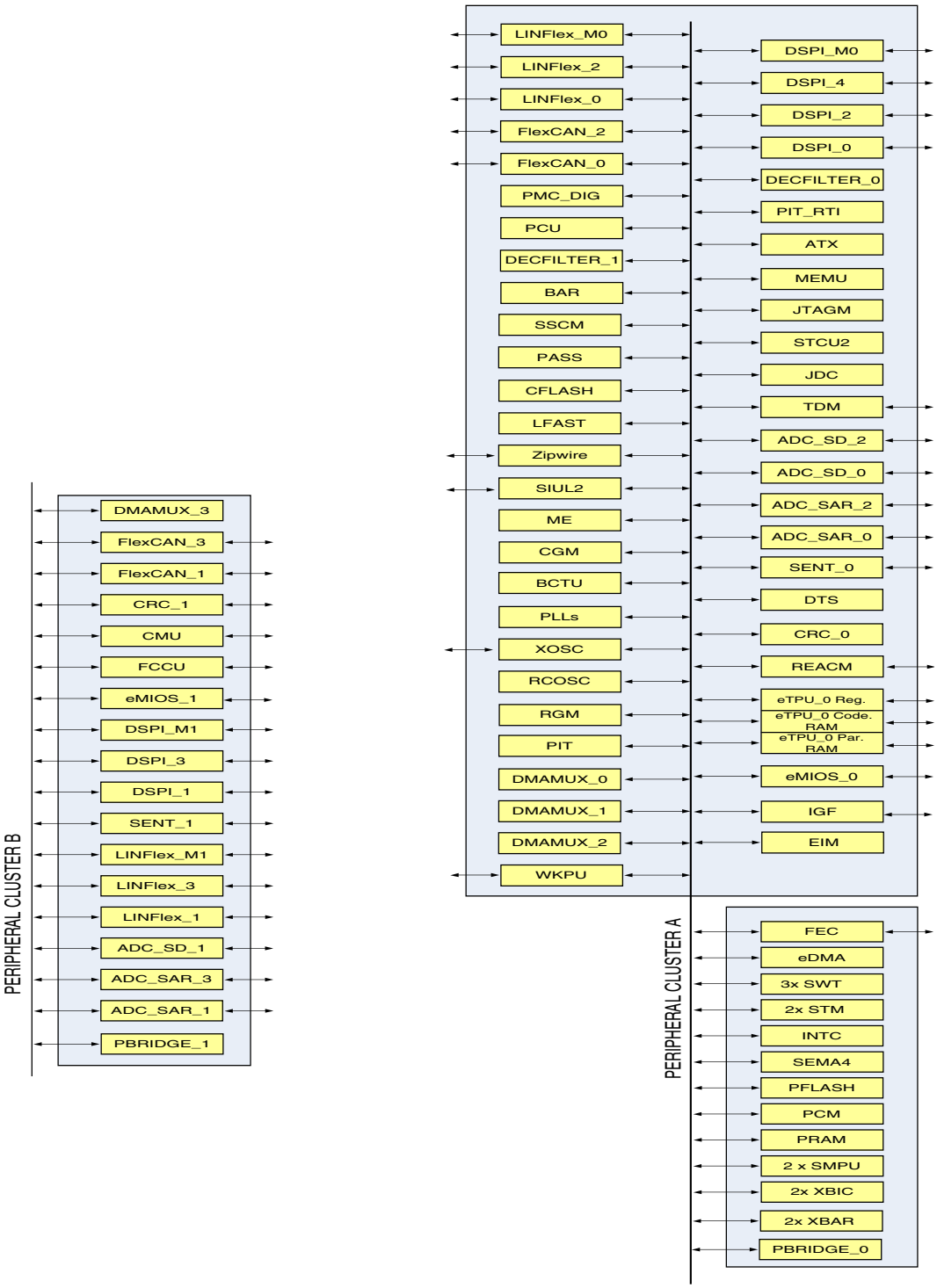


Figure 2-2. Peripherals allocation

2.4 Overall architecture

The architecture is split into two distinct regions: the computational shell and the I/O subsystem.

The computational shell consists of two CPUs, one additional core in lock-step for safety, fast local memories inside the CPUs, and fast system memory. All components of the computational shell are connected through a fast crossbar switch.

The I/O subsystem consists of the DMA and all other peripherals connected through a slow crossbar switch.

The computational shell is optimized for performance and the I/O subsystem is optimized for low power.

2.5 Core features

The device includes two user programmable CPU cores and one safety core. The two e200z425 CPUs operate at 200 MHz and include enhancements to implement Digital Signal Processing (DSP) algorithms. The third core is an e200z424 which runs as a safety checker in delayed lockstep mode with one of the e200z425 CPUs.

All CPUs are compatible with the Power Architecture™ VLE instruction set, which supports code size reduction. The Power Architecture has enhancements that improve the architecture's fit in embedded applications.

2.6 Memory hierarchy

The device computational shell has three levels of memory hierarchy.

- At the top level is a combination of 8 KB instruction cache, 16 KB Local Instruction RAM, and 32 KB Local Data RAM. These RAM arrays are embedded into the CPU's pipeline to provide the fastest possible access time.
- The second level of memory is the system RAM and flash.They are connected via a 200 MHz, 64-bit wide crossbar.
- The third level of memory system is the I/O subsystem. All the I/O peripherals are connected together via a 100 MHz, 32-bit wide crossbar switch. Peripherals are connected at either 100 MHz or 50 MHz depending on their need. The reduced speed for the peripherals helps to reduce power.

2.7 Features summary

On-chip modules available within the MPC5746R include the following features:

- Main CPUs:
  - Three dual issue, 32-bit CPU core complexes (e200z4), one of which is a dedicated lockstep core
  - Power Architecture embedded specification compliance
  - Instruction set enhancement with variable length encoding (VLE, encoding of mixed 16-bit and 32-bit instructions) for code size footprint reduction
  - Single-precision floating point operations
  - 16 KB local instruction RAM and 32 KB local data RAM
  - 8 KB I-Cache
- 4272 KB on-chip flash including 16 KB UTEST
  - Supports reads during program and erase operations, and multiple blocks allowing EEPROM emulation
- 256 KB on-chip general-purpose SRAM
  - 32 KB Standby RAM powered by a separate supply so the contents of this portion can be preserved when the main MCU is powered down
- 64-channel direct memory access controller (eDMA)
- Interrupt controller (INTC)
- Dual Phase-locked loops with stable clock domain for peripherals and FM modulation domain for computational shell
- Dual Crossbar switch architecture for concurrent access to peripherals, flash, or RAM from multiple bus masters
- System Integration Unit Lite 2 (SIUL2)
- Boot Assist Flash module (BAF) supports serial bootload via LINFlex and FlexCAN
- Seven deserial serial peripheral interface (DSPI) modules
- Six UART communication interface modules with LIN support (LINFlex)

    - LINFlexD_0 is a Master/Slave
    - LINFlexD_1, LINFlexD_2,LINFLEXD_3, LINFlexD_M0 and LINFlexD_M1 are Masters only
- Four FlexCAN modules
    - Two modules with 96 Message Buffers
    - Two modules with 64 Message Buffers
- Enhanced analog-to-digital converter system with
    - Four separate 12 bit SAR analog converters
    - Three separate 16 bit Sigma-Delta analog converters
- Two 32 channel enhanced time processing units (eTPU)
- 32 channel Input Glitch Filter (IGF) connected to eTPU and eMIOS inputs
- Two 16 channel timed IOs (eMIOS)
- 10 channel Reaction Moduile (REACM)
- Nexus development interface (NDI) per IEEE-ISTO 5001-2003 standard, with some support for 2010 standard
- Device and board test support per Joint Test Action Group (JTAG) (IEEE 1149.1)
- Self Test capability

See section Feature list for feature differences in the MPC5745R, MPC5743R, and MPC5742R.

2.8 Feature list

The following table lists a summary of major features of the device.

Table 2-1. Features list

Feature	MPC5746R	MPC5745R	MPC5743R
Main core	2	2	1[1]
Locked step core	1	1	1
Flash	4256KB + 16KB	3232KB + 16KB	2208KB + 16KB
EEPROM	2x64KB + 8x16KB	2x64KB + 8x16KB	2x64KB + 8x16KB

Table continues on the next page...

Table 2-1. Features list (continued)

Feature	MPC5746R	MPC5745R	MPC5743R
System RAM	256KB (including standby RAM)	192KB (including standby RAM)	128KB (including standby RAM)
I-CACHE	8KB/core	8KB/core	8KB/core
Local IRAM	16KB/core	16KB/core	16KB/core
Local DRAM	32KB/core	32KB/core	32KB/core
Stby RAM	32KB	32KB	32KB
Overlay RAM	16KB	16KB	16KB
Core freq	200MHz	150MHz[2]	200MHz
SAR	4	4	4
SD ADC	3	3	3
Decimation Filters	2	2	2
eTPU	32ch x 2	32ch x 2	32ch x 2
Reaction module	10ch	10ch	10ch
DMA controller	64ch	64ch	64ch
FlexCAN	4	4	4
LINFlex	4+2MSB support	4+2MSB support	3+2MSB support[3]
DSPI	5+2MSB support	5+2MSB support[4]	4+2MSB support[5]
Ethernet	10/100MBPS	10/100MBPS	10/100MBPS
SENT	6ch	6ch	6ch
Package	MAPBGA252 MAPBGA292 Emulation Device	LQFP176 or MAPBGA252	LQFP176 or LQFP144

1. Core0 + Locked step core will be supported. Core 1 will be used for BAF execution at start-up only.
2. 256 BGA package supports 200MHz operation.
3. LINFlex 3 is not supported.
4. For 176 LQFP DSPI 4 is not supported.
5. DSPI 4 is not supported. For 144 LQFP, DSPI M1 is also not supported.

2.9 Packaging

The device will be available in three production packages:

- 144 LQFP
- 176 LQFP
- 252 MAPBGA

The device uses a MAPBGA292 as the Emulation Device.

2.10 Software debug and calibration

The Production Device (PD) includes many features to facilitate software development and calibration of constant data. These features include:

- Nexus code execution trace on all CPUs and eTPUs
- Nexus data trace on all bus masters (DMA, Zipwire, CPUs, and others) and eTPUs
- Performance monitors for each CPU
- Sequence Processing Unit for complex trace triggering
- 16 KB overlay RAM

The microcontroller has a comprehensive feature set to assist debug and calibration. Specialized variants, referred to as Emulation and Debug Devices (EDs), include additional calibration and debug features. An ED consists of the normal production die plus an additional die, called a Buddy die or Buddy Device (BD), bonded together in the same package. A Production Device (PD) includes a production die only and no buddy die.

The Emulation and Debug Device (ED) will:

- Provide all of the features and modules that are included in the Production Device (PD)
- Be available in the MAPBGA292 package
- Have additional overlay RAM (up to 1 Mbytes)
- Have a high-speed Nexus trace output port consisting of four lanes of Aurora (LVDS) running at up to 1.25 Gbits/second

Chapter 3
Embedded memories

3.1 Overview

The embedded memory architecture for MPC5746R devices is a significant departure from previous-generation devices. Features include:

- On-chip SRAM, including system SRAM, local instruction and data memory for each processor core, and overlay SRAM
- On-chip system error correction code (ECC) flash memory
- End-to-end ECC (e2eECC) error detection and correction [1]
- Embedded memories in the peripherals [2] :
  - FlexCAN
  - eDMA
  - FEC
  - eTPU2
  - NAR

3.2 SRAM

3.2.1 System SRAM

MPC5746R includes 256 KB of general-purpose on-chip ECC SRAM, including 32 KB standby RAM. The SRAM can be configured for either zero- or one-wait state read operation latency using the PFCR1 register in the SRAM controller.

- See the RAM controller (PRAM) chapter for details on configuring the SRAM controller.

1. Not all modules support e2eECC.
2. Refer to the respective module chapter for details.

3.2.2 Standby SRAM Regulator

A total of 32 KB of the system SRAM on the device can remain powered when the rest of the device is powered down. This is achieved with the dedicated VDDSTBY pin in the packages. When the standby RAM is powered by VDDSTBY, the RAM is not operational, but the contents are retained.

The Standby RAM regulator has the following key features:

- Automatic, safe switching between operational (VDD_LV) and standby (VDDSTBY) supplies
- Internal regulation of the high voltage VDDSTBY supply to the low voltage standby RAM supply
- Brownout detection (minimum RAM data retention voltage violation) and reporting via a S/W clearable register bit

The Standby RAM regulator block diagram is given in Figure 3-1.

There are two primary modes of operation with the standby RAM feature:

- Standby RAM feature is not required in the application - in this case, the VDDSTBY pin is connected to ground, and the standby RAM on the device is supplied by VDD_LV and will behave as normal RAM for all conditions.
- Standby RAM feature is required in the application - in this case, the VDDSTBY pin is connected to a voltage within the range given in the device data sheet. The VDDSTBY pin is assumed to be connected to a constant supply, and present before and after the power up/down of the device LV and HV supplies. When the VDD_LV supply to the RAM is above the LV operational voltage threshold, the standby RAM is fully operational and behaves exactly the same as the non-standby system RAM. When the VDD_LV supply falls below the LV Detect threshold, the RAM low voltage supply is switched to the standby supply. See the device data sheet for min/max VDDSTBY values, LV supply thresholds, and the operational and brownout detection voltages.

The brownout detection feature of the standby RAM regulator informs the user if any voltage dip caused a RAM retention failure in standby mode. The brownout detector monitors the supply to the RAM array, and sets a latch in the PMC digital interface, when the voltage falls below the minimum RAM retention voltage given in the device data sheet. The latch is set on power up of the device. S/W then clears the latch by following the instructions in PMC_ADC_CS[STND_BY_RAM_BR_DETCT] description in the Power Management Controller digital interface (PMC_dig) chapter. If the minimum

RAM retention voltage is violated, the latch will be set again. The latch is set on every power-up of VDD_LV unless VDDSTBY is present and within the specified ranges in the data sheet.

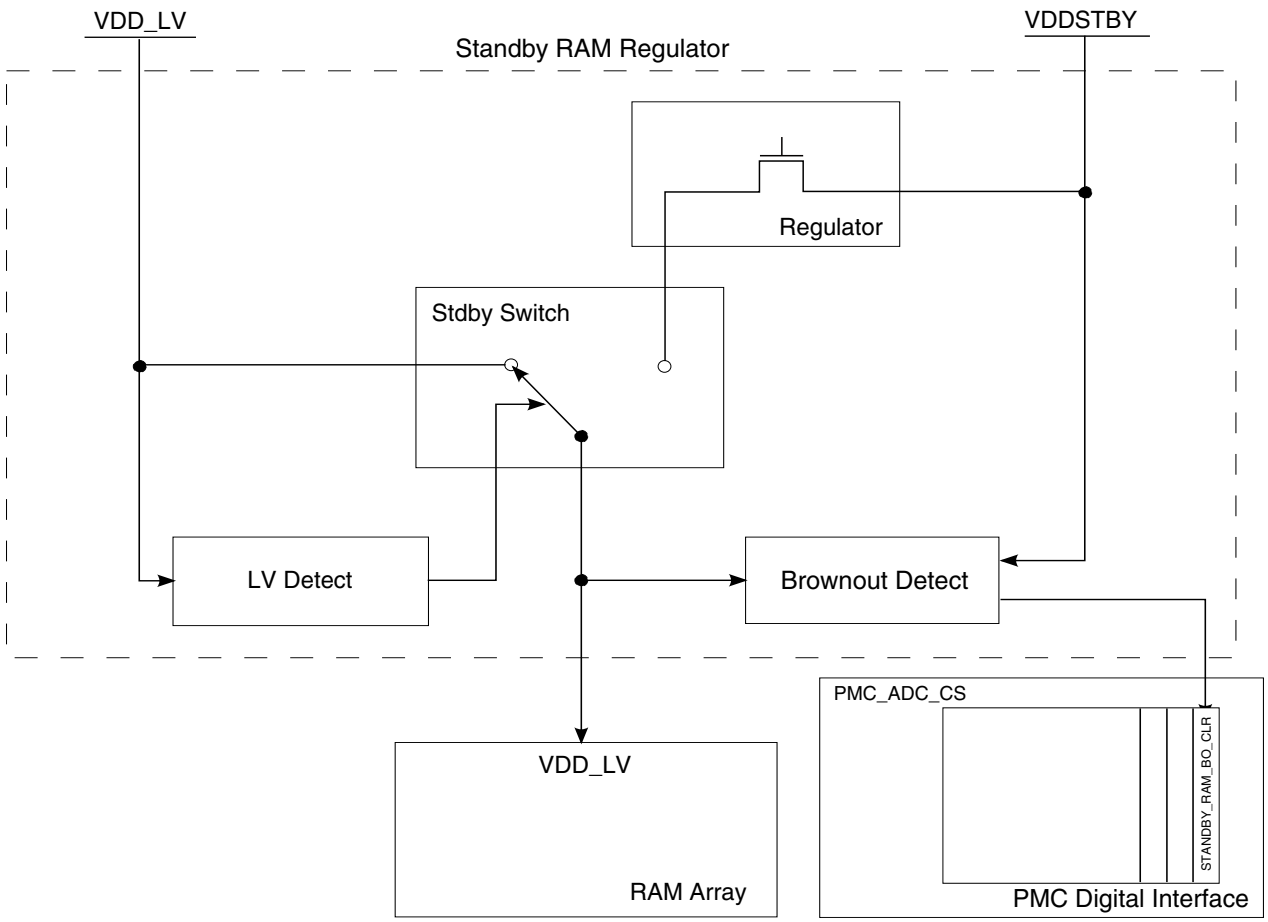


Figure 3-1. Standby RAM regulator block diagram

3.2.3 Overlay SRAM

Overlay SRAM is included in the MPC5746R family of devices as part of a comprehensive set of calibration and debug features. Overlay SRAM can be mapped over specific regions of on-chip flash memory so that any access to an overlaid flash address is routed to the overlay SRAM instead. This enables the calibration of constant data without requiring additional external RAMs and calibration memory interfaces.The microcontroller has two 8 KB banks of 8-bit ECC-protected RAM, for a total of 16 KB.

NOTE

All read/write accesses to Overlay RAM must be 32bit aligned.

NOTE

Overlay SRAM is normally used for system development purposes only. Code can be executed from internal overlay RAM and extended overlay RAM where available. Execution from overlay RAM remapped over Flash can be enabled or disabled by use of PFCRCR[IRMEM]. Code can be executed directly from Extended Overlay RAM (address range beginning 0x0C000000) on the ED device or internal Overlay RAM (address range beginning 0x0D000000) on the PD regardless of the PFCRCR[IRMEM] setting.

Overlay memory can also be used to hold the device debug trace stream without requiring complex debug tool hardware and access to a high-speed trace port. Timing for calibration accesses to a particular overlay SRAM is the same as access to the underlying flash memory if the overlay SRAM is not being used as a destination for trace streaming at the same time.

The calibration remap function supports three different types of overlay SRAM:

- An internal overlay RAM is included on all standard production devices.
- An extended overlay RAM is an additional memory that is available only on special emulation and debug devices.
- A portion of the system RAM can be used as the overlay RAM.

The overlay function is implemented using registers in the flash memory controller.

1. Using PFlash Calibration Region Descriptors (PFCRDn) define one or more (up to 32) overlay regions.
2. Enable individual regions using the appropriate bits in the PFLASH Remap Descriptor Enable Register (PFDCRDE).
3. Enable overlay using the PFCRCR[GRMEN] field.

See PFlash calibration remap support for details on memory overlay (remapping).

3.2.4 Processor core local SRAM

MPC5746R devices include local instruction and data SRAM (I-MEM and D-MEM) for each processor core to provide enhanced performance. Each main processor core has 16 KB of instruction SRAM and 32 KB of data SRAM.

The core SRAM supports zero wait-state, single-cycle latency on processor local accesses.

Each area of core SRAM is accessible by other processor cores and bus mastering peripheral devices.

3.3 Flash memory

Flash memory on the MPC5746R microcontroller consists of a flash memory controller and a flash memory array module. The flash controller provides flash configuration and control functions and manages the interface between the flash memory array and the device crossbar switch. Figure 3-2 shows the memory architecture.

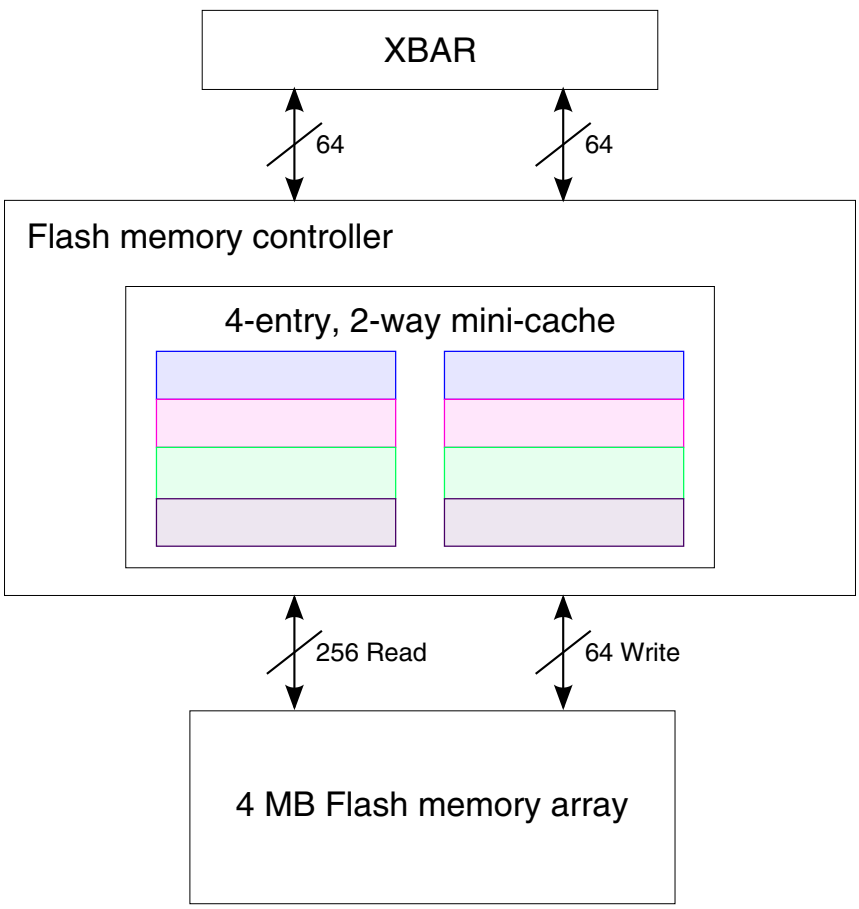


Figure 3-2. Flash memory block diagram

The remaining sections give the functional details of the flash controller and flash array, followed by the memory maps.

3.4 Flash memory controller

The MPC5746R flash memory controller acts as an interface between the system bus and the flash memory array and serves as the interface to the on-chip overlay RAM and off-chip Buddy Device. [3]

The flash memory controller contains a four-entry, two-way set-associative mini-cache that delivers flash read data with a zero-wait state response on lines that reside in the cache. Each entry contains one flash page, a 256-bit (32-byte) memory value. Read requests that miss the cache generate the needed flash memory array access.

The flash memory controller contains configuration registers that manage flash functionality such as read buffering in the mini-cache, access control, calibration RAM overlay remapping, and read wait state management of the flash memory.

See the flash memory controller chapter for details.

3.5 Flash memory array

MPC5746R devices include a single 4256 KB flash memory array, referred to as “main space,” plus an independent 16 KB block of one-time-programmable (OTP) flash memory included to support systems that require non-volatile memory for security features or system initialization information. The independent 16 KB block is referred to as “UTEST space.”

Reads of the embedded flash memory return a 256-bit page of data that may be buffered in the mini-cache in the flash memory controller. Programming of the flash may be done by double word (64-bits), page (256-bits), or quad-page (1024-bits). Erase of the flash memory is done on a block by block basis.

3.5.1 Features

- Flash segmentation
  - 16 KB, 32 KB, 64 KB, and 256 KB blocks provided for code or data
  - Eight 16 KB blocks and two 64 KB blocks provided for EEPROM emulation
  - Support for reading-while-writing when the accesses are to different partitions

3. The Buddy Device is an optional companion chip with additional memory and debug features. A Buddy Device is combined with a Production Device to form an Emulation and Debug Device (ED) for use during system development. It is not used in production systems.

- Flash memory protection: write protection and one-time-programmable (OTP) function available for each block
- Test and initialization data stored in a non-volatile 16 KB OTP UTEST block
- ECC with double bit detection, single bit correction
- Erase suspend, program suspend, and erase-suspended program all supported

3.5.2 Flash organization

The embedded flash memory consists of six address spaces in three groupings:

- 16 KB low address space, 32 KB low address space, and 64 KB low address space
- 16 KB mid address space and 64 KB mid address space
- 256 KB address space

Address space group (low-, mid-, or 256 KB address space) determines which registers/fields are used to select blocks for modify operations or lock them from modify operations. Each address space is independent.

The flash module is further divided into six partitions that determine locations for valid read-while-write (RWW) operations. While the embedded flash memory is performing a “write” (program or erase) to a given partition, it can simultaneously perform a read from any other partition.

- For program operations, only the address specified by an interlock write determines the partition being written (block locking and block select registers do not determine the RWW partitions being written).
- For erase operations, only blocks that are selected and unlocked determine the RWW partitions being written.

Figure 3-3 shows the flash block segmentation and read-while-write partitioning for MPC5746R devices.

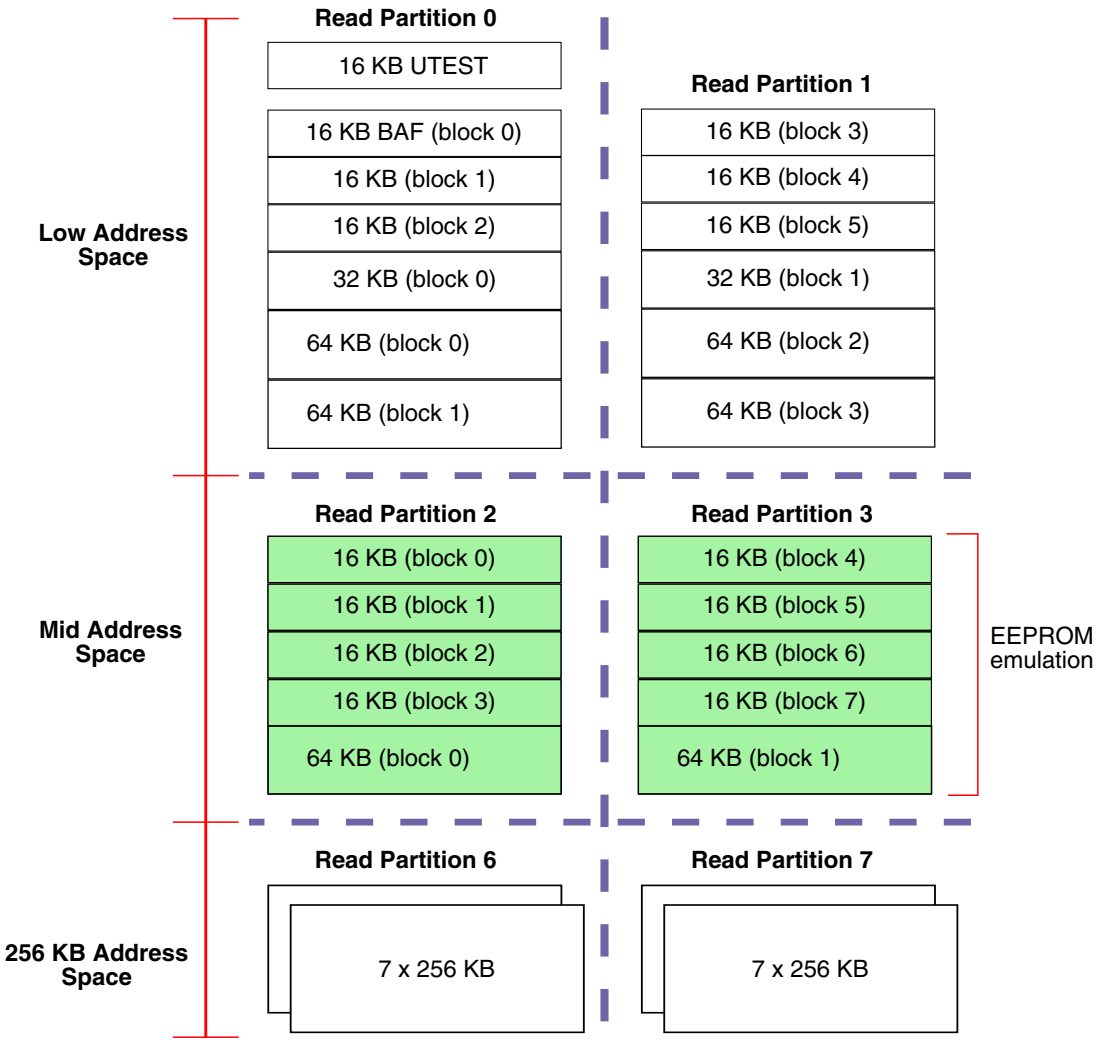


Figure 3-3. Flash memory segmentation and read-while-write partitioning

3.5.3 UTEST memory space

Devices in the MPC5746R family contain a 16 KB area of One-Time Programmable (OTP) flash memory for storing test information and device configuration data.

3.6 End-to-end Error Correction Code (e2eECC)

To support market requirements related to improved functional and transient fault detection capabilities, this family of automotive microcontrollers includes end-to-end ECC (e2eECC) support. This e2eECC is structurally different to traditional “ECC at

memory" functionality because it provides robust error detection capabilities from one endpoint of an information transfer to another endpoint, with temporary information storage in one or more intermediate components.

Memory protected by ECC/EDC traditionally generates and checks additional error parity information local to the memory unit to detect and/or correct errors that have occurred on data stored in the memory. On the other hand, e2eECC generates error protection codes at the source of data generation. It sends the encoded data and error protection codes to intermediate storage when a memory write is initiated by a bus master and performs a data integrity check using the previously stored error protection codes at a data memory when a read of stored information is requested by a bus master. The intermediate storage may transform the generated error protection codes into another format for storage and then regenerate the codes for provision when a request is made to read the stored information or it may simply store the original protection codes unaltered, depending on the particular unit.

Additionally, the error protection codes are generated on the basis of more than just the data associated with a storage location to protect additional information associated with an access. In particular, address information corresponding to the access location of the stored information is combined with the stored data at the data source to generate error protection codes that cover certain types of addressing errors in the system interconnect or in the memory unit. The error protection codes may be checked at the memory unit on a store to ensure that no address information or data has been corrupted while the request has transitioned through the device from the bus master source. The error protection codes may also be checked to ensure that address decoding within the storage memory was performed properly (although not all address decoding errors can be detected this way) or it may simply store the received data and protection codes at the address it receives.

On a read request from a bus master, the memory unit retrieves the data information and error protection codes corresponding to the received address from the storage location or locations and supplies the data along with the error protection codes to the requesting device. The requesting device uses a locally stored address value corresponding to the read request to check the returning data and error protection codes to ensure that no errors have occurred in either addressing the memory or in the retrieved data. This ensures that the address sent for the request was not corrupted, the addressed location was actually accessed (to the extent it is possible to ensure) and the stored data was error free, to the extent the ECC coding scheme is able to detect. As a result, the fault coverage provided by the end-to-end check is considerably more robust than the previous implementation of locally generated and checked/corrected error protection at each memory unit.

A comparison of the traditional and e2eECC approaches is shown in Table 3-1.

Table 3-1. ECC comparison of data write-then-read sequence

Traditional ECC	End-to-End ECC (e2eECC)
Bus master initiates data write	Bus master initiates data write and generates ECC checkbits based on 29-bit address and 64-bit data fields for the computational shell, or based on 30-bit address and 32-bit data field for the peripheral shell
Data write transfer routed from bus master to appropriate bus slave	Data write transfer (including checkbits) routed from bus master to appropriate bus slave
For a bus memory slave, generate the ECC checkbits based on the data value and store data + checkbits into the memory	For a bus memory slave, store data and checkbits into the memory
Bus master initiates data read of previously written memory location	Bus master initiates data read of previously written memory location
Data read transfer routed from bus master to appropriate bus slave	Data read transfer routed from bus master to appropriate bus slave
For a bus memory slave, the memory array is accessed, the controller performs the ECC checkbit decode and syndrome generation, performs any needed single-bit correction and drives the read data onto the system bus interconnect	For a bus memory slave, the memory array is accessed, and the controller passes the read data and associated checkbits onto the system bus interconnection
The bus master captures the read data and continues	The bus master captures the read data and associated checkbits, performs the ECC checkbit decode and syndrome generation, performs any needed single-bit correction and continues

The scope of differences in the operations “covered” by the ECC checks is readily apparent with the e2eECC concept providing improved fault detection capabilities in two important aspects:

- The entire data transaction; from the initiating bus master, through the entire platform crossbar steering mechanism, to the destination slave target is covered during write accesses. Likewise, a read is checked from the initiating bus master, through the crossbar steering mechanism, through the actual memory read and transmission of the data plus checkbits back to the bus master, where the integrity and correctness of the entire transaction is checked.
- The selected ECC provides protection of both the address field as well as the data field, again for improved fault coverage.

Additionally, this particular structure also provides a significant implementation improvement. The traditional ECC at the memory approach places the checkbit decode and error syndrome generation with the error/no_error state, affecting whether the system bus transfer must be stalled (to correct a single bit error or report a noncorrectable event) or is allowed to complete (for error-free transfers) in a critical timing arc which often sets the upper limit of the operating frequency of the microcontroller. With the e2eECC scheme, the checkbit decode and error syndrome logic is located in the requesting bus master where there are generally more degrees of implementation freedom and the error/no_error state determination is removed from the system bus cycle termination logic; producing an improved timing arc and generally higher operating speeds.

Errors that occur within the system interconnect are typically manifested as an incorrect address, incorrect write data, or incorrect read data to be presented to the slave or back to the master. Errors occurring within the storage typically manifest themselves eventually in corrupted read data or checkbits being returned to a requester. While not all possible errors can be detected this way, this approach provides a substantial improvement versus traditional methods. Additional on-line diagnostics, or additional hardware, or both should be able to catch a majority of the errors not covered directly by the e2eECC scheme.

Chapter 4
Signal Multiplexing and Signal Description

4.1 Generic pins/balls

The I/O Signal Description Table contains information on pins/balls of this device. See the device I/O Signal Description and Input Multiplexing Tables (Excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Table tab.

Multi-function pins are programmable via their respective Multiplexed Signal Configuration Register (MSCR) Source Signal Select (SSS) values. The SSS value selects which source signal is connected to the associated destination. For additional information, refer to System Integration Unit Lite2 (SIUL2) chapter.

The I/O Signal Table provides the port pin, SIUL MSCR number, MSCR SSS, function, module, signal description, direction, pad type, and package pin/ball numbers for each I/O pin on the device.

4.1.1 System pin functions

The following table describes the functions of the system pins.

Table 4-1. System pin functions

Function	Description	Direction	Active State	144QFP	176QFP	252MAP BGA	292MAP BGA (ED)
XTAL	External Oscillator Output Analog output of the oscillator amplifier circuit needs to be grounded if oscillator is used in bypass mode.	O		7	7	G1	G1
EXTAL	External Oscillator Input Analog input of the oscillator amplifier circuit when oscillator is not in bypass mode. Analog input for the clock generator when oscillator is in bypass mode.	I		6	6	F1	F1

Table continues on the next page...

Table 4-1. System pin functions (continued)

Function	Description	Direction	Active State	144QFP	176QFP	252MAP BGA	292MAP BGA (ED)
RESET	Functional Reset Ouput External functional reset with Schmitt trigger characteristics and noise filter. RESET is active low.	I/O	low	8	8	F3	F4
PORST	Power On Reset Input Power on reset with Schmitt trigger characteristics and noise filter. PORST is active low.	I/O	low	9	9	G3	G4
TESTMODE	Test Mode Enable Input Pin for testing purpose only. An internal pull-down is implemented on the TESTMODE pin to prevent the device from entering TESTMODE. It is recommended to connect the TESTMODE pin to VSS_HV_IO on the board. The value of the TESTMODE pin is latched at the negation of reset and has no affect afterward. The device will not exit reset with the TESTMODE pin asserted during power-up.	I	high	23	24	M2	M2

Chapter 5
Memory Map

5.1 Introduction

The memory map is provided as an Excel file attached to this document with the file name *Memory_Map_Table.xslx*. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it.

The Excel file contains six memory maps.

- Overview memory map
- Peripheral memory map
- NVM and overlay memory map
- Buddy device memory map
- RAM memory map
- UTEST flash memory map

All addresses on the chip, including those that are reserved, are identified in the tables. The addresses represent the physical addresses assigned to each region or module name.

The **Overview memory map** summarizes the complete 32-bit address map. A memory access, either read or write, to any region marked as 'Reserved' will result in a Transfer Error. This error can generate an interrupt request in the CPU.

The **Peripheral memmory map** lists the memory map of the peripheral region. This region is sub-divided into 16 KB regions or 'slots'. A peripheral occupies one or more slots. A memory access, either read or write, to any slot marked as 'Reserved' will result in a Transfer Error. This error can generate an interrupt request in the CPU.

Within a 16 KB peripheral slot, it is normal for the peripheral registers to only occupy part of that 16 KB slot, normally from the lowest address upwards. The individual chapters for each block define all the registers within that contiguous group of registers.

Any memory access outside the range of the defined register map will result in a Transfer Error. This error can generate an interrupt request in the CPU.

The other tables define the embedded memory maps and the buddy memory map. A memory access, either read or write, to any region marked as 'Reserved' in these tables, will result in a Transfer Error. This error can generate an interrupt request in the CPU.

Chapter 6
Device Configuration Format (DCF) Records

6.1 Overview

Device Configuration Format (DCF) records are used by the System Status and Configuration Module (SSCM) to configure certain registers in the device during system boot while the reset signal is asserted.

There are two kinds of DCF records:

- Factory-written DCF records: written to UTEST flash by the factory.
- User-supplied DCF records: written to UTEST flash by the user.

The following figure is a conceptual representation of the functionality of DCF records which is described in this chapter.

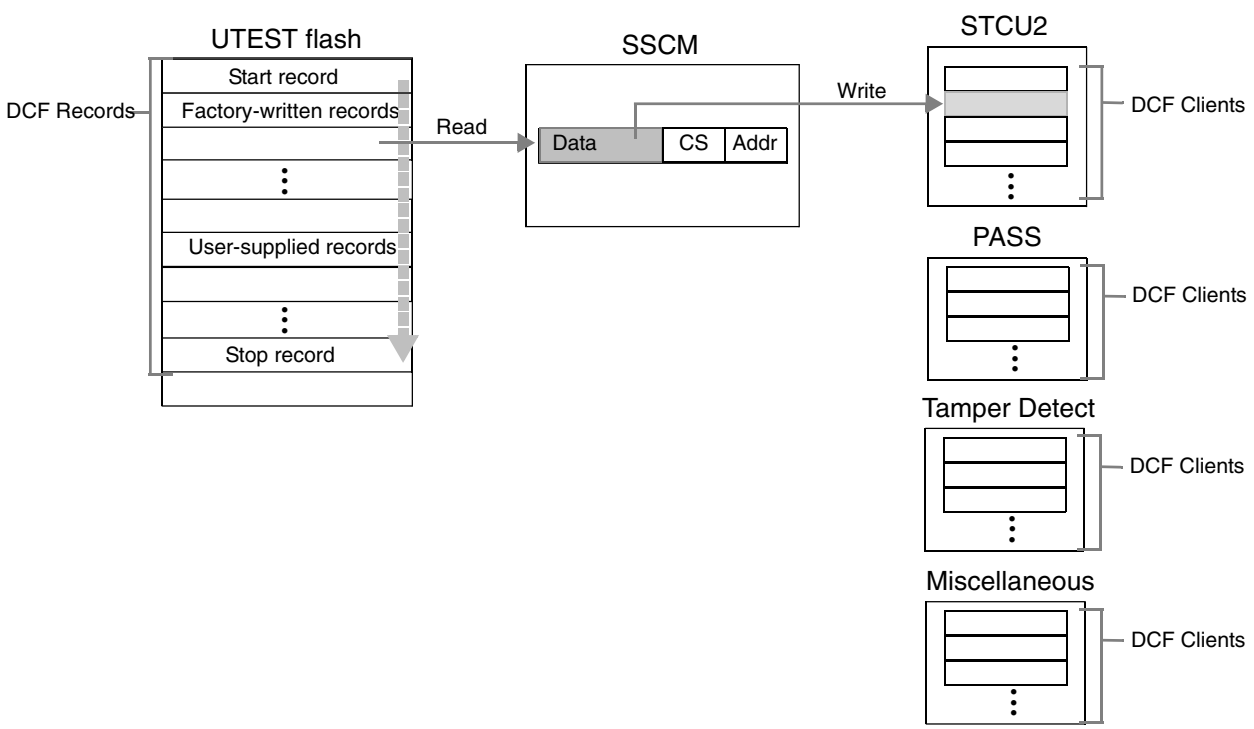


Figure 6-1. Conceptual representation of DCF records functionality

6.2 DCF records in boot sequence

For the device to be used in a user application, the system boot process must properly program the following to the respective flash memories before releasing reset.

- User application code
- Reset vectors for all CPUs
- DCF records

The following is a high-level summary of the boot sequence leading to the use of DCF records:

1. When power is applied to a properly programmed device, the Power Management Controller (PMC) takes control of the device.
2. After the system power supplies have reached predefined levels, the Power Management Controller signals the Reset Generation Module (RGM) to begin the boot sequence.
3. During the boot sequence, the RGM enables the System Status and Control Module (SSCM) to read the Device Configuration Format records.
4. SSCM writes the configuration information to the specified registers.
5. After registers are initialized, the SSCM passes the control of the boot sequence back to the RGM, which directs the STCU2 to start the memory and logic built in self tests (MBIST and LBIST).

6.3 DCF records function

User-supplied DCF records define the following:

- Setup of the initial memory map
- Tests the Self-Test Control Unit (STCU2) runs during the boot sequence
- Memory blocks assigned as OTP
- Flash memory block association with specific password groups
- Flash memory block association with specific tamper detect regions

6.4 DCF records location

Factory-written DCF records start at the first address in the UTEST flash memory area. See the UTEST flash memory map in the Memory map chapter.

User-supplied UTEST DCF records may be added at the next location in the UTEST memory map immediately following the factory-written UTEST DCF records.

6.5 DCF record structure

A DCF record is a double-word (64-bit) entry consisting of the following:

- Control word: Information to locate the corresponding DCF client internal to the device (pointer to the location of a register internal to the device)
- Data word: Data to be written to that client

DCF Records select the target DCF client by a 30-bit field in the DCF Record consisting of a 15-bit ‘Chip Select’ field and a 15-bit Address field. Each module that includes DCF clients is assigned a Chip Select during chip definition. The address field is only relevant to the address decoding within that module and may not necessarily relate to the address of a register visible to software.

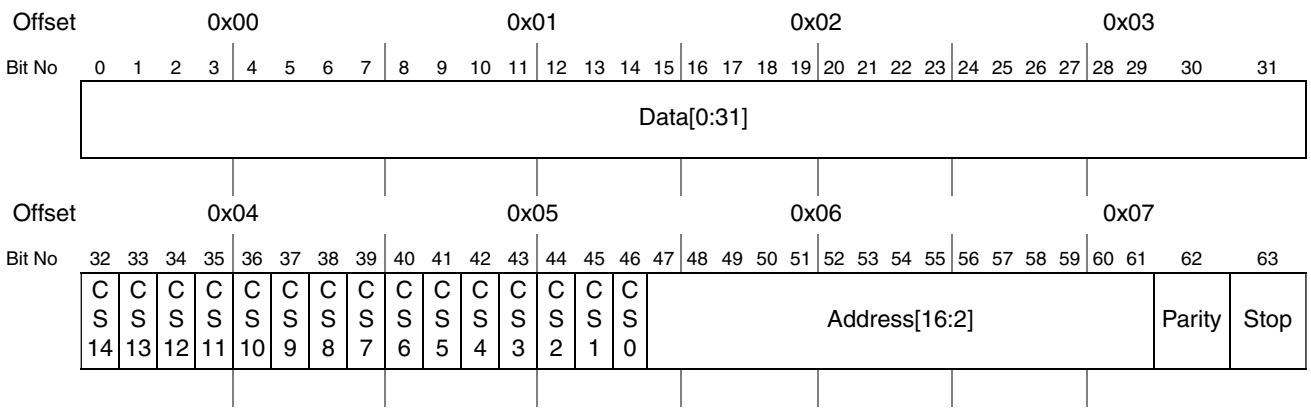


Figure 6-2. DCF Record Structure

Table 6-1. DCF Record Field Description

Field	Name	Description
0-31	Data[0:31]	32 bits of data that is to be written to the DCF Client
32-46	CS*n*	Chip Select *n* One Chip select is asserted (‘1’) per DCF record to select the target module for the DCF client. All other Chip Selects should be negated (‘0’).
47-61	Address[16:2]	Address of the DCF client within the selected module Note: Address decoding for DCF clients may not match the standard software address map decoding. Details of DCF Client addresses are defined in each module chapter.
62	Parity	Parity Bit for the DCF Record. Even parity.

Table continues on the next page...

Table 6-1. DCF Record Field Description (continued)

Field	Name	Description
63	Stop	Stop bit. Indicates the end of the list of DCF Records. Stop = '0' - NOT the end of the list Stop = '1' - End of the list Note: The Erased state of flash is 0xFFFF_FFFF_FFFF_FFFF. Therefore the list ends with the first unprogrammed double word. This location can be programmed with a new record to extend the list.

6.6 DCF records sequence

DCF records must appear as a contiguous series of entries programmed from the beginning of the UTEST flash memory area in the following order.

1. **Start DCF record**: The first DCF record must be a start record. This record must be found at the beginning of a DCF area in flash memory to indicate to the device that the following records must be processed.
2. **DCF records containing configuration data**: DCF records containing configuration data must immediately follow the start record with no blank records in between.
3. **Stop DCF record**: The end of the configuration records is indicated by the stop record. A record with the stop bit set is a stop record. All other bits are ignored. No DCF records following the stop record are processed.

SSCM recognizes the following record as a start record.

Table 6-2. DCF start record

0x00 (0:31)	0x04 (32:63)
0x05AA_55AF	0x0000_0000

The start bit is set by the factory at the beginning of the UTEST flash memory area.

SSCM recognizes the following record as a stop record.

Table 6-3. DCF stop record

0:31	32:62	63
Ignored	Ignored	1

The stop bit is never set by the factory, since this would not allow the user to add DCF records. The flash memory location following the last factory-written UTEST DCF record is unprogrammed, with the content of 0xFFFF_FFFF. This is interpreted as the stop record, since the stop bit is set.

User-supplied DCF records may be added in a contiguous manner immediately following the factory-written DCF records. There must never be an unprogrammed record in the series of DCF records as it will be interpreted as a stop record.

The following table shows the series of DCF records when *n* data records are stored in the UTEST flash.

Table 6-4. Series of DCF records in UTEST flash memory

Record type	ADDR offset	DATA			
Start record	0x00	0x05AA_55AF			
	0x04	0x0000_0000			STOP=0
Data records	0x08	WDATA[31:0]			
	0x0C	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
	0x10	WDATA[31:0]			
	0x14	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
	...	...			
Stop record	8*n*-1 + 0x0	Reserved			
	8*n*-1 + 0x4	Reserved			1
	8*n* + 0x0				
	8*n* + 0x4				

It is possible for more than one DCF Record to write to the same DCF client. In this case the later record usually overrides a DCF client value set by a previous record. However, not all DCF clients allow overwrites; this depends on the DCF client implementation.

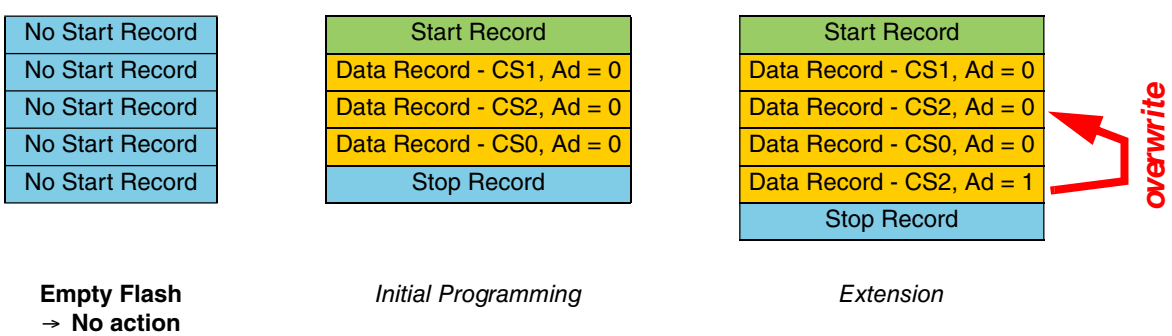


Figure 6-3. Appending DCF records

6.7 DCF clients

DCF clients are 32-bit wide hardware registers inside a module that receive and store the data from a DCF record. This stored data is used to initialize registers and to configure features.

DCF clients have the following characteristics:

- Have a default value before any DCF Records are written.
- Need not implement all 32 bits.

The table below shows the DCF clients available in this device. The meaning of the table columns are as follows:

- **DCFCS[14:0]**
  A 15-bit field with each bit in the field assigned to a particular module. These bits act as chip select signals for the internal modules. Only one bit in this field is be set in a DCF record.
- **DCF Address [16:2]**
  A 15-bit field that specifies an internal address for a DCF client. These addresses are used only by the SSCM when interpreting DCF records and writing the data to specified modules.
- **DCF Client Description**
  Name of the DCF clients
- **Reset Value of DCF Client**
  The reset value of DCF client before being written by the SSCM with the associated DCF record.
- **Typical Value of DCF Client**
  A typical value that you might program into the various DCF clients. This field is for information only.
- **IPS Read Direct Access**
  Whether or not a DCF client can be read by user software after the release of reset.
- **DCF Client Special Strategy**
  Though all DCF records have the same format, they can be used in different ways. As the device is being designed, registers must be designated as Device Configuration Format clients if they need to be written with a DCF record. There are other features that must be designated for DCF clients. For instance, DCF clients can have several different write strategies—Parity, Write Once, Triple Voted + Parity + Check with second write, Triple voted + Check with second write, or no strategy at all. The SSCM examines the DCF client to determine the correct special

strategy to use and then appropriately write a DCF record to that client. The following list defines the special strategies.

- None: No special DCF strategy is used.
- Parity: Parity Bit for the DCF Record. Even parity is expected of all DCFs which require parity evaluation. To confirm which UTEST DCFs require parity specification, refer to the DCF Client Special Strategy column in the table below.
- Write Once: A register using the Write Once strategy can only be written once. The DCF client ignores subsequent writes.
- Triple voting: These clients have three copies of the register. The SSCM will write to all three registers in a single write cycle. The outputs of the three registers are majority-voted to determine the correct data value. Triple voting allows for a ‘bit-flip’ error to occur without changing the DCF client output data.
- Triple Voting with second write: DCF clients that use the Triple Voted with 2nd write strategy have three copies of the register. The SSCM will write to all three registers in a single write cycle. The outputs of the three registers are majority-voted to determine the correct data value. During the second execution of Phase 3 of the reset sequence, the SSCM attempts to write the DCF client again. At this time, the DCF client checks to see that the register contains the same data that is being written again.
- Write 0 only: A bit in a DCF client can only be written from a logic 1 to a logic 0. An attempt to write a bit with this attribute to a logic 1 is ignored.
- Write 1 only: A bit in a DCF client can only be written from a logic 0 to a logic 1. An attempt to write a bit with this attribute to a logic 0 is ignored.
- Spread address (SPRD): This parameter is only available on safe_clients and dcf_safe_mclients. This parameter decides if the three instances share one address, or if they are addressable separately on three different addresses (i.e. spread out). If SPRD_ADR is instantiated as 1, the three dcf_clients are accessible at CFG_ADR + 0x04, CFG_ADR+0x08 and CFG_ADR+0x10. The lower 5 bits of CFG_ADR must be set to 0. In addition to spreading the addresses to three separate locations, SPRD_ADR also activates data transformation. Data transformation is used only in conjunction with triple-voting in the safe clients. It means that the data in the three sub-instances is not the same, but is stored in a transformed format. If the normal payload data is X, then the three sub-instances will store the data as a0=X, a1=~X and a2=shift_rotate_right(X). (So for e.g. X = 1011, a2 = 1101.)

- **DCF Order Dependency**

  Whether or not there are special conditions on the location of a particular DCF record in flash memory.

- **UTEST Flash Memory Space**

Estimated UTEST flash memory space the client occupies, in bytes

DCF clients are found in the following modules:

- STCU
- PASS
- Tamper Detect

Some other DCF clients are classified as Miscellaneous. See description of the individual clients in the table.

Table 6-5. DCF Clients

DCF CS[14:0]	DCF Address [16:2]	DCF Client Description	Reset Value of DCF Client	IPS Read Direct Access	DCF Client Special Strategy	Flash Special Strategy	DCF Order Dependency	UTEST Flash Memory Space
STCU - Support for 4 LBIST partitions; 48 Memory Cuts; 64-bit PRPG; 64-bit MISR								
000_0000_0000_0100	0000000_000010	STCU_SKC	0x00000000	yes	None	None	First Record among STCU, Key1/Key2 dcf record sequence; Write Key2 every (1024-8) clock edges	16
000_0000_0000_0100	0000000_000000	STCU_RUN	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000000_000011	STCU_CFG	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000000_000100	STCU_PLL_CFG	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000000_000101	STCU_WDG	0x0000FFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_000111	STCU_CRCE	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000000_001010	STCU_ERR_FM	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000000_001111	STCU_LBRMSW	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000000_010000	STCU_LBUFM	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000000_011101	STCU_MBUFML[31:0]	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000000_011110	STCU_MBUFMM[31:0]	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000000_010000	STCU_LB_CTRL_0	0x00000000	yes	None	None	None	8

Table continues on the next page...

Table 6-5. DCF Clients (continued)

DCF CS[14:0]	DCF Address [16:2]	DCF Client Description	Reset Value of DCF Client	IPS Read Direct Access	DCF Client Special Strategy	Flash Special Strategy	DCF Order Dependency	UTEST Flash Memory Space
000_0000_0000_0100	0000000_01000001	STCU_LB_PCS_0	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000000_01000010	STCU_LB_PRPGL_0	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_01000011	STCU_LB_PRPGH_0	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_01000100	STCU_LB_MISREL_0	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_01000101	STCU_LB_MISREH_0	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_01001000	STCU_LB_MISRELSW_0	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_01001001	STCU_LB_MISREHSW_0	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_01010000	STCU_LB_CTRL_1	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000000_01010001	STCU_LB_PCS_1	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000000_01010010	STCU_LB_PRPGL_1	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_01010011	STCU_LB_PRPGH_1	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_01010100	STCU_LB_MISREL_1	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_01010101	STCU_LB_MISREH_1	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_01011000	STCU_LB_MISRELSW_1	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_01011001	STCU_LB_MISREHSW_1	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_01100000	STCU_LB_CTRL_2	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000000_01100001	STCU_LB_PCS_2	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000000_01100010	STCU_LB_PRPGL_2	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_01100011	STCU_LB_PRPGH_2	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_01100100	STCU_LB_MISREL_2	0xFFFFFFFF	yes	None	None	None	8

Table continues on the next page...

Table 6-5. DCF Clients (continued)

DCF CS[14:0]	DCF Address [16:2]	DCF Client Description	Reset Value of DCF Client	IPS Read Direct Access	DCF Client Special Strategy	Flash Special Strategy	DCF Order Dependency	UTEST Flash Memory Space
000_0000_0000_0100	0000000_01100101	STCU_LB_MISREH_2	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_01101000	STCU_LB_MISRELSW_2	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_01101001	STCU_LB_MISREHSW_2	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_01110000	STCU_LB_CTRL_3	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000000_01110001	STCU_LB_PCS_3	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000000_01110010	STCU_LB_PRPGL_3	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_01110011	STCU_LB_PRPGH_3	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_01110100	STCU_LB_MISREL_3	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_01110101	STCU_LB_MISREH_3	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_01111000	STCU_LB_MISRELSW_3	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000000_01111001	STCU_LB_MISREHSW_3	0xFFFFFFFF	yes	None	None	None	8
000_0000_0000_0100	0000001_10000000	STCU_MB_CTRL_0	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10000001	STCU_MB_CTRL_1	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10000010	STCU_MB_CTRL_2	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10000011	STCU_MB_CTRL_3	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10000100	STCU_MB_CTRL_4	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10000101	STCU_MB_CTRL_5	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10000110	STCU_MB_CTRL_6	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10000111	STCU_MB_CTRL_7	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10001000	STCU_MB_CTRL_8	0x00000000	yes	None	None	None	8

Table continues on the next page...

Table 6-5. DCF Clients (continued)

DCF CS[14:0]	DCF Address [16:2]	DCF Client Description	Reset Value of DCF Client	IPS Read Direct Access	DCF Client Special Strategy	Flash Special Strategy	DCF Order Dependency	UTEST Flash Memory Space
000_0000_0000_0100	0000001_10001001	STCU_MB_CTRL_9	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10001010	STCU_MB_CTRL_10	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10001011	STCU_MB_CTRL_11	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10001100	STCU_MB_CTRL_12	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10001101	STCU_MB_CTRL_13	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10001110	STCU_MB_CTRL_14	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10001111	STCU_MB_CTRL_15	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10010000	STCU_MB_CTRL_16	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10010001	STCU_MB_CTRL_17	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10010010	STCU_MB_CTRL_18	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10010011	STCU_MB_CTRL_19	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10010100	STCU_MB_CTRL_20	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10010101	STCU_MB_CTRL_21	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10010110	STCU_MB_CTRL_22	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10010111	STCU_MB_CTRL_23	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10011000	STCU_MB_CTRL_24	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10011001	STCU_MB_CTRL_25	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10011010	STCU_MB_CTRL_26	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10011011	STCU_MB_CTRL_27	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10011100	STCU_MB_CTRL_28	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10011101	STCU_MB_CTRL_29	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10011110	STCU_MB_CTRL_30	0x00000000	yes	None	None	None	8

Table continues on the next page...

Table 6-5. DCF Clients (continued)

DCF CS[14:0]	DCF Address [16:2]	DCF Client Description	Reset Value of DCF Client	IPS Read Direct Access	DCF Client Special Strategy	Flash Special Strategy	DCF Order Dependency	UTEST Flash Memory Space
000_0000_0000_0100	0000001_10011111	STCU_MB_CTRL_31	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10100000	STCU_MB_CTRL_32	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10100001	STCU_MB_CTRL_33	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10100010	STCU_MB_CTRL_34	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10100011	STCU_MB_CTRL_35	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10100100	STCU_MB_CTRL_36	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10100101	STCU_MB_CTRL_37	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10100110	STCU_MB_CTRL_38	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10100111	STCU_MB_CTRL_39	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10101000	STCU_MB_CTRL_40	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10101001	STCU_MB_CTRL_41	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10101010	STCU_MB_CTRL_42	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10101011	STCU_MB_CTRL_43	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10101100	STCU_MB_CTRL_44	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10101101	STCU_MB_CTRL_45	0x00000000	yes	None	None	None	8
000_0000_0000_0100	0000001_10101110	STCU_MB_CTRL_46	0x00000000	yes	None	None	None	8
PASS								
000_0000_0000_1000	0000000_00000000-0000000_00101011	Reserved						
000_0000_0000_1000	0000000_00101100	Censorship	0x0000	no[1]	None	None	None	8
000_0000_0000_1000	0000000_00101101-0000000_00101111	Reserved						

Table continues on the next page...

Table 6-5. DCF Clients (continued)

DCF CS[14:0]	DCF Address [16:2]	DCF Client Description	Reset Value of DCF Client	IPS Read Direct Access	DCF Client Special Strategy	Flash Special Strategy	DCF Order Dependency	UTEST Flash Memory Space
000_0000_0000_1000	0000000_00110000	Production Disable	0x00000000	no	Write Once	None	None	8
000_0000_0000_1000	0000000_00110001-0000000_00111111	Reserved						
000_0000_0000_1000	0000000_01000000	LOCK0_PG0	0xFFFFFFFF	yes[2]	None	None	None	8
000_0000_0000_1000	0000000_01000001	LOCK1_PG0	0xFFFFFFFF	yes[2]	None	None	None	8
000_0000_0000_1000	0000000_01000010	LOCK2_PG0	0xFFFFFFFF	yes[2]	None	None	None	8
000_0000_0000_1000	0000000_01000011	LOCK3_PG0	0xFFFFFFFF	yes[2]	None	None	None	8
000_0000_0000_1000	0000000_01000100	LOCK0_PG1	0xFFFFFFFF	yes[2]	None	None	None	8
000_0000_0000_1000	0000000_01000101	LOCK1_PG1	0xFFFFFFFF	yes[2]	None	None	None	8
000_0000_0000_1000	0000000_01000110	LOCK2_PG1	0xFFFFFFFF	yes[2]	None	None	None	8
000_0000_0000_1000	0000000_01000111	LOCK3_PG1	0xFFFFFFFF	yes[2]	None	None	None	8
000_0000_0000_1000	0000000_01001000	LOCK0_PG2	0xFFFFFFFF	yes[2]	None	None	None	8
000_0000_0000_1000	0000000_01001001	LOCK1_PG2	0xFFFFFFFF	yes[2]	None	None	None	8
000_0000_0000_1000	0000000_01001010	LOCK2_PG2	0xFFFFFFFF	yes[2]	None	None	None	8
000_0000_0000_1000	0000000_01001011	LOCK3_PG2	0xFFFFFFFF	yes[2]	None	None	None	8
000_0000_0000_1000	0000000_01001100	LOCK0_PG3	0xFFFFFFFF	yes[2]	None	None	None	8
000_0000_0000_1000	0000000_01001101	LOCK1_PG3	0xFFFFFFFF	yes[2]	None	None	None	8
000_0000_0000_1000	0000000_01001110	LOCK2_PG3	0xFFFFFFFF	yes[2]	None	None	None	8
000_0000_0000_1000	0000000_01001111	LOCK3_PG3	0xFFFFFFFF	yes[2]	None	None	None	8
000_0000_0000_1000	0000000_01001100-11111111	Reserved						
Tamper Detect								

Table continues on the next page...

Table 6-5. DCF Clients (continued)

DCF CS[14:0]	DCF Address [16:2]	DCF Client Description	Reset Value of DCF Client	IPS Read Direct Access	DCF Client Special Strategy	Flash Special Strategy	DCF Order Dependency	UTEST Flash Memory Space
000_0000_0001_0000	0000000_00000000	Diary Base Address	0xFFFFFFFF	no	Write Once + Write '0' only	None	None	8
000_0000_0001_0000	0000000_00000001	Tamper Region Override	0x00	no	Write '1' only	None	None	8
000_0000_0001_0000	0000000_00000010-00000111	Reserved						
000_0000_0001_0000	0000000_00001000	OTP_EN0	0x00000000	no	Write '1' only	None	None	8
000_0000_0001_0000	0000000_00001001	OTP_EN1	0x00000000	no	Write '1' only	None	None	8
000_0000_0001_0000	0000000_00001010	OTP_EN2	0x00000000	no	Write '1' only	None	None	8
000_0000_0001_0000	0000000_00001011	OTP_EN3	0x00000000	no	Write '1' only	None	None	8
000_0000_0001_0000	0000000_00001100-00010011	Reserved						
000_0000_0001_0000	0000000_00010100	TDR0_LOCK0	0x00000000	no	Write '1' only	None	None	8
000_0000_0001_0000	0000000_00010101	TDR0_LOCK1	0x00000000	no	Write '1' only	None	None	8
000_0000_0001_0000	0000000_00010110	TDR0_LOCK2	0x00000000	no	Write '1' only	None	None	8
000_0000_0001_0000	0000000_00010111	TDR0_LOCK3	0x00000000	no	Write '1' only	None	None	8
000_0000_0001_0000	0000000_00011000	TDR1_LOCK0	0x00000000	no	Write '1' only	None	None	8
000_0000_0001_0000	0000000_00011001	TDR1_LOCK1	0x00000000	no	Write '1' only	None	None	8
000_0000_0001_0000	0000000_00011010	TDR1_LOCK2	0x00000000	no	Write '1' only	None	None	8
000_0000_0001_0000	0000000_00011011	TDR1_LOCK3	0x00000000	no	Write '1' only	None	None	8
000_0000_0001_0000	0000000_00011100	TDR2_LOCK0	0x00000000	no	Write '1' only	None	None	8
000_0000_0001_0000	0000000_00011101	TDR2_LOCK1	0x00000000	no	Write '1' only	None	None	8
000_0000_0001_0000	0000000_00011110	TDR2_LOCK2	0x00000000	no	Write '1' only	None	None	8
000_0000_0001_0000	0000000_00011111	TDR2_LOCK3	0x00000000	no	Write '1' only	None	None	8

Table continues on the next page...

Table 6-5. DCF Clients (continued)

DCF CS[14:0]	DCF Address [16:2]	DCF Client Description	Reset Value of DCF Client	IPS Read Direct Access	DCF Client Special Strategy	Flash Special Strategy	DCF Order Dependency	UTEST Flash Memory Space
000_0000_0001_0000	0000000_00100000	TDR3_LOCK0	0x00000000	no	Write ‘1’ only	None	None	8
000_0000_0001_0000	0000000_00100001	TDR3_LOCK1	0x00000000	no	Write ‘1’ only	None	None	8
000_0000_0001_0000	0000000_00100010	TDR3_LOCK2	0x00000000	no	Write ‘1’ only	None	None	8
000_0000_0001_0000	0000000_00100011	TDR3_LOCK3	0x00000000	no	Write ‘1’ only	None	None	8
000_0000_0001_0000	0000000_00100100	TDR4_LOCK0	0x00000000	no	Write ‘1’ only	None	None	8
000_0000_0001_0000	0000000_00100101	TDR4_LOCK1	0x00000000	no	Write ‘1’ only	None	None	8
000_0000_0001_0000	0000000_00100110	TDR4_LOCK2	0x00000000	no	Write ‘1’ only	None	None	8
000_0000_0001_0000	0000000_00100111	TDR4_LOCK3	0x00000000	no	Write ‘1’ only	None	None	8
000_0000_0001_0000	0000000_00101000	TDR5_LOCK0	0x00000000	no	Write ‘1’ only	None	None	8
000_0000_0001_0000	0000000_00101001	TDR5_LOCK1	0x00000000	no	Write ‘1’ only	None	None	8
000_0000_0001_0000	0000000_00101010	TDR5_LOCK2	0x00000000	no	Write ‘1’ only	None	None	8
000_0000_0001_0000	0000000_00101011	TDR5_LOCK3	0x00000000	no	Write ‘1’ only	None	None	8
000_0000_0001_0000	0000000_00101100-11111111	Reserved						
MISCELLANEOUS								
000_0000_1000_0000	0000000_00001000	DCF_UTEST_Miscellaneous	0x00028500	no	SPRD, Triple voting, and PARITY	None	None	24
000_0000_1000_0000	0000000_00010000	DCF_PMC_REE_CTRL	0x00014052	no	SPRD, Triple voting, and PARITY	None	None	24
000_0000_1000_0000	0000000_00011000	DCF_PMC_RES_CTRL	0x00000000	no	SPRD, Triple voting, and PARITY	None	None	24
000_0000_1000_0000	0000000_00100000	Reserved						

Table continues on the next page...

Table 6-5. DCF Clients (continued)

DCF CS[14:0]	DCF Address [16:2]	DCF Client Description	Reset Value of DCF Client	IPS Read Direct Access	DCF Client Special Strategy	Flash Special Strategy	DCF Order Dependency	UTEST Flash Memory Space
000_0000_1000_0000	0000000_00101000	DCF_PMC_REE_TD	0x00000000	no	SPRD, Triple voting, and PARITY	None	0	24
000_0000_1000_0000	0000000_00110000	DCF_IVPR0	0x50000000	no	SPRD, Triple voting, and PARITY	None	0	24
000_0000_1000_0000	0000000_00111000	DCF_IVPR1	0x51000000	no	SPRD, Triple voting, and PARITY	None	0	24
000_0001_0000_0000	0000000_00001000	DCF_SELFTEST_CONFIG_IPS_1	0x00010000	yes	SPRD and Triple voting	None	None	24
000_0001_0000_0000	0000000_00010000	DCF_SELFTEST_CONFIG_IPS_2	0x00000000	yes	SPRD and Triple voting	None	None	24
000_0001_0000_0000	0000000_00011000	DCF_SELFTEST_CONFIG_IPS_3	0x00000000	yes	SPRD and Triple voting	None	None	24
000_0001_0000_0000	0000000_00100000	DCF_SELFTEST_CONFIG_IPS_4	0x00DF7DF7	yes	SPRD and Triple voting	None	None	24
000_0001_0000_0000	0000000_00101000	Reserved						
000_0001_0000_0000	0000000_00110000	DCF_ERROR0_PAD_SELECT	0x00000000	no	SPRD, Triple voting, and PARITY	None	None	24
000_0001_0000_0000	0000000_00111000	DCF_ERROR1_PAD_SELECT	0x00000000	no	SPRD, Triple voting, and PARITY	None	None	24

1. Can be updated from DCF Bus (FLASH DCF records or SSCM test access port).
2. Can be accessed only from IPS. Password group has to be unlocked.

6.7.1 Miscellaneous DCF clients

This section describes the DCF clients classified in the Miscellaneous category.

The DCF_SELFTEST_CONFIG_IPS_n DCF clients are memory-mapped. All other DCF clients here are not memory-mapped.

6.7.1.1 DCF_UTEST_Miscellaneous DCF client

The DCF_UTEST_Miscellaneous register controls miscellaneous functionalities.

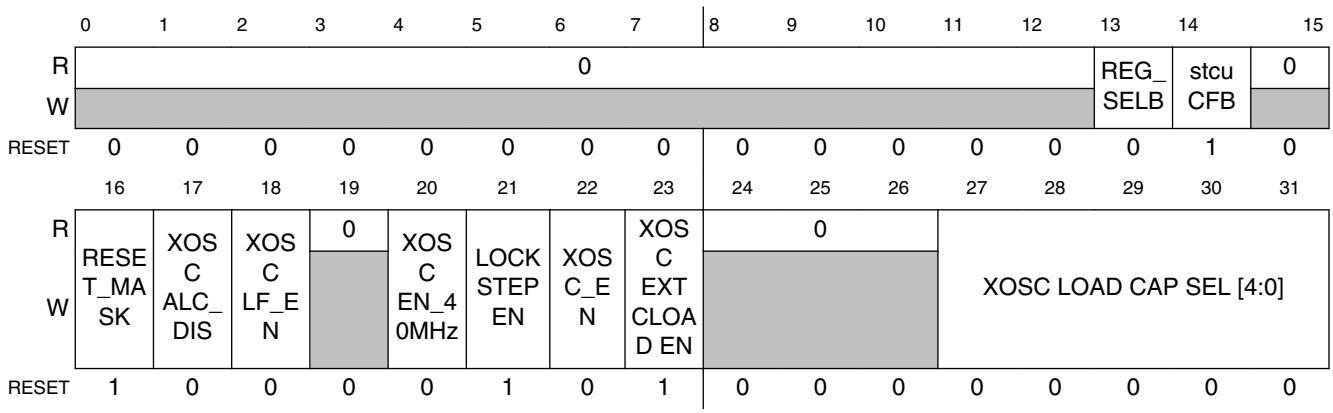


Table 6-7. DCF_UTEST_Miscellaneous Field Description

Field	Name	Description
0-12	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13	REG_SELB	If this bit is set to 0, internal regulation will be selected. If not external regulation is selected.
14	stcu CFB	If this is set to 1, critical fault during offline LBIST run will generate a destructive reset.
15	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16	RESET_MASK	Active Low reset Mask bit. 1- RESET PIN enabled, 0-RESET Pin Disabled Level Controller
17	XOSC ALC_DIS	Active high to disable Automatic Level Controller
18	XOSC LF_EN	Active high when oscillator is used for low frequency (~8MHz)
19	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20	XOSC EN_40MHz	Selects 20MHz against 40MHz XOSC. Set this bit to enable the 40Mhz mode.
21	LOCKSTEP EN	Enables Safety Core to be in lockstep with Checker Core. Value 0 = Lockstep disabled, Value 1 = Lockstep enabled
22	XOSC_EN	Enables XOSC.
23	XOSC EXT CLOAD EN	Selects XOSC internal capacitor (ok for 20-40MHz operation) or external (required for 4-16MHz operation). Set to 1 if external capacitors are used.
24-26	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27-31	XOSC LOAD CAP SEL [4:0]	5 Trim bits to program the load capacitance at crystal pins

6.7.1.2 DCF_PMC_REE_CTRL DCF client

The DCF_PMC_REE_CTRL register controls whether or not the voltage detect signal causes a reset.

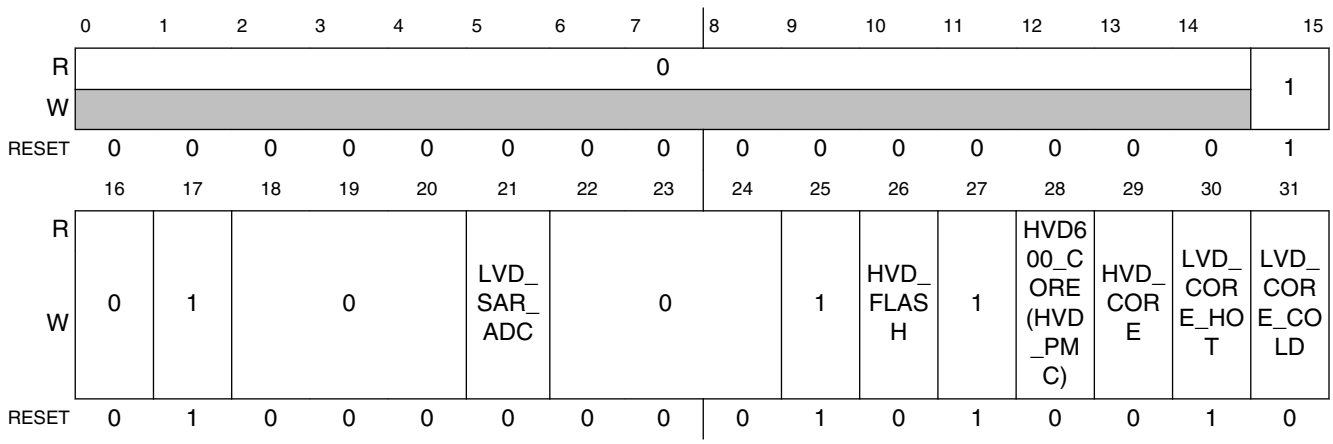


Table 6-9. DCF_PMC_REE_CTRL Field Description

Field	Name	Description
0-14	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15-20	Reserved	This field is reserved. **NOTE:** Always write the reset value to this field.
21	LVD_SAR_ADC	• 0: Disabled (The voltage detect signal does not cause a reset) • 1: Enabled (The voltage detect signal causes a reset)
22-24	Reserved	This field is reserved. **NOTE:** Always write the reset value to this field.
25	Reserved	This field is reserved. **NOTE:** Always write the reset value to this field.
26	HVD_FLASH	• 0: Disabled (The voltage detect signal does not cause a reset) • 1: Enabled (The voltage detect signal causes a reset)
27	Reserved	This field is reserved. **NOTE:** Always write the reset value to this field.
28	HVD600_CORE (HVD_PMC)	• 0: Disabled (The voltage detect signal does not cause a reset) • 1: Enabled (The voltage detect signal causes a reset)
29	HVD_CORE	• 0: Disabled (The voltage detect signal does not cause a reset) • 1: Enabled (The voltage detect signal causes a reset)
30	LVD_CORE_HOT	• 0: Disabled (The voltage detect signal does not cause a reset) • 1: Enabled (The voltage detect signal causes a reset)
31	LVD_CORE_COLD	• 0: Disabled (The voltage detect signal does not cause a reset) • 1: Enabled (The voltage detect signal causes a reset)

6.7.1.3 DCF_PMC_RES_CTRL DCF client

The DCF_PMC_RES_CTRL register controls whether voltage detect signal causes a destructive reset or a functional reset.

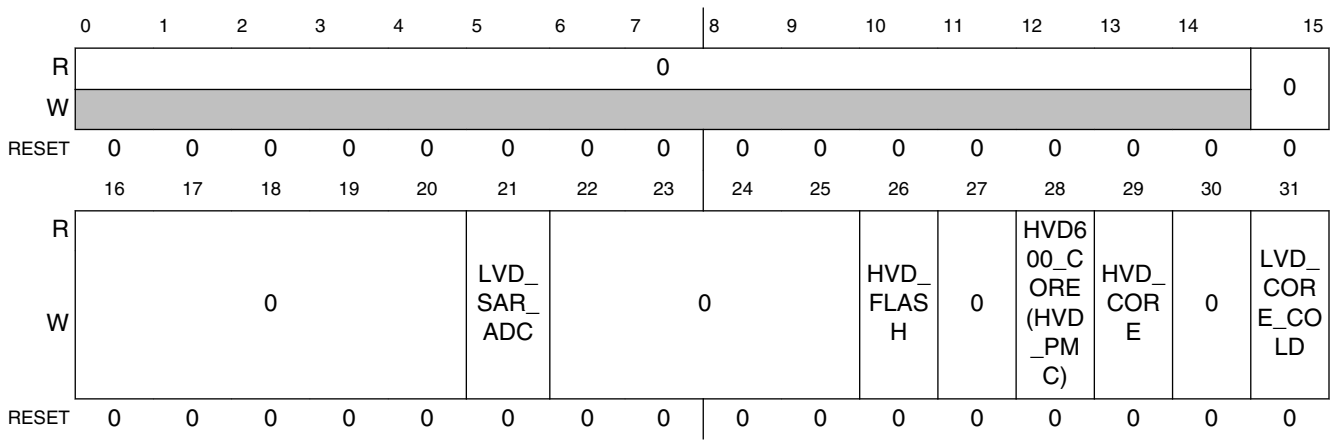


Table 6-11. DCF_PMC_RES_CTRL Field Description

Field	Name	Description
0-14	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15-20	Reserved	This field is reserved. **NOTE:** Always write the reset value to this field.
21	LVD_SAR_ADC	• 0: The voltage detect signal causes a destructive reset • 1: The voltage detect signal causes a functional reset
22-25	Reserved	This field is reserved. **NOTE:** Always write the reset value to this field.
26	HVD_FLASH	• 0: The voltage detect signal causes a destructive reset • 1: The voltage detect signal causes a functional reset
27	Reserved	This field is reserved. **NOTE:** Always write the reset value to this field.
28	HVD600_CORE (HVD_PMC)	• 0: The voltage detect signal causes a destructive reset • 1: The voltage detect signal causes a functional reset
29	HVD_CORE	• 0: The voltage detect signal causes a destructive reset • 1: The voltage detect signal causes a functional reset
30	Reserved	This field is reserved. **NOTE:** Always write the reset value to this field.
31	LVD_CORE_COLD	• 0: The voltage detect signal causes a destructive reset • 1: The voltage detect signal causes a functional reset

6.7.1.4 DCF_PMC_REE_TD DCF client

The DCF_PMC_REE_TD register controls whether the temperature detect signal event causes a reset.

If the desired flag bit is enabled, the temperature detect event causes a reset to be generated when the selected temperature passes the trigger event.

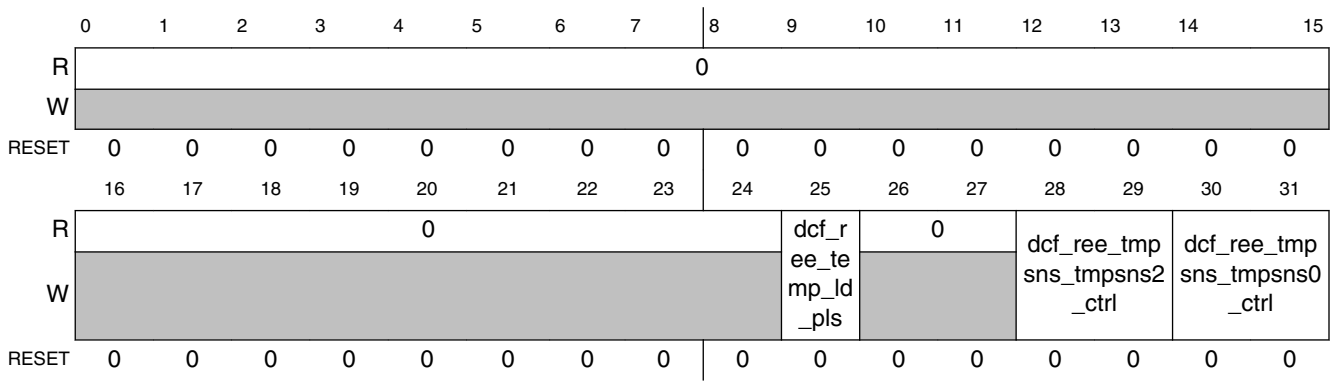


Table 6-13. DCF_PMC_REE_TD Field Description

Field	Name	Description
0-24	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25	dcf_ree_temp_ld_pls	Set this bit to 1 so that values uploaded becomes effective.
26-27	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28-29	dcf_ree_tmpsns_tmpsns2_ctrl	Defines whether a temperature detect assertion generates a system reset. This is associated with the temperature for the temperature sensor point (150C). • 00: Temperature sensor detect event does not cause system reset. • 01: Temperature sensor 0 detect causes system reset. • 10: Temperature sensor 1 detect causes system reset. • 11: Temperature sensor 0 and 1 detect causes system reset.
30-31	dcf_ree_tmpsns_tmpsns0_ctrl	Defines whether a temperature detect assertion generates a system reset. This is associated with the temperature for the cold temperature sensor,when the temperature exceeds its corresponding threshold. • 00: Temperature sensor detect event does not cause system reset. • 01: Temperature sensor 0 detect causes system reset. • 10: Temperature sensor 1 detect causes system reset. • 11: Temperature sensor 0 and 1 detect causes system reset.

6.7.1.5 DCF_IVPR0 DCF client

The DCF_IVPR0 is the reset interrupt vector prefix register DCF client for Main Core_0.

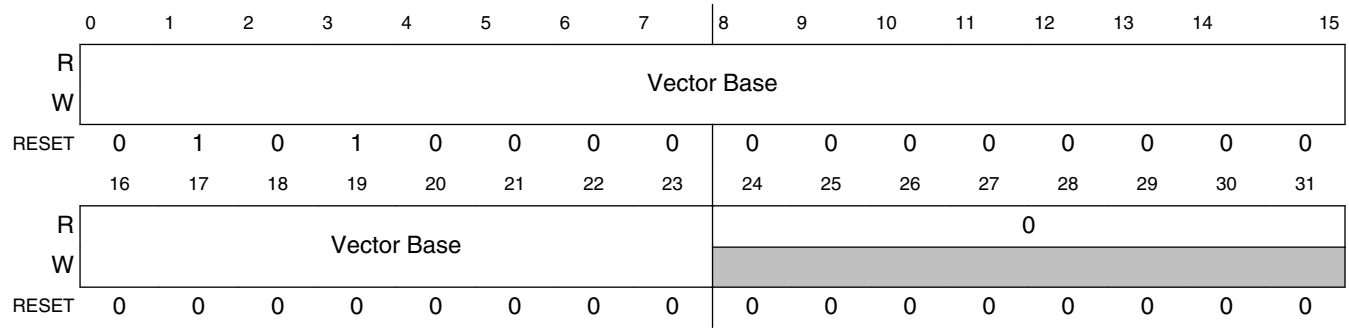


Table 6-15. DCF_IVPR0 Field Description

Field	Name	Description
0-23	Vector Base	This field is used to define the base location of the vector table, aligned to a 256 byte boundary. This field provides the high-order 24 bits of the location of all interrupt handlers (unless hardware vectoring is used). The vector offset value appropriate for the type of exception being processed is concatenated with the IVPR Vector Base to form the address of the handler in memory. For hardware-vectored interrupts, the value of the supplied interrupt vector is logically OR'ed with the low order bits of the Vec Base Field to determine the interrupt handler address. **NOTE:** dcf_ivpr[23:0] -> core.ivpr[0:23]
24-31	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

6.7.1.6 DCF_IVPR1 DCF client

The DCF_IVPR1 is the reset interrupt vector prefix register DCF client for Main Core_1.

NOTE

- This DCF client is put into effect only after BAF is executed and the application code starts executing.
- The value in this DCF client is left-shifted by 8 bits before programmed to the Core_1 IVPR.

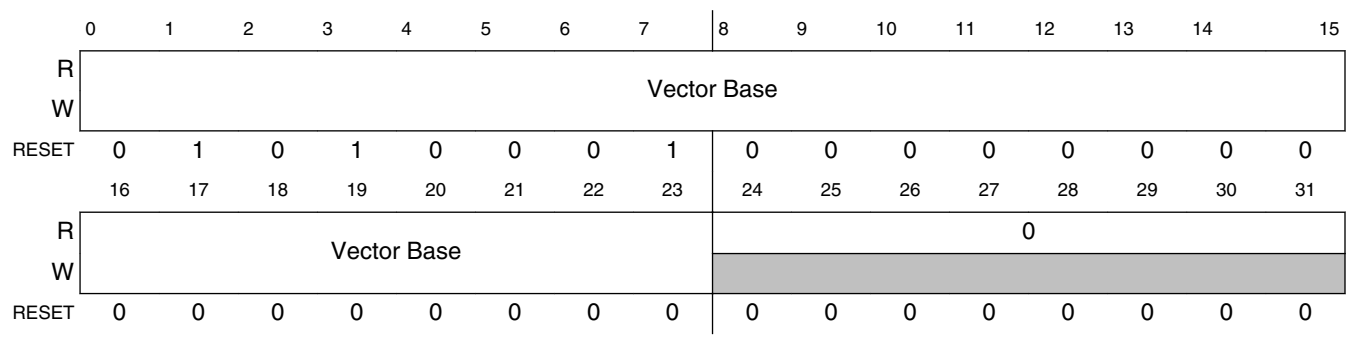


Table 6-17. DCF_IVPR1 Field Description

Field	Name	Description
0-23	Vector Base	This field is used to define the base location of the vector table, aligned to a 256 byte boundary. This field provides the high-order 24 bits of the location of all interrupt handlers (unless hardware vectoring is used). The vector offset value appropriate for the type of exception being processed is concatenated with the IVPR Vector Base to form the address of the handler in memory. For hardware-vectored interrupts, the value of the supplied interrupt vector is logically OR'ed with the low order bits of the Vec Base Field to determine the interrupt handler address. **NOTE:** dcf_ivpr[23:0] -> core.ivpr[0:23]
24-31	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

6.7.1.7 DCF_SELFTEST_CONFIG_IPS_1 DCF client

The DCF_SELFTEST_CONFIG_IPS_1 register controls the fault Model for LBIST and selects which PLL would be used for LBIST testing if any.

Address: 0x11_1111_1110_0000 base + 0x0 offset

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0									PLL_LOCK 1	PLL_LOCK 0	0			1
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			config[1]	0											
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-19. DCF_SELFTEST_CONFIG_IPS_1 Field Description

Field	Name	Description
0	Reserved	This field is reserved. Set this bit when running LBIST.
1-9	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10	PLL_LOCK1	Program this bit to "1" if PLL1 is to be used for Selftest. Only one of bits 20 and 21 should be programmed "1".
11	PLL_LOCK0	Program this bit to "1" if PLL0 is to be used for Selftest . Only one of bits 20 and 21 should be programmed "1".
12-14	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15	Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
16-18	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

Table 6-19. DCF_SELFTEST_CONFIG_IPS_1 Field Description (continued)

Field	Name	Description
19	config[1]	Program this bit to "1" if stuck-At LBIST is required to be run. In this mode PLL is locked at 200 Mhz but all partitions are run at 50 Mhz at 4 seperate edges of 200 Mhz so as to reduce the peak power.
20-31	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

6.7.1.8 DCF_SELFTEST_CONFIG_IPS_2 DCF client

The DCF_SELFTEST_CONFIG_IPS_2 register controls the division values for the dividers.

The division value set by these fields is the bit value + 1. (0b000 sets the division value to 1, 0b001 sets the division value to 2, and so on.)

Address: 0x11_1111_1110_0000 base + 0x100 offset

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0		Aux_MUX_0_Divider_4_Division_Value				Aux_MUX_0_Divider_3_Division_Value		Aux_MUX_0_Divider_2_Division_Value			Aux_MUX_0_Divider_1_Division_Value			Aux_MUX_0_Divider_0_Division_Value	
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	(Aux_MUX_0_Divider_0_Division_Value)	0			System_MUX_Divider_3_Division_Value		System_MUX_Divider_2_Division_Value		System_MUX_Divider_1_Division_Value			System_MUX_Divider_0_Division_Value				
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-21. DCF_SELFTEST_CONFIG_IPS_2 Field Description

Field	Name	Description
0-1	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2-5	Aux_MUX_0_Divider_4_Division_Value	Sets the division value of the divider.
6-7	Aux_MUX_0_Divider_3_Division_Value	Sets the division value of the divider.
8-10	Aux_MUX_0_Divider_2_Division_Value	Sets the division value of the divider.

Table continues on the next page...

Table 6-21. DCF_SELFTEST_CONFIG_IPS_2 Field Description (continued)

Field	Name	Description
11-13	Aux_MUX_0_Divider_1_Division_Value	Sets the division value of the divider.
14-16	Aux_MUX_0_Divider_0_Division_Value	Sets the division value of the divider.
17-19	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20-22	System_MUX_Divider_3_Division_Value	Sets the division value of the divider.
23-25	System_MUX_Divider_2_Division_Value	Sets the division value of the divider.
26-28	System_MUX_Divider_1_Division_Value	Sets the division value of the divider.
29-31	System_MUX_Divider_0_Division_Value	Sets the division value of the divider.

6.7.1.9 DCF_SELFTEST_CONFIG_IPS_3 DCF client

The DCF_SELFTEST_CONFIG_IPS_3 register controls the division values for the dividers.

The division value set by these fields is the bit value + 1. (0b000 sets the division value to 1, 0b001 sets the division value to 2, and so on.)

Address: 0x11_1111_1110_0000 base + 0x1000 offset

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0					Aux_Mux_10_Divider_0_Division_Value			Aux_Mux_9_Divider_0_Division_Value			Aux_Mux_8_Divider_0_Division_Value			0	
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	Aux_Mux_6_Divider_0_Division_Value			Aux_Mux_5_Divider_1_Division_Value			Aux_Mux_5_Divider_0_Division_Value			Aux_Mux_2_Divider_0_Division_Value			Aux_Mux_1_Divider_0_Division_Value		
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-23. DCF_SELFTEST_CONFIG_IPS_3 Field Description

Field	Name	Description
0-4	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5-7	Aux_Mux_10_Divider_0_Division_Value	Sets the division value of the divider.

Table continues on the next page...

Table 6-23. DCF_SELFTEST_CONFIG_IPS_3 Field Description (continued)

Field	Name	Description
8-10	Aux_Mux_9_Divider_0_Division_Value	Sets the division value of the divider.
11-13	Aux_Mux_8_Divider_0_Division_Value	Sets the division value of the divider.
14-16	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17-19	Aux_Mux_6_Divider_0_Division_Value	Sets the division value of the divider.
20-22	Aux_Mux_5_Divider_1_Division_Value	Sets the division value of the divider.
23-25	Aux_Mux_5_Divider_0_Division_Value	Sets the division value of the divider.
26-28	Aux_Mux_2_Divider_0_Division_Value	Sets the division value of the divider.
29-31	Aux_Mux_1_Divider_0_Division_Value	Sets the division value of the divider.

6.7.1.10 DCF_SELFTEST_CONFIG_IPS_4 DCF client

The DCF_SELFTEST_CONFIG_IPS_4 register controls the LBIST shift length for each partition.

Address: 0x11_1111_1110_0000 base + 0x1100 offset

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								LBIST_W1_Shift_Length						LBIST_W0_Shift_Length	
W																
RESET	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	1

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	(LBIST_W0_Shift_Length)				LBIST_L1_Shift_Length						LBIST_L0_Shift_Length					
W																
RESET	0	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1

Table 6-25. DCF_SELFTEST_CONFIG_IPS_4 Field Description

Field	Name	Description
0-7	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8-13	LBIST_W1_Shift_Length	Sets the LBIST shift length of the W1 partition.
14-19	LBIST_W0_Shift_Length	Sets the LBIST shift length of the W0 partition.

Table continues on the next page...

Table 6-25. DCF_SELFTEST_CONFIG_IPS_4 Field Description (continued)

Field	Name	Description
20-25	LBIST_L1_Shift_Length	Sets the LBIST shift length of the L1 partition.
26-31	LBIST_L0_Shift_Length	Sets the LBIST shift length of the L0 partition.

6.7.1.11 DCF_ERROR0_PAD_SELECT DCF client

The DCF_ERROR0_PAD_SELECT_CONTROL register selects either the functional muxing or the EOUT functionality on pads PJ[11], PJ[1], PI[0], and PH[0]. This is done at boot time according to the SELn bits of this DCF client:

- If ERROR0 OUT is selected (i.e., the SELn bits are set), the PAD controls of the pad are governed by this DCF client.
- If functional muxing is selected (i.e., the SELn bits are not set), the PAD controls are governed by the SIUL2_MSCRn register.

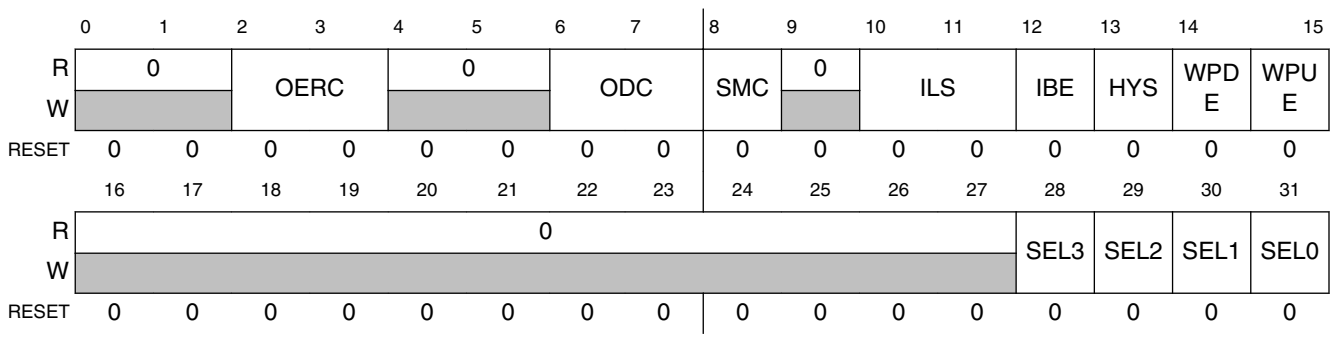


Table 6-27. DCF_ERROR0_PAD_SELECT_CONTROL Field Descriptions

Field	Name	Description
0-1	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2-3	OERC	Sets the Output Edge Rate Control when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
4-5	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6-7	ODC	Sets the Output Drive Control when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
8	SMC	Sets the Safe Mode Control when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
9	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10-11	ILS	Sets the Input Level Selection when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
12	IBE	Sets the Input Buffer Enable when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.

Table continues on the next page...

Table 6-27. DCF_ERROR0_PAD_SELECT_CONTROL Field Descriptions (continued)

Field	Name	Description
13	HYS	Sets the Input Hysteresis when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
14	WPDE	Sets the Weak Pulldown Enable when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
15	WPUE	Sets the Weak Pullup Enable when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
16-27	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28	SEL3	Enable ERROR0 OUT on PJ[11] irrespective of SIUL2_MSCRn[SSS] settings and override the pad setting as defined in this register.
29	SEL2	Enable ERROR0 OUT on PJ[1] irrespective of SIUL2_MSCRn[SSS] settings and override the pad setting as defined in this register
30	SEL1	Enable ERROR0 OUT on PI[0] irrespective of SIUL2_MSCRn[SSS] settings and override the pad setting as defined in this register
31	SEL0	Enable ERROR0 OUT on PH[0] irrespective of SIUL2_MSCRn[SSS] settings and override the pad setting as defined in this register

There is no hardware restriction on the programming of the SELn bits. All the four pads can be simultaneously be programmed as ERROR0 OUT Pins.

6.7.1.12 DCF_ERROR1_PAD_SELECT DCF client

The DCF_ERROR1_PAD_SELECT_CONTROL register selects either the functional muxing or the EOUT functionality on pads PJ[0], PI[1], and PH[3]. This is done at boot time according to the SELn bits of this DCF client:

- If ERROR1 OUT is selected (i.e., the SELn bits are set), the PAD controls of the pad are governed by this DCF client.
- If functional muxing is selected (i.e., the SELn bits are not set), the PAD controls are governed by the SIUL2_MSCR register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0		OERC		0		ODC		SMC	0	ILS		IBE	HYS	WPDE	WPUE
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0													SEL2	SEL1	SEL0
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-29. DCF_ERROR1_PAD_SELECT_CONTROL Field Descriptions

Field	Name	Description
0-1	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2-3	OERC	Sets the Output Edge Rate Control when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
4-5	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6-7	ODC	Sets the Output Drive Control when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
8	SMC	Sets the Safe Mode Control when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
9	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10-11	ILS	Sets the Input Level Selection when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
12	IBE	Sets the Input Buffer Enable when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
13	HYS	Sets the Input Hysteresis when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
14	WPDE	Sets the Weak Pulldown Enable when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
15	WPUE	Sets the Weak Pullup Enable when ERROR0 OUT is enabled. The function is the same as the corresponding field in SiUL2_MSCRn.
16-28	Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29	SEL2	Enable ERROR1 OUT on PJ[0] irrespective of SIUL2_MSCRn[SSS] settings and override the pad setting as defined in this register.
30	SEL1	Enable ERROR1 OUT on PI[1] irrespective of SIUL2_MSCRn[SSS] settings and override the pad setting as defined in this register.
31	SEL0	Enable ERROR1 OUT on PH[3] irrespective of SIUL2_MSCRn[SSS] settings and override the pad setting as defined in this register.

There is no hardware restriction on the programming of the SELn bits. All three pads can be simultaneously be programmed as ERROR1 OUT Pins.

Chapter 7
e200z4d Core Complex Overview

7.1 Introduction

The main computational shell consists of Main Core_0 and Main Core_1. Both cores use an e200z425n3 core. A third core, referred to as Checker Core_0s, is an e200z424 core that is a true subset of the e200z425n3 core. When enabled, Checker Core_0s operates in lock step mode with Main Core_0, executing exactly the same instructions as Main Core_0. Thus, Checker Core_0s checks to insure that Main Core_0 is executing correctly. This chapter provides overviews of the e200z425n3 core and the e200z424 core.

The e200z425n3 and the e200z424 cores are very similar. The register set for these cores is identical. The e200z424 core differs in hardware from the e200z425n3 in that no cache or D-memory is implemented.

7.2 Overview of the e200z425n3 and e200z424 cores

The e200z processor family is a set of CPU cores that implement low-cost versions of the Power architecture.

The e200z425n3 and e200z424 are dual-issue, 32-bit Power VLE compliant designs with 32-bit general purpose registers (GPRs).

An Embedded Floating-point (EFPU2) APU is provided to support real-time single-precision embedded numeric operations using the general-purpose registers.

The e200z425n3 and e200z424 cores implement the VLE (variable-length encoding) ISA, providing improved code density. The VLE ISA is further documented in PowerISA 2.06, a separate document.

Note that Data Cache is not implemented on any core on this device.

The e200z425n3 and e200z424 processor cores integrate a pair of integer execution units, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining six read and three write operations per clock. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

A Nexus Class 3+ module is integrated into each e200z425n3 and e200z424 processor core.

A Memory Protection Unit is integrated in each e200z425n3 and e200z424 processor core.

7.3 Features

The following is a list of some of the key features of the e200z425n3 and e200z424 processor cores:

- Dual issue, 32-bit *PowerISA 2.06*-VLE compliant CPU
- In-order execution and retirement
- Precise exception handling
- Branch processing unit
  - Dedicated branch address calculation adder
  - Branch target prefetching using BTB
- Load/store unit
  - 2 cycle load latency
  - Fully pipelined
  - Misaligned access support
- 32-bit General Purpose Register (GPR) file
- Dual AHB 2.v6 64-bit System buses
- Local Instruction (IMEM) and Data Memory (DMEM) with shared AHB 2.v6 64-bit slave interface
  - e200z425n3 core—16 KB IMEM, 32 KB DMEM
  - e200z424 core does not contain either IMEM or DMEM
- Memory Protection Unit (MPU) implementing a 24-entry region descriptor table with support for 6 arbitrary-sized instruction memory regions, 12 arbitrary-sized data memory regions, and 6 additional arbitrary-sized instruction or data memory regions
- 2-Way Set Associative Harvard Instruction Cache (ICache)
  - e200z425n3 core—8 KB ICache
  - e200z424 core does not contain ICache
- Embedded Floating-point APU (EFPU2) supporting single-precision floating-point operations
- Performance Monitor APU supporting execution profiling
- Nexus Class 3-plus Real-time Development Unit

- Power management
  - Low power design with extensive clock gating
  - Power saving mode: wait
  - Dynamic power management of execution units, caches and Local memories
- Testability
  - Synthesizeable, MuxD scan design
  - ABIST/MBIST for arrays
  - Built-in LBIST unit

7.4 Microarchitecture summary

The e200z425n3 and e200z424 processor cores utilize a five stage instruction pipeline with two stages for execution. The Instruction Fetch, Instruction Decode/Register File Read/ EA Calc, Execute 0/ Memory Access0, Execute1/Memory Access1, and Register Writeback stages operate in an overlapped fashion allowing single clock instruction execution for most instructions. The following figure shows a block diagram of the e200z architecture.

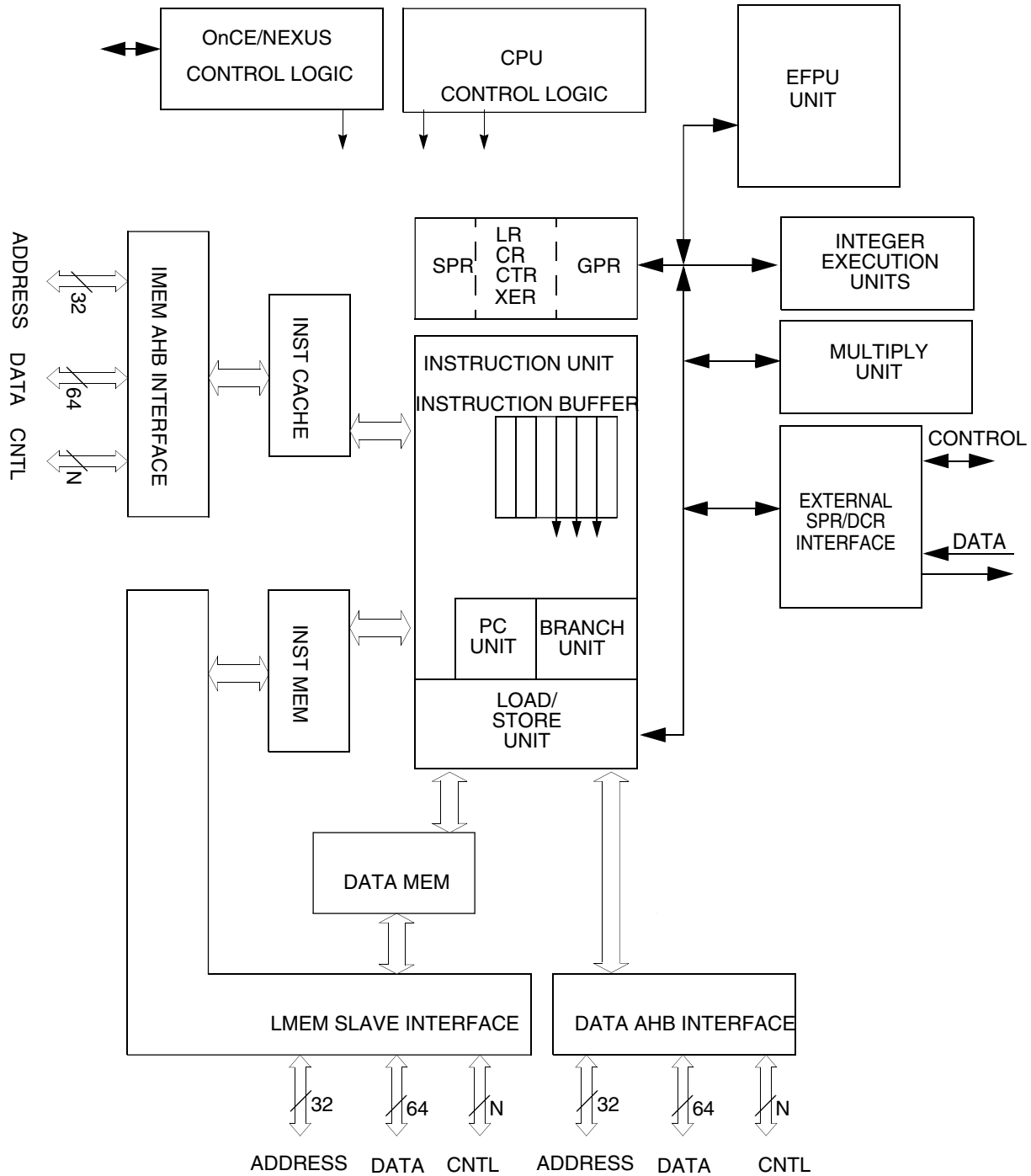


Figure 7-1. e200z block diagram

The integer execution units each consists of a 32-bit Arithmetic Unit (AU), a Logic Unit (LU), a 32-bit Barrel shifter (Shifter), a Mask-Insertion Unit (MIU), a Condition Register manipulation Unit (CRU), a Count-Leading-Zeros unit (CLZ), and result feed-forward hardware. Integer EU1 also supports hardware division and a 32x32 Hardware Multiplier array.

Most arithmetic and logical operations are executed in a single cycle with the exception of multiply, which is implemented with a pipelined hardware array, and the divide instructions. A Count-Leading-Zeros unit operates in a single clock cycle.

The Instruction Unit contains a PC incrementer and dedicated Branch Address adders to minimize delays during change of flow operations. Sequential prefetching is performed to ensure a supply of instructions into the execution pipeline. Branch target prefetching is performed to accelerate taken branches. Prefetched instructions are placed into an instruction buffer.

Branch target addresses are calculated in parallel with branch instruction decode, resulting in an execution time of two clocks for correctly predicted branches. Conditional branches which are not taken execute in a single clock. Branches with successful BTB target prefetching have an effective execution time of one clock if correctly predicted.

Memory load and store operations are provided for byte, halfword, word (32-bit), and doubleword data with automatic zero or sign extension of byte and halfword load data as well as optional byte reversal of data. These instructions can be pipelined to allow effective single cycle throughput. Load and store multiple word instructions allow low overhead context save and restore operations. The load/store unit contains a dedicated effective address adder to allow effective address generation to be optimized.

The Condition Register unit supports the condition register (CR) and condition register operations defined by the Power architecture. The condition register consists of eight 4-bit fields that reflect the results of certain operations, such as move, integer and floating-point compare, arithmetic, and logical instructions, and provide a mechanism for testing and branching.

Vectored and autovectored interrupts are supported by the CPU. Vectored interrupt support is provided to allow multiple interrupt sources to have unique interrupt handlers invoked with no software overhead.

7.4.1 Instruction unit features

The features of the e200z425n3 and e200z424 Instruction units are:

- 64-bit path to cache supports fetching of two 32-bit instructions per clock
- Instruction buffer holds up to eight 32-bit instructions
- Dedicated PC incrementer supporting instruction prefetches
- Branch unit with dedicated branch address adder and branch lookahead logic (BTB) supporting single cycle execution of successfully predicted branches

7.4.2 Integer unit features

The e200z425n3 and e200z424 integer units support single cycle execution of most integer instructions:

- 32-bit AU for arithmetic and comparison operations
- 32-bit LU for logical operations
- 32-bit priority encoder for the count leading zero's function
- 32-bit single cycle barrel shifter for static shifts and rotates
- 32-bit mask unit for data masking and insertion
- Divider logic for signed and unsigned divide in 4-14 clocks with minimized execution timing (EU1 only)
- Pipelined 32x32 hardware multiplier array supports 32x32->32 multiply with 2 clock latency, 1 clock throughput (EU1 only)

7.4.3 Load/Store unit features

The e200z425n3 and e200z424 load/store units support load, store, and the load multiple / store multiple instructions:

- 32-bit effective address adder for data memory address calculations
- Pipelined operation supports throughput of one load or store operation per cycle
- Dedicated 64-bit interface to memory supports saving and restoring of up to two registers per cycle for load multiple and store multiple word instructions and context save/restore instructions

7.4.4 MPU features

The features of the MPU are as follows:

- 24-entry fully-associative Range Table
- Arbitrary range size support from 1 byte to 4 GB
- 8-bit Process Identifier
- Bypass capability for individual access types
- Maskable Address and Process Identifier
- Programmable Memory Attributes
- Entry Invalidation Protection
- Write-Once Option
- Alternate Debug Breakpoint/Watchpoint function

7.4.5 Cache features

The features of the Caches are as follows:

- 2-Way Set Associative Harvard Instruction Cache
  - e200z425n3—8 KB ICache
  - e200z424—No ICache
- Writethrough support
- 8-entry store buffer
- Linefill buffer
- 32-bit address bus plus attributes and control
- Separate uni-directional 64-bit read data bus and 64-bit write data bus
- Support for Way locking
- Support for Write allocation policies
- Support for multi-bit EDC with Correction/Auto-invalidation capability for the ICache
- Software and Hardware Debug access to cache tag, status and data storage via dedicated control registers

7.4.6 Local memory features

The features of the Local Instruction and Data Memories are as follows:

- Local instruction and data memories (IMEM, DMEM) with shared AHB 2.v6 64-bit slave interface
  - e200z425n3—16 KB IMEM, 32 KB DMEM
  - e200z424—No IMEM, No DMEM
- Support for zero wait-state access relative to cache hit timings
- Support for multi-bit (SECDED) ECC with correction/scrubbing capability
- 4-entry ECC R-M-W Store Buffer for DMEM
- 32-bit address bus plus attributes and control
- Separate uni-directional 64-bit read data bus and (for DMEM only) 64-bit write data bus

7.4.7 e200z425n3 system bus features

The features of the e200z425n3 System Bus interface are as follows:

- Independent instruction and data interfaces
- Advanced microcontroller bus architecture (AMBA) AHB2.v6 protocol
- 32-bit address bus, 64-bit data bus, plus attributes and control
- Data interface provides separate uni-directional 64-bit read and write data buses
- Support for HCLK running at a slower rate than CPU clock

7.4.8 e200z424 system features

Checker Core_0s is implemented using an e200z424 core. When enabled, Checker Core_0s is used in Lock Step with Main_Core_0. The e200z424 core can only be enabled or disabled by the user. For this device, the Checker Core_0s has no system bus and cannot independently execute user code. The core is only used to execute the same instruction stream as Main Core 0.

The e200z424 core is identical to the e200z425n3 with these exceptions:

- IMEM memory array is not present
- ICache memory array is not present
- DMEM memory array is not present
- The e200z424 has a pair of system bus interfaces, which are driven/sampled by the delayed lockstep logic.

Chapter 8
Core description

8.1 Overview of the e200z425n3 core

The e200z425n3 is a dual-issue 32-bit *PowerISA 2.06* VLE compliant design with 32-bit general-purpose registers (GPRs). The e200z425n3 core implements the VLE (variable-length encoding) ISA, providing improved code density. The VLE ISA is further documented in *PowerISA 2.06*, a separate document.

An Embedded Floating-point (EFPU2) APU is provided to support real-time single-precision floating-point embedded numerics operations using the general-purpose registers.

A Lightweight Signal Processing Extension (LSP) APU is provided to support real-time SIMD fixed-point embedded numerics operations using the general-purpose registers. All arithmetic instructions that execute in the core operate on data in the general purpose registers (GPRs). The GPRs support register pairing in order to support vector instructions defined by the LSP APU. These instructions operate on a vector pair of 16-bit or 32-bit data types, and deliver 16-bit vector and 32-bit scalar results. A wide variety of data manipulation, multiply, multiply/accumulate, and dot product instructions along with specialized memory access instructions allow a sustained throughput of up to two 16x16-bit multiply/accumulate operations per cycle.

The e200z425n3 processor integrates a pair of integer execution units, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining six read and three write operations per clock cycle. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

e200z425n3 contains an 8 KB Instruction Cache, as well as a Nexus Class 3+ real-time debug module with support for program and data trace features as well as extensive trace controls. Data cache is not supported on any of the cores on this device.

A Memory Protection Unit that protects various instruction and data memory areas is also included.

For details on the instruction sets supported by this core, refer to the following documents:

- Basic 32-bit instruction set: EREF_RM.pdf – EREF 2.0: A Programmer's Reference Manual for Freescale Power Architecture Processors.
- VLE instruction set: VLEPEM.pdf – Variable-Length Encoding (VLE) Programming Environments Manual: A Supplement to the EREF
- LSP instruction set: LSPAPURM.pdf – Lightweight Signal Processing APU (LSP APU) Reference Manual
- Decorated storage instructions: This Reference Manual

8.2 Register model

This section describes the registers implemented in the e200z425n3 core. The *PowerISA 2.06* architecture defines register-to-register operations for all computational instructions.

e200z425n3 extends usage of the general purpose registers to support EFPU /LSP APU operations on the 32-bit GPR registers.

Figure 8-1 and Figure 8-2 show the complete e200z425n3 register set, all of which are accessible in supervisor mode. Figure 8-3 and Figure 8-4 show the subset of registers accessible in user mode. The number to the right of the special-purpose registers (SPRs) is the decimal number used in the instruction syntax to access the register (for example, the integer exception register (XER) is SPR 1).

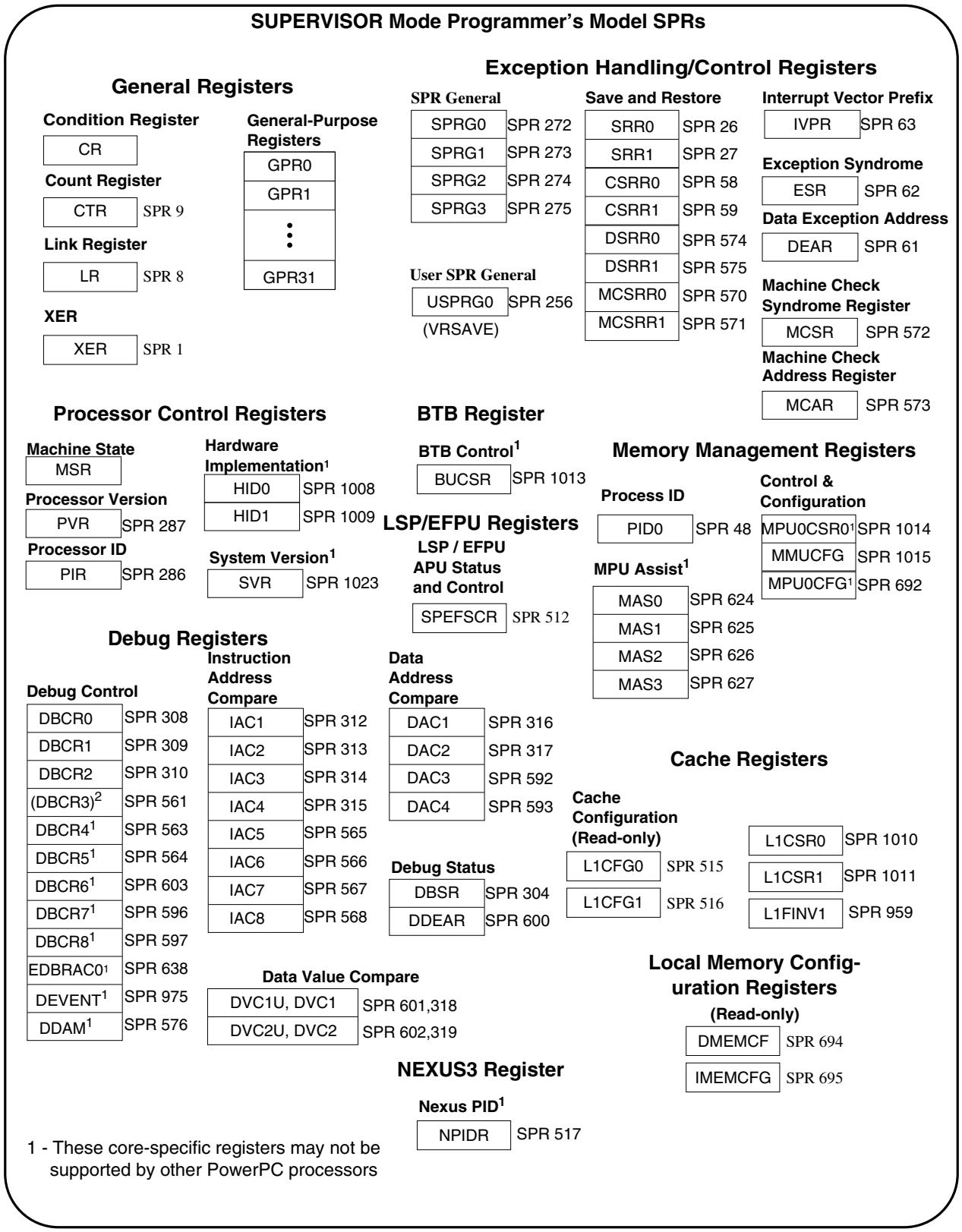


Figure 8-1. e200z425n3 Supervisor Mode Programmer's Model SPRs

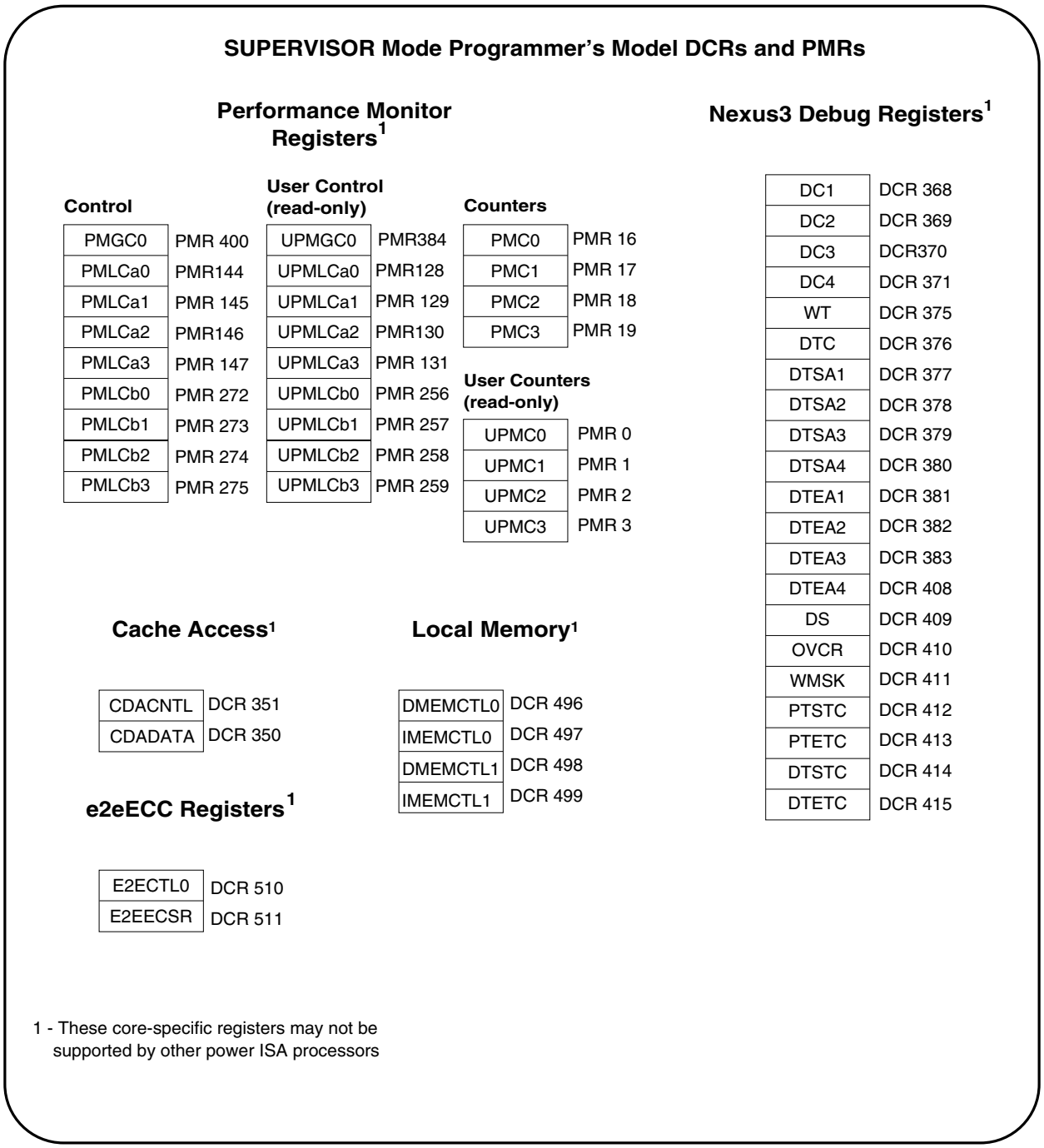


Figure 8-2. e200z425n3 Supervisor Mode Programmer's Model DCRs and PMRs

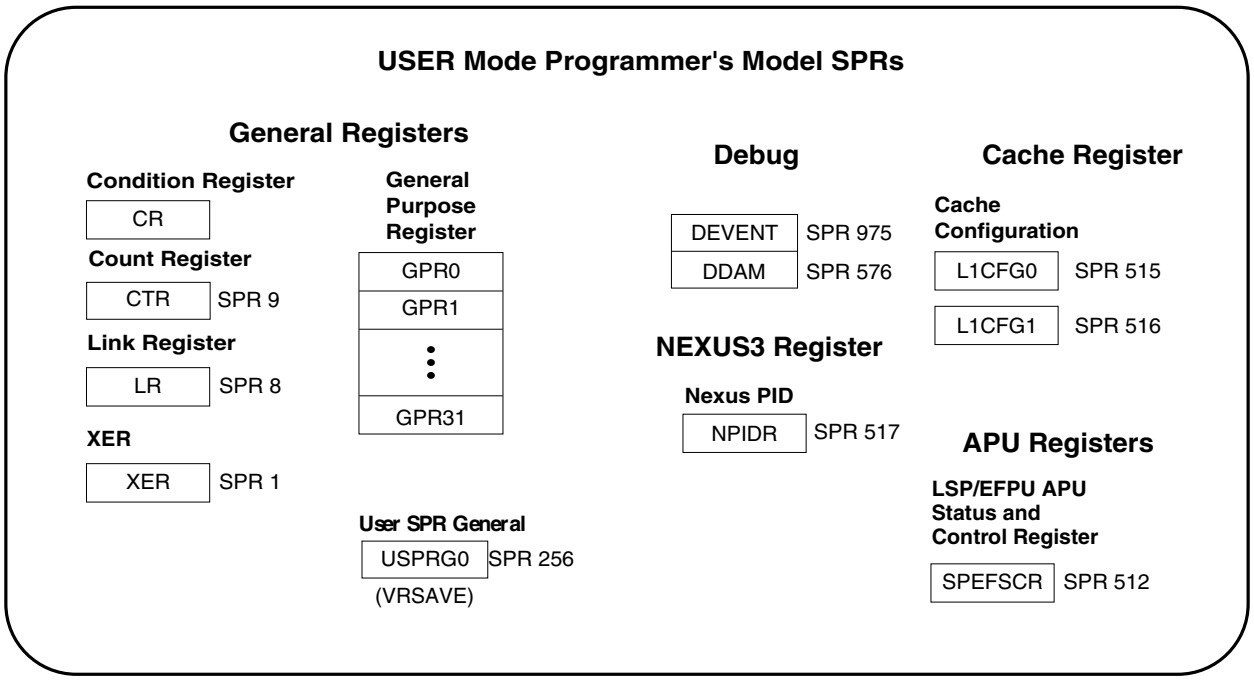


Figure 8-3. e200z425n3 User Mode Programmer's Model SPRs

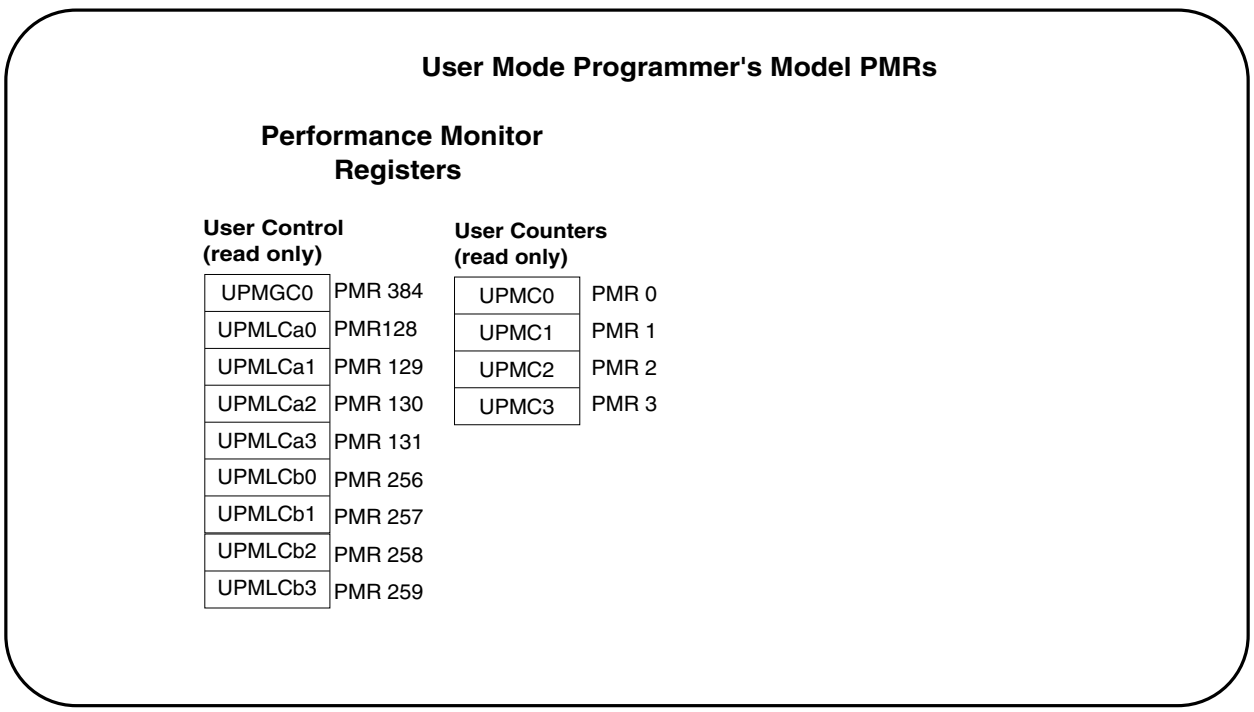


Figure 8-4. e200z425n3 User Mode Programmer's Model PMRs

8.3 Dual-issue operation

The instruction issue unit attempts to issue a pair of instructions to the execution units each cycle. Source operands for each of the instructions are provided from the GPRs or from the operand feed-forward muxes. Data or resource hazards may create stall conditions that cause instruction issue to be stalled for one or more cycles until the hazard is eliminated.

The execution units write the result of a finished instruction onto the proper result bus and into the destination registers. The writeback logic retires an instruction when the instruction has finished execution. Up to three results can be simultaneously written.

Two execution units are provided to allow dual issue of most instructions. Only a single load/store unit is provided. Only a single integer multiply and divide unit is provided, thus a pair of multiply or divide instructions cannot issue simultaneously. In addition, the divide unit is blocking.

The following table shows the concurrent instruction issue capabilities. Note that data dependencies between instructions will generally preclude dual-issue of those instructions.

Table 8-1. Concurrent instruction issue capabilities for e200z425n3

Class of instruction	Branch	Load/ Store	Scalar integer	Scalar float	Vector integer non-MAC	Vector integer MAC	Special
Branch	—	yes	yes	yes	yes	yes	—
Load/Store	yes	—	yes	yes	yes	yes	—
Scalar integer	yes	yes	yes[1]	yes	yes	yes[2, 2]	—
Scalar float	yes	yes	yes	—	yes	yes	—
Vector integer non-MAC	yes	yes	yes	yes	yes	yes	—
Vector integer MAC	yes	yes	yes[2]	yes	yes	—	—
Special	—	—	—	—	—	—	—

1. Excludes multiply or divide class instructions occurring in both issue slots.
2. Excludes vector MAC/multiply class instructions occurring with scalar multiply, or divide class instructions.

8.4 Instruction timing

Instruction timing in number of processor clock cycles for various instruction classes is shown in Table 8-2. Pipelined instructions are shown with cycles of total latency and throughput cycles. Divide instructions are not pipelined and block other instructions from executing during execution. Timing for EFPU instructions is detailed in Table 8-5. Timing for LSP instructions is detailed in:

- LSP simple instruction timing—Table 8-6
- LSP complex instruction timing—Table 8-7
- LSP shift/rotate instruction timing—Table 8-8
- LSP vector compare instruction timing—Table 8-9
- LSP vector select instruction timing—Table 8-10
- LSP vector data arrangement instruction timing—Table 8-11
- LSP multiply and multiply/accumulate instruction timing—Table 8-12
- LSP dot product instruction timing—Table 8-13
- LSP miscellaneous vector instruction timing—Table 8-14
- LSP load and store instruction timing—Table 8-15

Load/store multiple instruction cycles are represented as a fixed number of cycles plus a variable number of cycles where *n* is the number of words accessed by the instruction. In addition, cycle times marked with & require variable number of additional cycles due to serialization.

Table 8-2. Instruction class cycle counts

Class of Instructions	Latency	Description	Special notes
integer: **add**, **sub**, **shift**, **rotate**, **logical**, **cntlzw**	1	1	—
integer: **compare**			—
Branch	3/2/1	3/2/1	Correct branch lookahead allows single-cycle execution. Worst-case mispredicted branch is 3 cycles.
multiply	2	1	—
divide	4–14	4–14	Data-dependent timing
CR logical	1	1	—
loads (non-multiple)	2	1	—
load multiple	2 + *n*/2 (max)	1 + *n*/2 (max)	Actual timing depends on *n* and address alignment.
stores (non-multiple)	2	1	—
store multiple	2 + *n*/2 (max)	2 + *n*/2 (max)	Actual timing depends on *n* and address alignment.
mtmsr, **wrtee**, **wrteei**	3&	3&	—

Table continues on the next page...

Table 8-2. Instruction class cycle counts (continued)

Class of Instructions	Latency	Description	Special notes
mcrf	1	1	—
mfspr, **mtspr**	4&	4&	Applies to Debug SPRs, optional unit SPRs
mfspr, **mfmsr**	1	1	Applies to certain limited internal, non Debug SPRs
mfcr, **mtcr**	1	1	—
se_rfi, **se_rfci**, **se_rfdi**, **se_rfmci**	3	—	—
se_sc, **e_sc**	4	—	—
e_tw	4	—	Trap taken timing

Detailed timing for each instruction mnemonic along with serialization requirements is shown in Table 8-3 and Table 8-4.

Table 8-3. Instruction timing by mnemonic—16-bit instructions

Mnemonic	Latency	Serialization
se_add	1	—
se_addi	1	—
se_and[.]	1	—
se_andc	1	—
se_andi	1	—
se_bc	3/2/1	—
se_bclri	1	—
se_bctr	3/2	—
se_bctrl	3/2	—
se_bgeni	1	—
se_bl	3/2/1	—
se_blr	3/2	—
se_blrl	3/2	—
se_bmaski	1	—
se_b	3/2/1	—
se_bseti	1	—
se_btsti	1	—
se_cmp	1	—
se_cmph	1	—
se_cmphl	1	—
se_cmpi	1	—
se_cmpl	1	—
se_cmpl	1	—

Table continues on the next page...

Table 8-3. Instruction timing by mnemonic—16-bit instructions (continued)

Mnemonic	Latency	Serialization
se_dnh	1+	—
se_dni	1+	—
se_extsb	1	—
se_extsh	1	—
se_extzb	1	—
se_extzh	1	—
se_illegal	4	—
se_isync	6[1]	Refetch
se_lbz	2	—
se_lhz	2[2]	—
se_li	1	—
se_lwz	2[2]	—
se_mfar	1	—
se_mfctr	1	—
se_mflr	1	—
se_mr	1	—
se_mtar	1	—
se_mtctr	1	—
se_mtlr	1	—
se_mullw	2	—
se_neg	1	—
se_not	1	—
se_or	1	—
se_rfci	3	Refetch
se_rfdi	3	Refetch
se_rfi	3	Refetch
se_rfmci	3	Refetch
se_sc	4	Refetch
se_slw	1	—
se_slwi	1	—
se_sraw	1	—
se_srawi	1	—
se_srw	1	—
se_srwi	1	—
se_stb	2	—
se_sth	2[2]	—
se_stw	2[2]	—
se_sub	1	—
se_subf	1	—

Table continues on the next page...

Table 8-3. Instruction timing by mnemonic—16-bit instructions (continued)

Mnemonic	Latency	Serialization
se_subi[.]	1	—

1. Plus additional synchronization time.
2. Aligned

Table 8-4. Instruction timing by mnemonic—32-bit instructions

Mnemonic	Latency	Serialization
add[.]	1	—
e_add16i	1	—
e_add2i.	1	—
e_add2is	1	—
addc[.]	1	—
addco[.]>	1	—
adde[.]	1	—
addeo[.]	1	—
e_addi[.]	1	—
e_addic[.]	1	—
addme[.]	1	—
addmeo[.]	1	—
addo[.]	1	—
addze[.]	1	—
addzeo[.]	1	—
and[.]	1	—
e_and2i.	1	—
e_and2is.	1	—
andc[.]	1	—
e_andi[.]	1	—
e_b	3/2/1	—
e_bc	3/2/1	—
e_bcl	3/2/1	—
e_bl	3/2/1	—
cmp	1	—
e_cmp16i	1	—
e_cmph	1	—
e_cmph16i	1	—
e_cmphl	1	—
e_cmphl16i	1	—
e_cmpi	1	—
cmpl	1	—
e_cmpl16i	1	—

Table continues on the next page...

Table 8-4. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
e_cmpli	1	—
cntlzw[.]	1	—
e_crand	1	—
e_crandc	1	—
e_creqv	1	—
e_crnand	1	—
e_crnor	1	—
e_cror	1	—
e_crorc	1	—
e_crxor	1	—
dcba	1	—
dcbf	1	—
dcbi	1	—
dcbst	1	—
dcbt	1	—
dcbtst	1	—
dcbz	— (alignment error)	—
divw[.]	4–14[1]	—
divwo[.]	4–14[1]	—
divwu[.]	4–14[1]	—
divwuo[.]	4–14[1]	—
e_dnh	1+	—
e_dni	1+	—
eqv[.]	1	—
extsb[.]	1	—
extsh[.]	1	—
icbi	1	—
icbt	1	—
isel	1	—
lbarx	2	—
e_lbz	2	—
e_lbzu	2	—
lbzux	2	—
lbzx	2	—
e_lha	2[2]	—
lharx	2	—
e_lhau	2[2]	—
lhaux	2[2]	—
lhax	2[2]	—

Table continues on the next page...

Table 8-4. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
lhbrx	2[2]	—
e_lhz	2[2]	—
e_lhzu	2[2]	—
lhzux	2[2]	—
lhzx	2[2]	—
e_li	1	—
e_lis	1	—
e_lmw	2 +(*n*/2)	—
lwarx	2	—
lwbrx	2[2]	—
e_lwz	2[2]	—
e_lwzu	2[2]	—
lwzux	2[2]	—
lwzx	2[2]	—
mbar	1[3]	Pseudo-dispatch
e_mcrf	1	—
mcrxr	1	Completion
mfcr	1	—
mfdcr	1	Completion
mfmsr	1	—
mfspr (except DEBUG, CACHE, LMEM < MPU)	1	None
mfspr (DEBUG, CACHE, LMEM < MPU)	4[3]	Completion
mpure	4[3]	Completion
mpusync	1	Completion
mpuwe	4[3]	Completion
msync	1[3]	Completion
mtcrf	2	—
mtmsr	3[3]	Completion
mtspr (except DEBUG, **msr**, hid0/1)	i	None
mtspr (DEBUG, CACHE, LMEM < MPU)	4[3]	Completion
mulhw[.]	2	—
mulhwu[.]	2	—
e_mull2i	2	—
e_mulli	2	—
mullw[.]	2	—
mullwo[.]	2	—
nand[.]	1	—

Table continues on the next page...

Table 8-4. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
neg[.]	1	—
nego[.]	1	—
nor[.]	1	—
or[.]	1	—
e_or2i	1	—
e_or2is	1	—
orc[.]	1	—
e_ori[.]	1	—
e_rlw[.]	1	—
e_rlwi[.]	1	—
e_rlwimi	1	—
e_rlwinm	1	—
e_sc	1	Refetch
slw[.]	1	—
e_slwi[.]	1	—
sraw[.]	1	—
srawi[.]	1	—
srw[.]	1	—
e_srwi[.]	1	—
e_stb	2	—
stbcx.	2	—
e_stbu	2	—
stbux	2	—
stbx	2	—
e_sth	2[2]	—
sthbrx	2[2]	—
sthcx.	2	—
e_sthu	2[2]	—
sthux	2[2]	—
sthx	2[2]	—
e_stmw	2 + (*n*/2)	—
e_stw	2[2]	—
stwbrx	2[2]	—
stwcx.	2	—
e_stwu	2[2]	—
stwux	2[2]	—
stwx	2[2]	—
subf[.]	1	—
subfc[.]	1	—

Table continues on the next page...

Table 8-4. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
subfco[.]	1	—
subfe[.]	1	—
subfeo[.]	1	—
e_subfic[.]	1	—
subfme[.]	1	—
subfmeo[.]	1	—
subfo[.]	1	—
subfze[.]	1	—
subfzeo[.]	1	—
tw	4	—
wait	—	Completion
wrtee	3	Completion
wrteei	3	Completion
xor[.]	1	—
e_xori[.]	1	—

1. With early-out capability, timing is data-dependent.
2. Aligned.
3. Plus additional synchronization time.

Instruction timing for EFPU single-precision scalar floating-point instructions is shown in Table 8-5. The table is sorted by opcode.

Table 8-5. EFPU single-precision scalar floating-point instruction timing

Instruction	Latency	Throughput	Comments
efsabs	2	1	—
efsadd	2	1	—
efscfh	2	1	—
efscfsf	2	1	—
efscfsi	2	1	—
efscfuf	2	1	—
efscfui	2	1	—
efscmpeq	2	1	—
efscmpgt	2	1	—
efscmplt	2	1	—
efscth	2	1	—
efsctsf	2	1	—
efsctsi	2	1	—
efsctsiz	2	1	—
efsctuf	2	1	—
efsctui	2	1	—

Table continues on the next page...

Table 8-5. EFPU single-precision scalar floating-point instruction timing (continued)

Instruction	Latency	Throughput	Comments
efsctuiz	2	1	—
efsdiv	13	13	Blocking, no execution overlap with next instruction
efsmadd	2	1[1]	Destination also used as source
efsmsub	2	1[1]	Destination also used as source
efsmax	2	1	—
efsmin	2	1	—
efsmul	2	1	—
efsnabs	2	1	—
efsneg	2	1	—
efsnmadd	2	1[1]	Destination also used as source
efsnmsub	2	1[1]	Destination also used as source
efssqrt	11	11	Blocking, no overlap with next instruction
efssub	2	1	—
efststeq	2	1	—
efststgt	2	1	—
efststlt	2	1	—

1. Destination register is also a source register, so for full throughput, back-to-back operations must use a different destination register.

Instruction timing in number of processor clock cycles for LSP instructions are shown in the following tables. Pipelined instructions are shown with cycles of total latency and throughput cycles. Divide instructions are not pipelined and block other instructions from executing during divide execution.

Instruction timing for LSP simple instructions is shown in Table 8-6.

Table 8-6. LSP simple instruction timing

Basic operation	Variants	Latency	Throughput
Absolute value	**zvabsh**, **zabsw**	1	1
	zvabshs, **zabsws**	1	1
Add	**zaddd**, **zvaddh**, **zvaddw**	1	1
	zadddss, **zvaddhss**, **zaddwss**, **zvaddwss**, **zadddus**, **zvaddhus**, **zaddwus**	1	1
	zvaddhx, **zvaddhxss**	1	1
	zaddhe[s,u]w, **zaddho[s,u]w**	1	1
	zvaddih	1	1
	zaddwgsf	1	1
	zaddwgsi, **zaddwgui**	1	1

Table continues on the next page...

Table 8-6. LSP simple instruction timing (continued)

Basic operation	Variants	Latency	Throughput
AddSubf	**zvaddsubfh**, **zvaddsubfhss**, **zvaddsubfw**, **zvaddsubfwss**	1	1
	zvaddsubfhx, **zvaddsubfhxss**	1	1
Count Leading	**zvcntlsh**, **zvcntlzh**, **zcntlsw**	1	1
Negate	**zvnegh**	1	1
	zvneghs, **znegws**	1	1
	zvnegho, **zvneghos**	1	1
Round	**zrndwh**, **zrndwhss**	1	1
Saturate	**zsatsdsw**, **zsatsduw**	1	1
	zsatuduw	1	1
	zvsatshuh	1	1
	zvsatuhsh	1	1
	zsatswsh, **zsatswuh**	1	1
	zsatswuw	1	1
	zsatuwsh, **zsatuwuh**	1	1
	zsatuwsw	1	1
Subf	**zsubfd**, **zvsubfh**, **zvsubfw**	1	1
	zsubfdss, **zvsubfhss**, **zsubfwss**, **zvsubfwss**, **zsubfdus**, **zvsubfhus**, **zsubfwus**, **zvsubfwus**	1	1
	zvsubfhx, **zvsubfhxss**	1	1
	zsubfhe[s,u]w, **zsubfho[s,u]w**	1	1
	zvsubifh	1	1
	zsubfwgsf	1	1
	zsubfwgsi, **zsubfwgui**	1	1
SubfAdd	**zvsubfaddh**, **zvsubfaddhss**, **zvsubfaddw**, **zvsubfaddwss**	1	1
	zvsubfaddhx, **zvsubfaddhxss**	1	1

Instruction timing for LSP complex instructions is shown in Table 8-7. For the divide instruction, the number of stall cycles is (latency) for following instructions.

Table 8-7. LSP complex instruction timing

Operation	Instruction	Latency	Throughput
Divide	**zdivwsf**	4–14[1]	4–14[1]

1. Timing is data-dependent.

Instruction timing for LSP shift/rotate instructions is shown in Table 8-8.

Table 8-8. LSP shift/rotate instruction timing

Basic operation	Variants	Latency	Throughput
Logical Shift Left	**zvslh**, **zvslhi**	1	1
Logical Shift Right	**zvsrhu**, **zvsrhiu**	1	1
Rotate Left	**zvrlh**, **zvrlhi**	1	1
Signed Shift Left and Saturate	**zslwss**, **zslwiss**, **zvslhss**, **zvslhiss**	1	1
Unsigned Shift Left and Saturate	**zslwus**, **zslwius**, **zvslhus**, **zvslhius**	1	1
Arithmetic Shift Right	**zvsrhs**, **zvsrhis**	1	1

Instruction timing for LSP vector compare instructions is shown in Table 8-9.

Table 8-9. LSP vector compare instruction timing

Basic comparison operation	Instruction	Latency	Throughput
=	**zvcmpeqh**>	1	1
>	**zvcmpgths**, **zvcmpgthu**	1	1
<	**zvcmplthu**, **zvcmplthu**	1	1

Instruction timing for LSP vector select instructions is shown in Table 8-10.

Table 8-10. LSP vector select instruction timing

Operation	Instruction	Latency	Throughput
Select	**zvselh**	1	1

Instruction timing for LSP vector data arrangement instructions is shown in Table 8-11.

Table 8-11. LSP vector data arrangement instruction timing

Operation	Instruction	Latency	Throughput
Merge	**zvmergehih**	1	1
	zvmergehiloh	1	1
	zvmergeloh	1	1
	zvmergelohih	1	1
Pack	**zvpkswshfrs**	1	1
	zvpkswshs, **zvpkswuhs**, **zvpkuwuhs**	1	1
	zpkswgswfrs	1	1
	zvpkshgwshfrs	1	1
Splat	**zvsplatfih**	1	1
	zvsplatih	1	1
Unpack	**zunpkwgsf**	1	1
	zvunpkhgwsf	1	1
	zvunpkhsf, **zvunpkhsi**, **zvunpkhui**	1	1

Instruction timing for LSP multiply and multiply/accumulate instructions is shown in Table 8-12.

Table 8-12. LSP multiply and multiply/accumulate instruction timing

Instruction	Latency	Throughput
all **z[v]m{h,w}** instructions	—	1

Instruction timing for LSP dot product instructions is shown in Table 8-13.

Table 8-13. LSP dot product instruction timing

Instruction	Latency	Throughput
all **z[v]dotp** instructions	—	1

Instruction timing for LSP miscellaneous instructions is shown in Table 8-14.

Table 8-14. LSP miscellaneous vector instruction timing

Operation	Instruction	Latency	Throughput
Circular increment	**zcircinc**	1	1
Bit reversed increment	**zbrminc**	1	1

Instruction timing for LSP load and store instructions instructions is shown in Table 8-15.

Table 8-15. LSP load and store instruction timing

Instruction	Latency	Throughput
all **zv** loads	—	1
all **zv** stores	—	1

8.5 Reservation instructions and cache interactions

If the CPU supports reservation instruction functionality, the CPU treats reservation instruction (**lbarx**, **lharx**, **lwarx**, **stbcx**, **sthcx**, and **stwcx**) accesses as though they were cache-inhibited, and forces a cache miss. A reservation access is always issued to the bus. This is done to allow external reservation logic to be built that properly signals a reservation failure. The bus access will be treated as a single-beat transfer.

8.6 Signal Processing Extension/Embedded Floating-point Status and Control Register (SPEFSCR)

Status and control for embedded floating-point operations use the SPEFSCR. This register is also used by the LSP APU. The SPEFSCR is implemented as special-purpose register (SPR) number 512 and is read and written by the mfspr and mtspr instructions. The SPEFSCR is shown in the following figure.

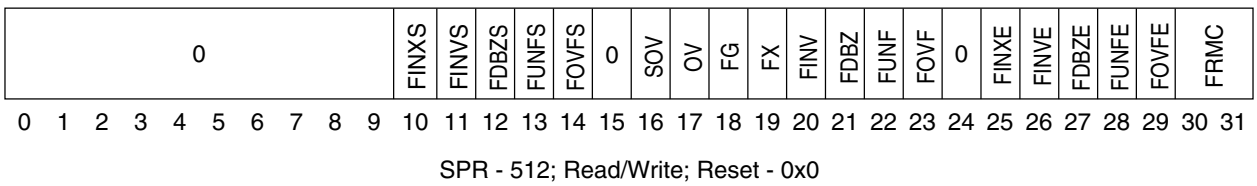


Figure 8-5. LSP/EFPU Status and Control Register (SPEFSCR)

The SPEFSCR bits are defined in the following table.

Table 8-16. LSP/EFPU Status and Control Register field descriptions

Bits	Name	Description
0:9 (32:41)	—	Reserved
10 (42)	FINXS	Embedded Floating-point Inexact Sticky Flag The FINXS bit is set to 1 whenever the execution of a floating-point instruction delivers an inexact result and no Floating-point Data exception is taken, or if the result of a Floating-point instruction results in overflow (FOVF = 1), but Floating-point Overflow exceptions are disabled (FOVFE = 0), or if the result of a Floating-point instruction results in underflow (FUNF = 1), but Floating-point Underflow exceptions are disabled (FUNFE = 0), and no Floating-point Data exception occurs. The FINXS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
11 (43)	FINVS	Embedded Floating-point Invalid Operation Sticky Flag The FINVS bit is set to a 1 when a floating-point instruction sets the FINV bit to 1. The FINVS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
12 (44)	FDBZS	Embedded Floating-point Divide by Zero Sticky Flag The FDBZS bit is set to 1 when a floating-point divide instruction sets the FDBZ bit to 1. The FDBZS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
13 (45)	FUNFS	Embedded Floating-point Underflow Sticky Flag The FUNFS bit is set to 1 when a floating-point instruction sets the FUNF bit to 1. The FUNFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
14 (46)	FOVFS	Embedded Floating-point Overflow Sticky Flag The FOVFS bit is set to 1 when a floating-point instruction sets the FOVF bit to 1. The FOVFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
15 (47)	—	Reserved

Table continues on the next page...

Table 8-16. LSP/EFPU Status and Control Register field descriptions (continued)

Bits	Name	Description
16 (48)	SOV	Summary integer overflow Defined by LSP APU.
17 (49)	OV	Integer overflow Defined by LSP APU.
18 (50)	FG	Embedded Floating-point Guard bit FG is supplied for use by the Floating-point Round exception handler. FG is zeroed if a Floating-point Data Exception occurs.
19 (51)	FX	Embedded Floating-point Sticky bit FX is supplied for use by the Floating-point Round exception handler. FX is zeroed if a Floating-point Data Exception occurs.
20 (52)	FINV	Embedded Floating-point Invalid Operation / Input error In mode 0, the FINV bit is set to 1 if the A or B operand of a floating-point instruction is Infinity, NaN, or Denorm, or if the operation is a divide and the dividend and divisor are both 0. In mode 1, the FINV bit is set on an IEEE754 invalid operation (IEEE754-1985 sec7.1).
21 (53)	FDBZ	Embedded Floating-point Divide by Zero The FDBZ bit is set to 1 when a floating-point divide instruction executed with divisor of 0, and the l dividend is a finite non-zero number.
22 (54)	FUNF	Embedded Floating-point Underflow The FUNF bit is set to 1 when the execution of a floating-point instruction results in an underflow.
23 (55)	FOVF	Embedded Floating-point Overflow The FOVF bit is set to 1 when the execution of a floating-point instruction results in an overflow.
24 (56)	—	Reserved
25 (57)	FINXE	Embedded Floating-point Inexact Exception Enable If the exception is enabled, a Floating-point Round exception is taken if the result of a Floating-point instruction does not result in overflow or underflow, and the result is inexact (FG l FX = 1), or if the result of a Floating-point instruction does result in overflow (FOVF = 1) but Floating-point Overflow exceptions are disabled (FOVFE = 0), or if the result of a Floating-point instruction results in underflow (FUNF = 1 or FUNFH = 1) but Floating-point Underflow exceptions are disabled (FUNFE = 0), and no Floating-point Data exception occurs. 0 Exception disabled 1 Exception enabled
26 (58)	FINVE	Embedded Floating-point Invalid Operation / Input Error Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FINV bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled

Table continues on the next page...

Table 8-16. LSP/EFPU Status and Control Register field descriptions (continued)

Bits	Name	Description
27 (59)	FDBZE	Embedded Floating-point Divide by Zero Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FDBZ bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
28 (60)	FUNFE	Embedded Floating-point Underflow Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FUNF bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
29 (61)	FOVFE	Embedded Floating-point Overflow Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FOVF bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
30:31 (62:63)	FRMC	Embedded Floating-point Rounding Mode Control 00 Round to Nearest 01 Round toward Zero 10 Round toward +Infinity 11 Round toward -Infinity

8.7 Cache

This section describes the cache registers, cache control instructions, and various cache operations.

8.7.1 Cache overview

The e200z425n3 processor supports an 8 KB 2-way set-associative instruction cache with a 32-byte line size. The cache improves system performance by providing low-latency data to the e200z425n3 instruction pipeline, which decouples processor performance from system memory performance.

The e200z425n3 processor also contains an 8-entry store buffer to decouple store completion to the processor from store completion on the data interface to further improve system performance.

Instruction addresses from the processor are used to index the cache arrays. If the access address matches a valid cache tag entry, the access hits in the cache.

8.7.2 Cache organization

The e200z425 cache is organized as two ways of 64 sets with each line containing 32 bytes (four doublewords) of storage. Figure 8-6 illustrates the cache organization along with the cache line format.

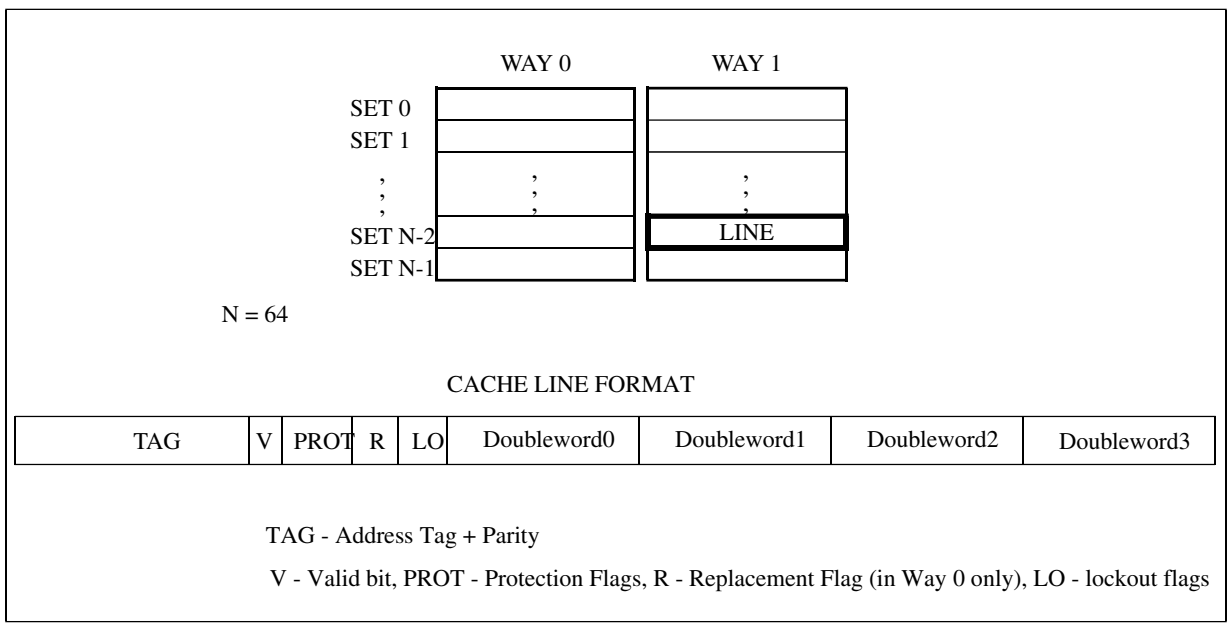


Figure 8-6. Cache organization and line format

Address bits A[20:26] provide an index to select a set. Ways are selected according to the rules of set association.

Each line consists of a address tag, protection bit(s), status bits, and four doublewords of data. Address bits A[27:29] select the word within the line. The address tag consists of the upper address of the access value used to fill the cache line. The protection field (PROT) contains a Supervisor Protect bit (SP). The protection bits are used to enforce access protection provided from an external SoC-level memory protection unit (MPU). Lockout bits (LO) are provided to support the cache line lockout functionality when errors are encountered.

8.7.3 L1 Cache Control and Status Register 0 (L1CSR0)

L1CSR0 implements the WID control field for instruction cache way disabling. The other fields are reserved. The L1CSR0 register is accessed using a **mfspr** or **mtspr** instruction. The L1CSR0 register is shown in the following figure.

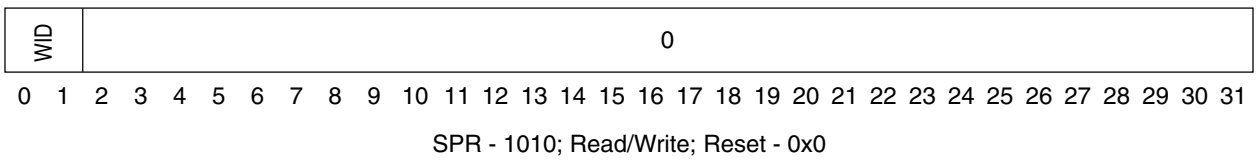


Figure 8-7. L1 Cache Control and Status Register 0 (L1CSR0)

The L1CSR0 fields are described in the following table.

Table 8-17. L1CSR0 field descriptions

Bits	Name	Description
0:1	WID	Way Instruction Disable Bit 0 corresponds to way 0. Bit 1 corresponds to way 1. The WID bits may be used for locking ways of the instruction cache, and also affect the replacement policy of the instruction cache. 0 The corresponding way in the instruction cache is available for replacement by instruction miss line fills. 1 The corresponding way instruction cache is not available for replacement by instruction miss line fills.
2:31	—	Reserved[1]

1. These bits are not implemented and should be written with zero for future compatibility.

8.7.4 L1 Cache Control and Status Register 1 (L1CSR1)

The L1 Cache Control and Status Register 1 (L1CSR1) is a 32-bit register used for general control of the instruction cache. The L1CSR1 register is accessed using a **mfspr** or **mtspr** instruction. The L1CSR1 register is shown in the following figure.

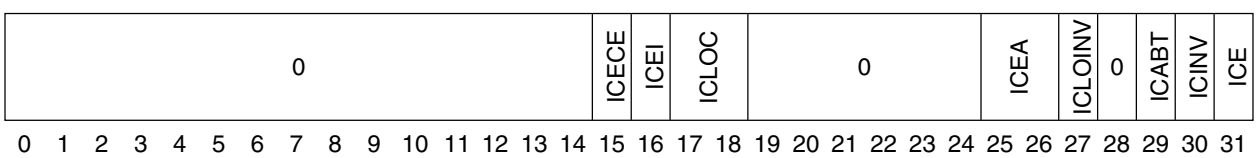


Figure 8-8. L1 Cache Control and Status Register 1 (L1CSR1)

The L1CSR1 fields are described in the following table.

Table 8-18. L1CSR1 field descriptions

Bits	Name	Description
0:14	—	Reserved[1, 1]
15	ICECE	Instruction Cache Error Checking Enable 0 Error Checking is disabled 1 Error Checking is enabled
16	ICEI	Instruction Cache Error Injection Enable ICEI will cause injection of errors regardless of the setting of ICECE, although reporting of errors will be masked when ICECE = 0. 0 Cache Error Injection is disabled 1 A double-bit error will be injected into each doubleword written into the cache by inverting the two uppermost parity check bits.
17:18	ICLOC	Instruction Cache Lockout Control **Note:** For the **icbi** instruction, no lockout operation is performed, regardless of the occurrence of EDC/ECC error conditions, and errors are handled in the same manner as when lockout is disabled. See ECC/EDC Error Handling for Cache Control Operations and Instructions. **Note:** When operating in MC mode (ICEA = 00), detected errors on cache-inhibited accesses will not cause an instruction cache line to be locked out, instead the cache line contents are ignored. 00 Cache line lockout is disabled (and array LO indicators are ignored). 01 Cache line lockout is enabled. Cache lines with any tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. 10 Cache line lockout is enabled. Cache lines with any tag or data errors will have the corresponding lockout indicators set. A Machine check will still be generated for an error if the operation would have normally generated one. 11 Cache line lockout is enabled. Cache lines with uncorrectable tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. If a correctable tag error occurs, and is re-detected on recycling the access after a correction cycle for the tag is performed, the line is locked out regardless of detecting a single-bit or multi-bit error.
19:24	—	Reserved[1]
25:26	ICEA	Instruction Cache Error Action 00 Error Detection causes Machine Check exception. 01 Error Detection causes Correction/Auto-invalidation. No machine check is generated for most cases. Correction is performed for single-bit tag errors, and lines with multi-bit tag errors are invalidated. Correction is performed for single or multi-bit data errors on cache hits by reloading of the line. 10 Reserved 11 Reserved

Table continues on the next page...

Table 8-18. L1CSR1 field descriptions (continued)

Bits	Name	Description
27	ICLOINV	Instruction Cache Lockout Indicator Invalidate When written to a 1 in conjunction with writing ICINV to a 1, a cache lockout indicator invalidation operation is initiated by hardware, and the LO bits are cleared, removing all line lockout status. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache lockout bit invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the instruction cache enable (ICE) value. When ICINV = 0: Reserved, do not set to 1 When ICINV = 1: 0 No cache lockout bit invalidate 1 Cache lockout indicator invalidation operation
28	—	Reserved[1]
29	ICABT	Instruction Cache Operation Aborted Indicates a Cache Invalidate operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location.
30	ICINV	Instruction Cache Invalidate When written to a 1, a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 130 cycles to complete. Invalidation occurs regardless of the enable (ICE) value. 0 No cache invalidate 1 Cache invalidation operation
31	ICE	Instruction Cache Enable When disabled, cache lookups are not performed for instruction accesses. Other L1CSR1 cache control operations are still available and are not affected by ICE. 0 Cache is disabled 1 Cache is enabled

1. These bits are not implemented and should be written with zero for future compatibility.

8.7.5 L1 Cache Configuration Register 0 (L1CFG0)

The L1 Cache Configuration Register 0 (L1CFG0) is a 32-bit read-only register. L1CFG0 provides information about the configuration of the e200z425n3 L1 data cache design. The contents of the L1CFG0 register can be read using a **mfspr** instruction. The L1CFG0 register is shown in the following figure.

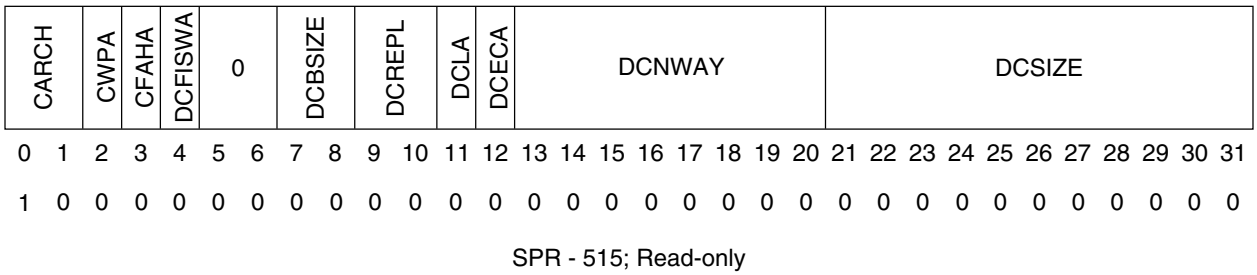


Figure 8-9. L1 Cache Configuration Register 0 (L1CFG0)

Table 8-19. L1CFG0 field descriptions

Bits	Name	Description
0:1	CARCH	Cache Architecture 00 The cache architecture is Harvard 10 An I-Cache but no D-Cache is present
2	CWPA	Cache Way Partitioning Available 0 The caches do not support partitioning of way availability for I/D accesses
3	DCFAHA	Data Cache Flush All by Hardware Available 0 The data cache does not support Flush All in Hardware
4	DCFISWA	Data Cache Flush/Invalidate by Set and Way Available 0 The data cache does not support invalidation by Set and Way via L1FINV0
5:6	—	Reserved - read as zeros
7:8	DCBSIZE	Data Cache Block Size 00 The data cache implements a block size of 32 bytes
9:10	DCREPL	Data Cache Replacement Policy 00 The data cache does not implement a replacement policy
11	DCLA	Data Cache Locking APU Available 0 The data cache does not implement the line locking APU
12	DCECA	Data Cache Error Checking Available 0 The data cache does not implement error checking
13:20	DCNWAY	Data Cache Number of Ways 0x00 The data cache is 0-way set-associative
21:31	DCSIZE	Data Cache Size 0x000 No data cache is present

8.7.6 L1 Cache Configuration Register 1 (L1CFG1)

The L1 Cache Configuration Register 1 (L1CFG1) is a 32-bit read-only register. L1CFG1 provides information about the configuration of the e200z425n3 L1 instruction cache design. The contents of the L1CFG1 register can be read using a **mfspr** instruction. The L1CFG1 register is shown in the following figure.

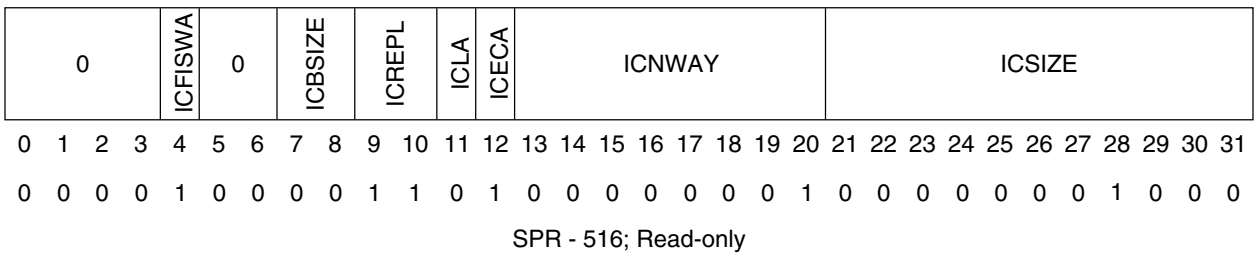


Figure 8-10. L1 Cache Configuration Register 1 (L1CFG1)

The L1CFG1 fields are described in the following table.

Table 8-20. L1CFG1 field descriptions

Bits	Name	Description
0:3	—	Reserved - read as zeros
4	ICFISWA	Instruction Cache Flush/Invalidate by Set and Way Available 1 The instruction cache supports invalidation by Set and Way via L1FINV1
5:6	—	Reserved - read as zeros
7:8	ICBSIZE	Instruction Cache Block Size 00 The instruction cache implements a block size of 32 bytes
9:10	ICREPL	Instruction Cache Replacement Policy 11 The instruction cache implements a FIFO replacement policy
11	ICLA	Instruction Cache Locking APU Available 0 The instruction cache does not implement the line locking APU
12	ICECA	Instruction Cache Error Checking Available 1 The instruction cache implements error checking
13:20	ICNWAY	Instruction Cache Number of Ways 0x01 The instruction cache is 2-way set-associative
21:31	ICSIZE	Instruction Cache Size 0x008 The size of the instruction cache is 8 KB

8.7.7 Cache Invalidate by Set and Way

e200z425n3 supports cache set/way invalidation under software control. The instruction cache may be invalidated by index and way through a **mtspr l1finv1** instruction.

The L1 Flush and Invalidate Control Registers (L1FINV1) are 32-bit SPRs used to select a cache set and way to be invalidated. This function is available even when the instruction cache is disabled.

8.7.7.1 L1FINV1

The L1FINV1 register is shown in following figure.

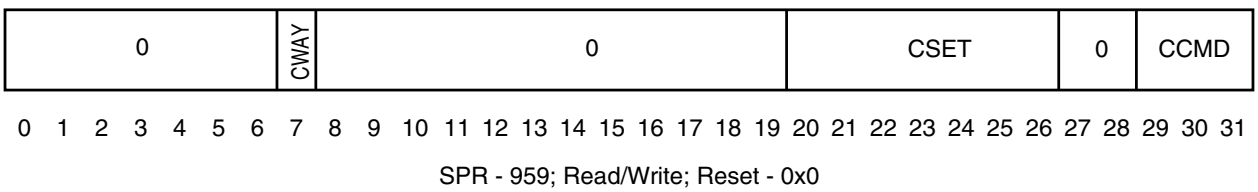


Figure 8-11. L1 Flush/Invalidate Register 1 (L1FINV1)

The L1FINV1 bits are described in the following table.

Table 8-21. L1FINV1 field descriptions

Bits	Name	Description
0:6	—	Reserved
7	CWAY	Cache Way Specifies the instruction cache way to be selected
8:19	—	Reserved
20:26	CSET	Cache Set Specifies the instruction cache set to be selected
27:28	—	Reserved
29:31	CCMD	Cache Command 000 The data contained in this entry is invalidated. LO bits are unaffected 001 Reserved 01x Reserved 100 Reserved 101 The data contained in this entry is invalidated. LO bits are cleared 11x Reserved

For invalidation operations via L1FINV1, tag ECC errors and data EDC errors are ignored, and the selected line will be invalidated regardless of any error. Invalidations conditionally affect the state of the lockout bits (LO).

8.7.8 Cache EDC/ECC Parity Protection

Cache parity protection is supported for both the tag and data arrays of the instruction cache. Seven parity check bits are provided for each tag entry for the tag array to support error detection and correction (ECC - SECDED). Eight parity check bits are provided for each doubleword in the data arrays of the I-Cache, which are used for single- and double-bit error detection (EDC - DED, double error detection). Utilizing ECC/EDC protection, many multi-bit errors are also detected.

The error detection codes also cover the cache index address of the cache line, providing additional protection against addressing errors. If any type of error (including a single-bit error in one of the index bits) is encountered in one of the index address bits, it is treated as an uncorrectable tag ECC error to protect against internal addressing failures. No attempt will be made to correct single-bit errors where the syndrome indicates an index address bit.

Tag ECC and data EDC checking is controlled by the L1CSR1[ICECE] control field. When error checking is enabled, checking is performed on each cache access. Errors are not signaled by the instruction cache when cache error checking is disabled (L1CSR1[ICECE] = 0).

If an uncorrectable tag ECC error is detected on any portion of a tag accessed during cache lookups performed because of instruction fetching, a tag ECC error is signaled, regardless of whether a cache hit or miss occurs. Otherwise, if a cache hit for an instruction fetch occurs and a data EDC error is detected on any portion of the accessed data, a parity error is also signaled.

Signaling of an ECC error or EDC error may cause a Machine Check exception to occur. One or more syndrome bits may be set in the Machine Check Syndrome register, or may instead result in a correction/auto-invalidation operation and not result in an exception being signaled. Both may occur, depending on the error action control setting in the appropriate cache control register.

8.7.8.1 Cache Line Lockout

In addition to the ECC/EDC protection employed for the cache, each cache line in the I-Cache has a lockout indicator composed of a redundant set of lockout bits. These lockout bits can selectively be set when certain errors occur in either the tag or the data portion of the cache line. Use of the lockout function and control over error conditions that cause a lockout to occur are controlled by $L1CSR1_{ICLOC}$. When the lockout function is enabled, lines that encounter selected tag ECC or data EDC errors on instruction fetch, accesses will have their lockout bits set. When the lockout indicator is set, the line will not be

replaced and will remain in an invalid state, effectively disabling it. Also, future tag ECC or data EDC errors on the line are ignored. Cache-inhibited accesses will only set lockout indicators on lines not in a locked way. In addition, no lockout indicators are set by cache-inhibited accesses when operating in machine check mode.

When operating in machine check mode, if lockout controls are enabled via $L1CSR1_{ICLOC}$, lockout bit parity errors on any line detected on a cacheable cache lookup will generate a machine check exception.

If correction/auto-invalidation is instead enabled, on each cache lookup operation for an instruction fetch, if a single- or double-bit lockout bit parity error is detected in one or more ways and lockout controls are enabled, the lockout error(s) will be corrected by rewriting all lockout bits to the asserted state, and no machine check is generated, unless the line is in a locked way. If a line in a locked way incurs a LO bit parity error, a machine check will be generated to ensure that any possible new lockout of the line is reported. For non-cacheable accesses, lockout bit parity errors will only be corrected for lines not in a locked way. Lockout bit parity errors do not generate a machine check for non-cacheable accesses in either error action mode, thus lockout bit correction is not performed for lockout bit parity errors detected on a line in a locked way.

In addition, cache contents are ignored for lines with those errors. Cache invalidate instructions do not report or correct lockout bit parity errors. For both of these cases, a future cacheable access will perform the lockout function if required.

8.7.9 Cache Error Injection

Cache error injection provides a way to test error recovery by intentionally injecting parity errors into the instruction cache.

Error injection into the instruction cache operates as follows:

- If $L1CSR1_{ICEI}$ is set, any instruction cache line fill to the instruction cache data has the associated two most significant parity check bits inverted in the instruction cache data array for each doubleword loaded.

Cache parity error injection is not performed for cache debug write accesses, since parity bit values written can be directly controlled.

In order to clear the parity errors, a cache invalidation or an invalidation of the lines that could have had an injected parity error may be performed. Line invalidation may be performed by an **icbi** instruction or an L1FINV1 invalidation operation.

8.8 Exceptions

Interrupts implemented in e200z425n3 and the exception conditions that cause them are listed in the following table.

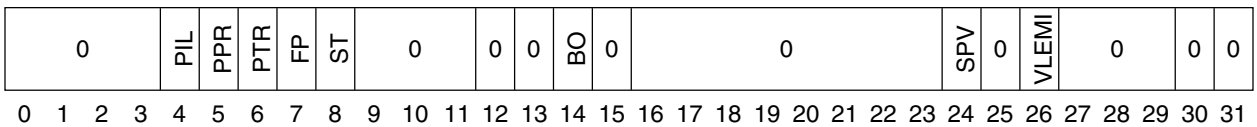
Table 8-22. Exceptions and Conditions

Interrupt type	Interrupt vector offset value	Causing conditions
System reset	none, vector to [p_rstbase[0:29]] || 2'b00	Reset
Critical Input	0x00[1, 1]	**p_critint_b** is asserted and MSR_{CE} = 1.
Machine check	0x10	1. **p_mcp_b** transitions from negated to asserted 2. ISI or Bus Error on first instruction fetch for an exception handler 3. Parity Error signaled on Cache access 4. Parity Error signaled on Local Memory access 5. External bus error 6. Stack limit check failure
Machine check (NMI)	0x10	Non-Maskable Interrupt
Data Storage	0x20	Access control
Instruction Storage	0x30	Access control
External Input	0x40[2]	Interrupt Controller interrupt and MSR_{EE} = 1
Alignment	0x50	1. **lmw**, **stmw** not word aligned 2. **lwarx** or **stwcx.** not word aligned, **lharx** or **sthcx.** not halfword aligned 3. LSP ld and st instructions not properly aligned 4. **dcbz**
Program	0x60	Illegal, Privileged, Trap
Performance Monitor	0x70	Performance Monitor Enabled Condition or Event w/$PMGC0_{UDI}$ = 0
System call	0x80	Execution of the System Call (**se_sc**) instruction
Debug	0x90	Trap, Instruction Address Compare, Data Address Compare, Instruction Complete, Branch Taken, Return from Interrupt, Interrupt Taken, Debug Counter, External Debug Event, Unconditional Debug Event, Performance Monitor Enabled Condition or Event w/$PMGC0_{UDI}$ = 1, MPU
EFPU Data Exception	0xA0	Embedded Floating-point Data Exception
EFPU Round Exception	0xB0	Embedded Floating-point Round Exception
TBD	0xC0–0xF0	Reserved for future processor use

1. Autovectored Critical Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.
2. Autovectored External Input interrupts use this offset value. Vectored interrupts supply an interrupt vector offset directly.

8.8.1 Exception Syndrome Register (ESR)

The Exception Syndrome Register (ESR) provides a *syndrome* to differentiate between exceptions that can generate the same interrupt type.



SPR - 62; Read/Write; Reset - 0x0

Figure 8-12. Exception Syndrome Register (ESR)

The ESR bits are defined in the following table.

Table 8-23. ESR field descriptions

Bit(s)	Name	Description	Associated interrupt type
0:3	—	Reserved	—
4	PIL	Illegal Instruction exception For e200z425n3, PIL is used for both illegal and unimplemented instructions.	Program
5	PPR	Privileged Instruction exception	Program
6	PTR	Trap exception	Program
7	FP	Floating-point operation	Program
8	ST	Store operation	Alignment Data Storage
9:11	—	Reserved	—
12	—	Reserved	—
13	—	Reserved	—
14	BO	Byte Ordering exception Mismatched Instruction Storage exception	Instruction Storage
15	—	Reserved	—
16:23	—	Reserved	—
24	SPV	LSP/EFPU APU Operation	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Alignment Data Storage
25	—	Reserved	—

Table continues on the next page...

Table 8-23. ESR field descriptions (continued)

Bit(s)	Name	Description	Associated interrupt type
26	VLEMI	VLE Mode Instruction	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Data Storage Instruction Storage Alignment Program System Call
27:29	—	Reserved	—
30	—	Reserved	—
31	—	Reserved	—

8.8.2 Machine State Register (MSR)

The Machine State Register defines the state of the processor. The e200z425n3 MSR is shown in the following figure.

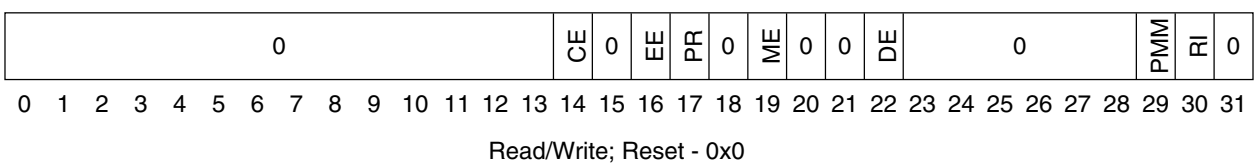


Figure 8-13. Machine State Register (MSR)

The MSR bits are defined in the following table.

Table 8-24. MSR field descriptions

Bit(s)	Name	Description
0:13	—	Reserved
14	CE	Critical Interrupt Enable 0 Critical Input interrupts are disabled. 1 Critical Input interrupts are enabled.
15	—	Reserved
16	EE	External Interrupt Enable 0 External Input interrupts are disabled. 1 External Input interrupts are enabled.

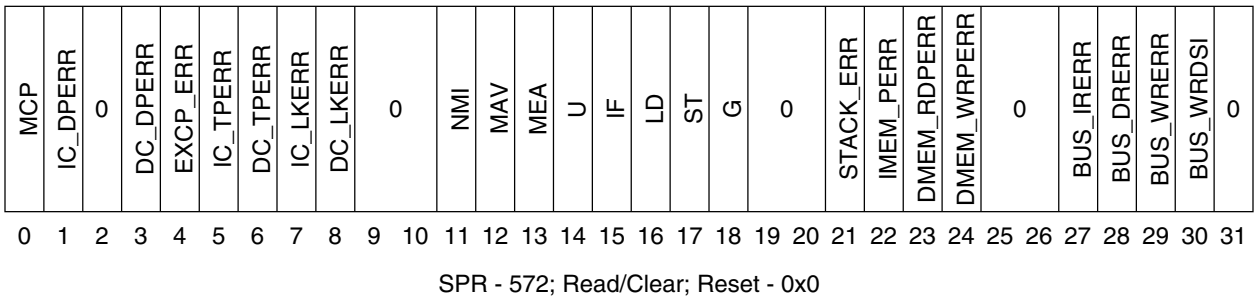
Table continues on the next page...

Table 8-24. MSR field descriptions (continued)

Bit(s)	Name	Description
17	PR	Problem State 0 The processor is in supervisor mode, can execute any instruction, and can access any resource (e.g. GPRs, SPRs, MSR, etc.). 1 The processor is in user mode, cannot execute any privileged instruction, and cannot access any privileged resource.
18	—	Reserved
19	ME	Machine Check Enable 0 Asynchronous Machine Check interrupts are disabled. 1 Asynchronous Machine Check interrupts are enabled.
20	—	Reserved
21	—	Reserved
22	DE	Debug Interrupt Enable 0 Debug interrupts are disabled. 1 Debug interrupts are enabled.
23:28	—	Reserved
29	PMM	PMM Performance monitor mark bit System software can set PMM when a marked process is running to enable statistics to be gathered only during the execution of the marked process. MSR_{PR} and MSR_{PMM} together define a state that the processor (supervisor or user) and the process (marked or unmarked) may be in at any time. If this state matches an individual state specified in the PMLCa*n* Performance Monitor registers, the state for which monitoring is enabled, counting is enabled.
30	RI	Recoverable Interrupt This bit is provided for software use to detect nested machine check exception conditions. This bit is cleared by hardware when a Machine Check interrupt is taken.
31	—	Reserved

8.8.3 Machine Check Syndrome Register (MCSR)

When the processor takes a machine check interrupt, it updates the Machine Check Syndrome Register (MCSR) to differentiate between machine check conditions. The MCSR is shown in the following figure.



SPR - 572; Read/Clear; Reset - 0x0

Figure 8-14. Machine Check Syndrome Register (MCSR)

The following table describes MCSR fields. The MCSR indicates the source of a machine check condition.

All bits in the MCSR are implemented as "write 1 to clear." Software in the machine check handler is expected to clear the MCSR bits it has sampled prior to re-enabling MSR_{ME} to avoid a redundant machine check exception and to prepare for updated status bit information on the next machine check interrupt.

Note that any set bit in the MCSR other than status-type bits will cause a subsequent machine check interrupt once $MSR_{ME} = 1$.

Table 8-25. Machine Check Syndrome Register (MCSR) field descriptions

Bit	Name	Description	Exception type[1, 1]	Recoverable
0	MCP	Machine check input pin	Async Mchk	Maybe
1	IC_DPERR	Instruction Cache data array parity error	Async Mchk	Precise
2	—	Reserved	—	—
3	DC_DPERR	Data Cache data array parity error Will not be set by e200z425n3 since no data cache is present.	Async Mchk	Maybe
4	EXCP_ERR	ISI or Bus Error on first instruction fetch for an exception handler	Async Mchk	Precise
5	IC_TPERR	Instruction Cache Tag parity error	Async Mchk	Precise
6	DC_TPERR	Data Cache Tag parity error Will not be set by e200z425n3 since no data cache is present.	Async Mchk	Maybe
7	IC_LKERR	Instruction Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the I-Cache for certain situations. May also be set on locked line refill error.	Async Mchk	—

Table continues on the next page...

Table 8-25. Machine Check Syndrome Register (MCSR) field descriptions (continued)

Bit	Name	Description	Exception type[1, 1]	Recoverable
8	DC_LKERR	Data Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the D-Cache for certain situations. May also be set on locked line refill error. Will not be set by e200z425n3 since no data cache is present.	Async Mchk	—
9:10	—	Reserved	—	—
11	NMI	NMI input pin	NMI	—
12	MAV	MCAR Address Valid Indicates that the address contained in the MCAR was updated by hardware to correspond to the first detected Async Mchk error condition	Status	—
13	MEA	MCAR holds Effective Address If MAV = 1, MEA = 1 indicates that the MCAR contains an effective address and MEA = 0 indicates that the MCAR contains a physical address. Note that since no address translation mechanism is present on e200z425n3, the MEA value is always set to \0'.	Status	—
14	U	User Indicates the value captured in MCAR was generated in user mode.	Status	—
15	IF	Instruction Fetch Error Report An error occurred during the fetch of an instruction and the instruction attempted to execute. This could be due to an internal parity error, or an external bus error. MCSRR0 contains the instruction address.	Error Report	Precise
16	LD	Load type instruction Error Report An error occurred during the attempt to execute the load type instruction located at the address stored in MCSRR0. This could be due to an internal parity error, stack limit check error, or an external bus error.	Error Report	Precise
17	ST	Store type instruction Error Report An error occurred during the attempt to execute the store type instruction located at the address stored in MCSRR0. This could be due to an internal parity error, stack limit check error, or on certain external bus errors.	Error Report	Precise

Table continues on the next page...

Table 8-25. Machine Check Syndrome Register (MCSR) field descriptions (continued)

Bit	Name	Description	Exception type[1, 1]	Recoverable
18	G	Guarded instruction Error Report An error occurred during the attempt to execute the load or store type instruction located at the address stored in MCSRR0 and the access was guarded and encountered an error on the external bus, or an uncorrectable DMEM error.	Error Report	Precise
19:20	—	Reserved	—	—
21	STACK_ERR	Stack Access Limit Check Error Indicates a limit check failure on a CPU data access using R1 in the <EA> calculation.	Async Mchk	Precise
22	IMEM_PERR	Instruction Mem (IMEM) Parity Error Indicates an uncorrectable error in the IMEM on a CPU port access.	Async Mchk	Precise
23	DMEM_RDPERR	Data Mem (DMEM) Parity Error Indicates an uncorrectable error in the DMEM on a CPU port read access.	Async Mchk	Precise
24	DMEM_WRPERR	Data Mem (DMEM) Write Parity Error Indicates an uncorrectable error in the DMEM on a CPU port write access.	Async Mchk	Maybe
25:26	—	Reserved	—	—
27	BUS_IRERR	Read bus error on Instruction recovery linefill	Async Mchk	Precise if data used
28	BUS_DRERR	Read bus error on data load	Async Mchk	Precise if data used
29	BUS_WRERR	Write bus error on store to bus or DMEM imprecise write error	Async Mchk	Unlikely
30	BUS_WRDSI	Write bus error on buffered store to bus with DSI signaled. Set concurrently with BUS_WRERR for this case	Async Mchk	Unlikely
31	—	Reserved	—	—

1. The Exception Type indicates the exception type associated with a given syndrome bit.
   - "Error Report" indicates that this bit is only set for error report exceptions which cause machine check interrupts. These bits are only updated when the machine check interrupt is actually taken. Error report exceptions are not gated by MSR_{ME}. These are synchronous exceptions. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.
   - "Status" indicates that this bit is provides additional status information regarding the logging of a machine check exception. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.
   - "NMI" indicates that this bit is only set for the non-maskable interrupt type exception which causes a machine check interrupt. This bit is only updated when the machine check interrupt is actually taken. NMI exceptions are not gated by MSR_{ME}. This is an asynchronous exception. This bit will remain set until cleared by software writing a '1' to the bit position.
   - "Async Mchk" indicates that this bit is set for an asynchronous machine check exception. These bits are set immediately upon detection of the error. Once any "Async Mchk" bit is set in the MCSR, a machine check interrupt will occur if $MSR_{ME} = 1$. If $MSR_{ME} = 0$, the machine check exception will remain pending. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.

8.8.4 Interrupt Vector Prefix Registers (IVPR)

The Interrupt Vector Prefix Register is used during interrupt processing for determining the starting address of a software handler used to handle an interrupt. The value of the Vector Offset selected for a particular interrupt type is concatenated with the Vector Base value held in the IVPR to form an instruction address from which execution is to begin. The format of IVPR is shown in the following figure.

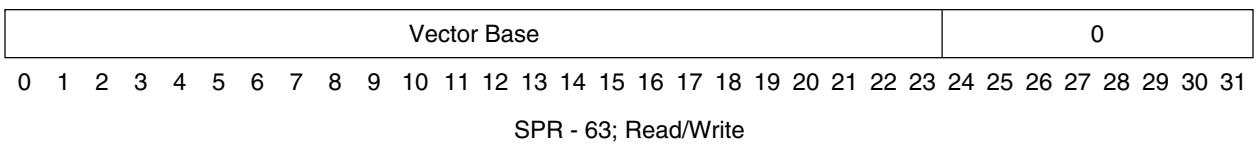


Figure 8-15. Interrupt Vector Prefix Register (IVPR)

The IVPR fields are defined in the following table.

Table 8-26. IVPR field descriptions

Bit(s)	Name	Description
0:23 (32:55)	Vector Base	Vector Base This field is used to define the base location of the vector table, aligned to a 256 byte boundary. This field provides the high-order 24 bits of the location of all interrupt handlers (unless hardware vectoring is used). The vector offset value appropriate for the type of exception being processed is concatenated with the IVPR Vector Base to form the address of the handler in memory. For hardware-vectored interrupts, the value of the supplied interrupt vector is logically OR'ed with the low order bits of the Vec Base Field to determine the interrupt handler address.
24:31 (56:63)	—	Reserved

8.8.5 Interrupt Definitions

8.8.5.1 Critical Input Interrupt (offset 0x00)

A Critical Input exception is signaled to the processor by the assertion of the critical interrupt pin. If the exception is enabled by MSR_{CE}, the Critical Input interrupt is taken.

A Critical Input interrupt may be delayed by other higher priority exceptions or if MSR_{CE} is cleared when the exception occurs.

The following table lists register settings when a Critical Input interrupt is taken.

Table 8-27. Critical Input Interrupt—register settings

Register	Setting description					
CSRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
CSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	0	FE0	0	DS	0
	EE	0	DE	—/0[1]	PMM	0
	PR	0			RI	—
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x00 (autovectored)					
	$IVPR_{0:15}$ || (IVPR16:29 | **p_voffset[0:13]**) || 2'b00 (non-autovectored)					

1. Clearing of DE is optionally supported by control in HID0.

The MSR_{DE} bit is not automatically cleared by a Critical Input interrupt, but it can be configured to be cleared via the HID0 register ($HID0_{CICLRDE}$).

8.8.5.2 Machine Check Interrupt (offset 0x10)

The EIS Machine Check APU defines a separate set of save/restore registers (MCSRR0/1), a Machine Check Syndrome Register (MCSR) to record the source(s) of machine checks, and a Machine Check Address Register (MCAR) to hold an address associated with a machine check for certain classes of machine checks. Return from Machine Check instructions (**se_rfmci**) are also provided to support returns using MCSRR0/1.

The MSR_{DE} bit is not automatically cleared by a Machine Check exception, but it can be configured to be cleared or left unchanged via the HID0 register ($HID0_{MCCLRDE}$).

When a Machine Check interrupt is taken, registers are updated as shown in the following table.

Table 8-28. Machine Check Interrupt—register settings

Register	Setting description
MCSRR0	On a best-effort basis, set to the address of some instruction that was executing or about to be executing when the machine check condition occurred.
MCSRR1	Set to the contents of the MSR at the time of the interrupt

Table continues on the next page...

Table 8-28. Machine Check Interrupt—register settings (continued)

Register	Setting description					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	0	IS	0
	CE	0	FE0	0	DS	0
	EE	0	DE	0/—[1]	PMM	0
	PR	0			RI	0
ESR	Unchanged					
MCSR	Updated to reflect the source(s) of a machine check. Hardware only sets appropriate bits, no previously set bits are cleared by hardware.					
Vector	$IVPR_{0:23}$ || 0x10					

1. Clearing of DE is optionally supported by control in HID0.

8.8.5.3 Data Storage Interrupt (offset 0x20)

A Data Storage interrupt (DSI) may occur if no higher priority exception exists and a Read or Write Access Control exception condition exists.

The following table lists register settings when a DSI is taken.

Table 8-29. Data Storage Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting load/store instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Access:		[ST], [SPV], VLEMI. All other bits cleared.			
MCSR	Unchanged					
DEAR	For Access Control exceptions, set to the effective address of the access that caused the violation.					
Vector	$IVPR_{0:23}$ || 0x20					

8.8.5.4 Instruction Storage Interrupt (offset 0x30)

An Instruction Storage interrupt (ISI) occurs when no higher priority exception exists and an Execute Access Control exception occurs.

The following table lists register settings when an ISI is taken.

Table 8-30. Instruction Storage Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x30					

8.8.5.5 External Input Interrupt (offset 0x40)

An External Input exception is signaled to the processor by the assertion of an interrupt from the interrupt controller. The input is a level-sensitive signal expected to remain asserted until the core acknowledges the external interrupt. If the input is negated early, recognition of the interrupt request is not guaranteed. When the core detects the exception, if the exception is enabled by MSR_{EE}, it takes the External Input interrupt.

An External Input interrupt may be delayed by other higher priority exceptions or if MSR_{EE} is cleared when the exception occurs.

The following table lists register settings when an External Input interrupt is taken.

Table 8-31. External Input Interrupt—Register Settings

Register	Setting description					
SRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Unchanged					

Table continues on the next page...

Table 8-31. External Input Interrupt—Register Settings (continued)

Register	Setting description
MCSR	Unchanged
DEAR	Unchanged
Vector	$IVPR_{0:23}$ || 0x40 (autovectored) IVPR0:15 || (IVPR16:29 | **p_voffset[0:13]**) || 2'b00 (non-autovectored)

8.8.5.6 Alignment Interrupt (offset 0x50)

An Alignment exception is generated when any of the following occurs:

- The operand of **lmw** or **stmw** is not word aligned.
- The operand of **lwarx** or **stwcx.** is not word aligned.
- The operand of **lharx** or **sthcx.** is not halfword aligned.
- Execution of a **dcbz** instruction is attempted.
- Execution of an LSP APU load or store instruction that is not properly aligned.

The following table lists register settings when an alignment interrupt is taken.

Table 8-32. Alignment Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting load/store/dcbz instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	[ST], [SPV], VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Set to the effective address of a byte of the load or store access causing the violation.					
Vector	$IVPR_{0:23}$ || 0x50					

8.8.5.7 Program Interrupt (offset 0x60)

A program interrupt occurs when no higher priority exception exists and one or more of the following exception conditions occur:

- Illegal Instruction exception
- Privileged Instruction exception
- Trap exception

The e200z425n3 will invoke an Illegal Instruction program exception on attempted execution of the following instructions:

- Unimplemented instructions
- An instruction from the illegal instruction class
- **mtspr** and **mfspr** instructions with an undefined SPR specified
- **mtdcr** and **mfdcr** instructions with an undefined DCR specified

The e200z425n3 will invoke a Privileged Instruction program exception on attempted execution of the following instructions when $MSR_{PR} = 1$ (user mode):

- A privileged instruction
- **mtspr** and **mfspr** instructions that specify a SPRN value with $SPRN_5 = 1$ (even if the SPR is undefined)

The e200z425n3 will invoke an Trap exception on execution of the **tw** instruction if the trap conditions are met and the exception is not also enabled as a Debug interrupt.

The core will invoke an Illegal instruction program exception on attempted execution of the instructions **lswi**, **lswx**, **stswi**, **stswx**, **mfapidi**, **mfdcrx**, **mtdcrx**, or on any *PowerISA 2.06* floating-point instruction. All other defined or allocated instructions that are not implemented by core will cause an illegal instruction program exception.

The following table lists register settings when a Program interrupt is taken.

Table 8-33. Program Interrupt—register settings

Register	Setting description
SRR0	Set to the effective address of the excepting instruction.
SRR1	Set to the contents of the MSR at the time of the interrupt

Table continues on the next page...

Table 8-33. Program Interrupt—register settings (continued)

Register	Setting description					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Illegal, Unimplemented:		PIL, VLEMI. All other bits cleared.			
	Privileged:		PPR, VLEMI. All other bits cleared.			
	Trap:		PTR, VLEMI. All other bits cleared.			
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x60					

8.8.5.8 Performance Monitor Interrupt (offset 0x70)

A performance monitor interrupt that may be generated by an enabled condition or event. An enabled condition or event is as follows:

A PMC*x* register overflow condition occurs with the following settings:

- PMLCax_{CE} = 11; that is, for the given counter the overflow condition is enabled.
- PMCx_{OV} = 11; that is, the given counter indicates an overflow.

For a performance monitor interrupt to be signaled on an enabled condition or event, $PMGC0_{PMIE}$ must be set.

Although an exception condition may occur with MSR_{EE} = 0, the interrupt cannot be taken until MSR_{EE} = 1.

The priority of the performance monitor interrupt is below all other asynchronous interrupts.

The following table lists register settings when an performance monitor interrupt is taken.

Table 8-34. Performance Monitor Interrupt—Register Settings

Register	Setting description
SRR0/ DSRR0[1, 1]	Set to the effective address of the next instruction to be executed.

Table continues on the next page...

Table 8-34. Performance Monitor Interrupt—Register Settings (continued)

Register	Setting description					
SRR1/ DSRR1[1]	Set to the contents of the MSR at the time of the interrupt.					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—/0[2]	FE0	0	DS	0
	EE	0/—[3]	DE	—/0[4]	PMM	0
	PR	0			RI	—
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x70					

1. DSRR0/1 are used if $PMGC0_{UDI}$ = 1
2. CE is cleared if $PMGC0_{UDI}$ = 1 and $HID0_{DCLRCE}$ = 1
3. EE is not cleared if $PMGC0_{UDI}$ = 1 and $HID0_{DCLREE}$ = 0
4. DE is cleared if $PMGC0_{UDI}$ = 1

8.8.5.9 System Call Interrupt (offset 0x80)

A System Call interrupt occurs when a System Call (**se_sc**) instruction is executed and no higher priority exception exists.

The following table lists register settings when a System Call interrupt is taken.

Table 8-35. System Call Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the instruction *following* the system call instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x80					

8.8.5.10 Debug Interrupt (offset 0x90)

There are multiple sources that can signal a Debug exception. A Debug interrupt occurs when no higher priority exception exists, a Debug exception exists in the Debug Status Register, and Debug interrupts are enabled (both $DBCR0_{IDM} = 1$ (internal debug mode) and $MSR_{DE} = 1$).

The following table lists register settings when a Debug interrupt is taken.

Table 8-36. Debug Interrupt—register settings

Register	Setting description					
DSRR0[1]	Set to the effective address of the excepting instruction for IAC, BRT, RET, CRET, and TRAP.Set to the effective address of the next instruction to be executed *following* the excepting instruction for DAC (usually) and ICMP. For a UDE, IRPT, CIRPT, DCNT, or DEVT type exception, set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
DSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—/0[2, 2]	FE0	0	DS	0
	EE	—/0[2]	DE	0	PMM	0
	PR	0			RI	—
DBSR[3]	Unconditional Debug Event:		UDE			
	Instr. Complete Debug Event:		ICMP			
	Branch Taken Debug Event:		BRT			
	Interrupt Taken Debug Event:		IRPT			
	Critical Interrupt Taken Debug Event:		CIRPT			
	Trap Instruction Debug Event:		TRAP			
	Instruction Address Compare:		{IAC}			
	Data Address Compare:		{DACR, DACW}			
	Debug Notify Interrupt:		DNI			
	Return Debug Event:		RET			
	Critical Return Debug Event:		CRET			
	External Debug Event:		{DEVT1, DEVT2}			
	Performance Monitor Debug Event:		PMI			
	MPU Debug Event:		MPU			
	and optionally, an Imprecise Debug Event flag		{IDE}			
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x90					

1. Assumes that the Debug interrupt is precise

2. Conditional based on control bits in HID0
3. Note that multiple DBSR bits may be set

8.8.5.11 Embedded Floating-point Data Interrupt (offset 0xA0)

The Embedded Floating-point Data interrupt is taken if no higher priority exception exists and an EFPU Floating-point Data exception is generated. When a Floating-point Data exception occurs, the processor suppresses execution of the instruction causing the exception.

The following table lists register settings when an EFPU Floating-point Data interrupt is taken.

Table 8-37. Embedded Floating-point Data Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting EFPU instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	SPV, VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0xA0					

8.8.5.12 Embedded Floating-point Round Interrupt (offset 0xB0)

The Embedded Floating-point Round interrupt is taken when an EFPU floating-point instruction generates an inexact result and inexact exceptions are enabled.

The following table lists register settings when an EFPU Floating-point Round interrupt is taken.

Table 8-38. Embedded Floating-point Round Interrupt—register settings

Register	Setting description
SRR0	Set to the effective address of the instruction following the excepting EFPU instruction.
SRR1	Set to the contents of the MSR at the time of the interrupt

Table continues on the next page...

Table 8-38. Embedded Floating-point Round Interrupt—register settings (continued)

Register	Setting description					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	SPV, VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0xB0					

8.8.5.13 System Reset Interrupt

The System Reset exception is a non-maskable, asynchronous exception signaled to the processor through the assertion of system-defined signals.

A System Reset may be initiated by either a software reset or during power-on reset.

When a reset request occurs, the processor branches to the system reset exception vector without attempting to reach a recoverable state. If reset occurs during normal operation, all operations cease and the machine state is lost.

The following table lists register settings when a System Reset interrupt is taken.

Table 8-39. System Reset Interrupt—register settings

Register	Setting description					
CSRR0	Undefined					
CSRR1	Undefined					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	0	IS	0
	CE	0	FE0	0	DS	0
	EE	0	DE	0	PMM	0
	PR	0			RI	0
ESR	Cleared					
DEAR	Undefined					
Vector	[**p_rstbase[0:29]**] || 2'b00					

8.9 Memory Protection Unit (MPU)

8.9.1 MPU Overview

The e200z425n3 Memory Protection Unit (MPU) protects regions of memory, with the following feature set:

- 24-entry region descriptor table with support for 6 arbitrary-sized instruction memory regions, 12 arbitrary-sized data memory regions, and 6 additional arbitrary-sized regions programmable as instruction or data memory regions
- Ability to set access permissions and memory attributes on a per-region basis
- Process ID aware, with per-bit masking of TID values
- Capability for masking upper address bits in the range comparison
- Capability of bypassing permissions checking for selected access types
- Per-entry write-once logic for entry protection
- Hardware flash invalidation support and per-entry invalidation protection controls
- Ability to optionally utilize region descriptors for generating debug events and watchpoints
- Software managed by **mpure** and **mpuwe** instructions

8.9.2 Software Interface and MPU Instructions

The MPU entries are accessed indirectly through several MPU Assist (MAS) registers. Software can write and read the MAS registers with **mtspr** and **mfspr** instructions. These registers contain information related to reading and writing a given entry within the MPU. Data is read from the MPU into the MAS registers with an **mpure** (MPU read entry) instruction. Data is written to the MPU from the MAS registers with an **mpuwe** (MPU write entry) instruction.

The **mpure**, **mpuwe**, and **mpusync** instructions are privileged.

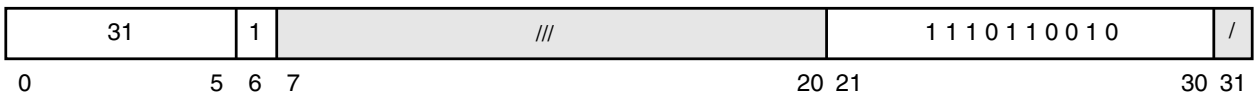
8.9.3 MPU Read Entry Instruction (mpure)

The MPU read entry instruction causes the content of a single MPU entry to be placed in the MPU assist registers. The entry is specified by the INST, SHD, and ESEL fields of the MAS0 register. The entry contents are placed in the MAS0, MAS1, MAS2, and MAS3 registers.

mpure

mpu read entry

mpure




```
mpu_entry_id = MAS0(INST, SHD, ESEL)
result = MPU(mpu_entry_id)
MAS0, MAS1, MAS2, MAS3 = result
```

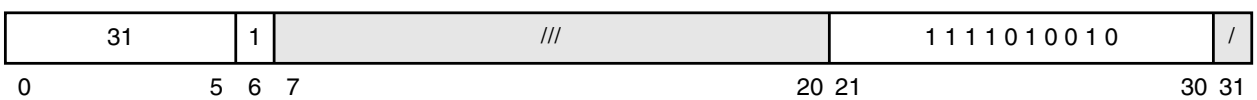
8.9.4 MPU Write Entry Instruction (mpuwe)

The MPU write entry instruction causes the contents of certain fields within the MPU assist registers MAS0, MAS1, MAS2, and MAS3 to be written into a single MPU entry in the MPU. The entry written is specified by the INST, SHD, and ESEL fields of the MAS0 register.

mpuwe

mpu write entry

mpuwe




```
mpu_entry_id = MAS0(INST, SHD, ESEL)
MPU(mpu_entry_id) = MAS0, MAS1, MAS2, MAS3
```

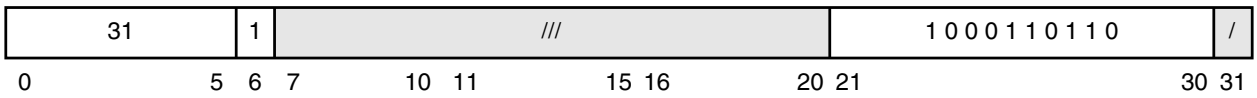
8.9.5 MPU Synchronize Instruction (mpusync)

The MPU Synchronize instruction is treated as a privileged no-op by the core.

mpusync

MPU Synchronize
mpusync

mpusync



8.9.6 MMU/MPU Configuration Register (MMUCFG)

The MMU/MPU Configuration Register (MMUCFG) is a 32-bit read-only register. The SPR number for MMUCFG is 1015 in decimal. MMUCFG provides information about the configuration of the e200z425n3 MPU design. The MMUCFG register is shown in the following figure.

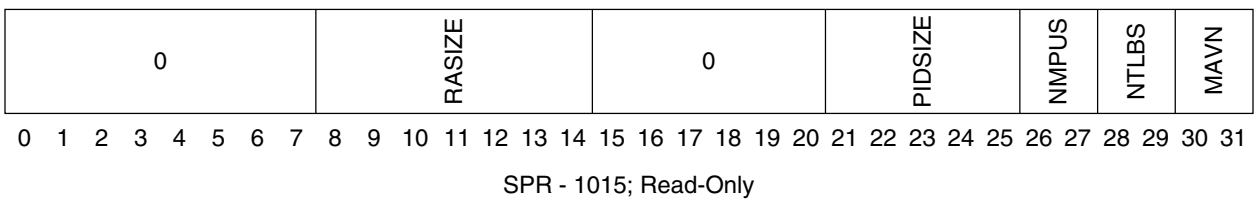


Figure 8-16. MMU/MPU Configuration Register (MMUCFG)

The MMUCFG bits are described in the following table.

Table 8-40. MMUCFG field descriptions

Bits	Name	Function
0:7	—	Reserved
8:14	RASIZE	Number of Bits of Real Address supported 0100000 This version of the MPU implements 32 real address bits
15:16	—	Reserved
17:20	—	Reserved
21:25	PIDSIZE	PID Register Size 00111 PID registers contain 8 bits in this version of the MPU
26:27	NMPUS	Number of MPUs 01 This version of the MMU implements one MPU structure: a fully associative MPU for MPU0
28:29	NTLBS	Number of TLBs 00 This version of the MMU implements no TLB structures
30:31	MAVN	MMU Architecture Version Number 11 This version of the MMU implements Version 3.1 of the EIS MMU Architecture

8.9.7 MPU0 Configuration Register (MPU0CFG)

The MPU0 Configuration Register (MPU0CFG) is a 32-bit read-only register. The SPR number for MPU0CFG is 692 in decimal. MPU0CFG provides information about the configuration of MPU0 in the e200z425n3 MPU. The MPU0CFG register is shown in following figure.

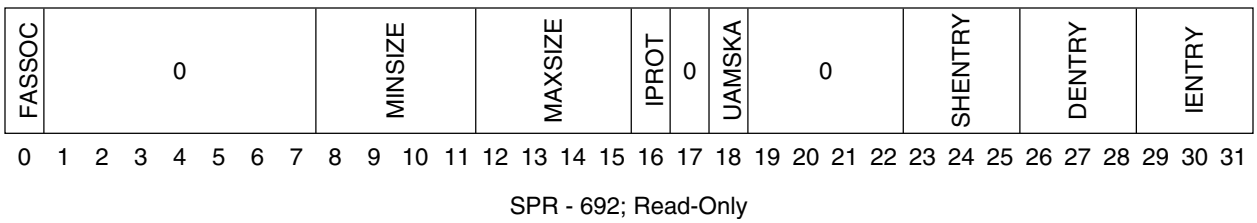


Figure 8-17. MPU0 Configuration Register (MPU0CFG)

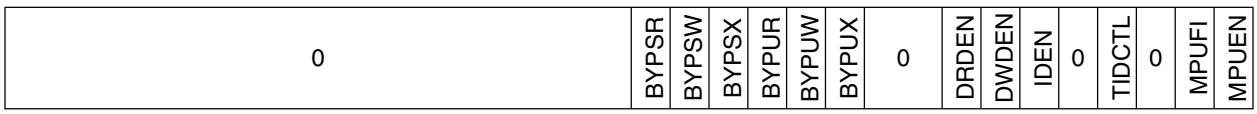
The MPU0CFG bits are described in the following table.

Table 8-41. MPU0CFG field descriptions

Bits	Name	Function
0	FASSOC	Fully Associative 1 Indicates that MPU0 is fully associative
1:7	—	Reserved for non-fully associative implementation use
8:11	MINSIZE	Minimum Region Size Due to the use of access address matching, the effective smallest region size is 8 bytes 0x0 Smallest region size is 1 byte
12:15	MAXSIZE	Maximum Region Size 0xB Largest region size is 4 GB
16	IPROT	Invalidate Protect Capability 1 Invalidate Protect Capability is supported in MPU0
17	—	Reserved
18	UAMSKA	Upper Address Masking Availability 1 Upper Address Masking is Available
19:22	—	Reserved
23:25	SHENTRY	Number of Shared (configurable for I or D) Entries 0x2 Indicates that MPU0 contains six shared entries
26:28	DENTRY	Number of Data Entries 0x4 Indicates that MPU0 contains 12 dedicated data entries
29:31	IENTRY	Number of Instruction Entries 0x2 Indicates that MPU0 contains six dedicated Instruction entries

8.9.8 MPU0 Control and Status Register 0 (MPU0CSR0)

The MPU0 Control and Status Register 0 (MPU0CSR0) is a 32-bit register. The SPR number for MPU0CSR0 is 1014 in decimal. MPU0CSR0 controls the operation of the MPU. The MPU0CSR0 register is shown in the following figure.



SPR - 1014; Read/ Write; Reset - 0x0[1]

Figure 8-18. MPU0 Control and Status Register 0 (MPU0CSR0)

Note

[1] Reset by processor reset **p_reset_b** if EDBCR0[EDM] = 0, as well as unconditionally by an internal power-on reset signal. If EDBCR0[EDM] = 1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**. Specifically, this applies to the DRDEN, DWDEN, and IDEN control bits.

The MPU0CSR0 bits are described in the following table.

Table 8-42. MPU0CSR0 field descriptions

Bits	Name	Description
0:15	—	Reserved
16	BYPSR	Bypass MPU protections for Supervisor Read accesses This bit controls whether supervisor-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode read accesses do not generate a DSI when no MPU match occurs. Supervisor-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for Supervisor Read accesses 1 Bypass MPU protections for Supervisor Read accesses
17	BYPSW	Bypass MPU protections for Supervisor Write accesses This bit controls whether supervisor-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode write accesses do not generate a DSI when no MPU match occurs. Supervisor-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for Supervisor Write accesses 1 Bypass MPU protections for Supervisor Write accesses

Table continues on the next page...

Table 8-42. MPU0CSR0 field descriptions (continued)

Bits	Name	Description
18	BYPSX	Bypass MPU protections for Supervisor Instruction accesses This bit controls whether supervisor-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, supervisor-mode instruction accesses do not generate an ISI when no MPU match occurs. Supervisor-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI). 0 No Bypass of MPU protections for Supervisor Instruction accesses 1 Bypass MPU protections for Supervisor Instruction accesses
19	BYPUR	Bypass MPU protections for User Read accesses This bit controls whether user-mode read accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode read accesses do not generate a DSI when no MPU match occurs. User-mode read accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for User Read accesses 1 Bypass MPU protections for User Read accesses
20	BYPUW	Bypass MPU protections for User Write accessesThis bit controls whether user-mode write accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode write accesses do not generate a DSI when no MPU match occurs. User-mode write accesses are still compared to MPU entries, and a hit will provide access attributes (CI, G). 0 No Bypass of MPU protections for User Write accesses 1 Bypass MPU protections for User Write accesses
21	BYPUX	Bypass MPU protections for User Instruction accesses This bit controls whether user-mode instruction accesses bypass the MPU protection mechanism. When bypass is enabled, user-mode instruction accesses do not generate an ISI when no MPU match occurs. User-mode instruction accesses are still compared to MPU entries, and a hit will provide access attributes (CI). 0 No Bypass of MPU protections for User Instruction accesses 1 Bypass MPU protections for User Instruction accesses
22:23	—	Reserved
24	DRDEN	Data Read Debug Enable This bit controls whether data read accesses are enabled to generate MPU debug events. When disabled, no data read access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data read accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for data read accesses 1 MPU debug events are enabled for data read accesses

Table continues on the next page...

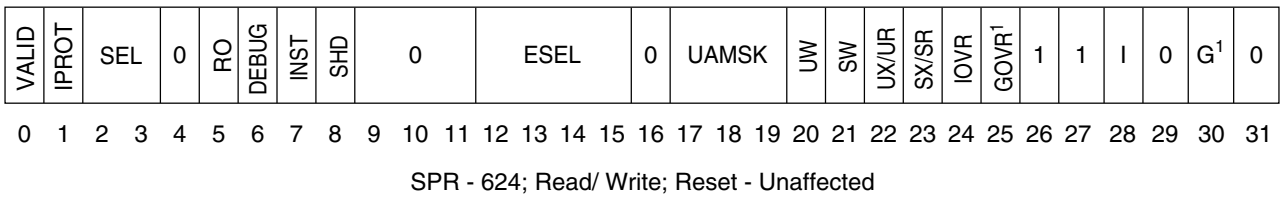
Table 8-42. MPU0CSR0 field descriptions (continued)

Bits	Name	Description
25	DWDEN	Data Write Debug Enable This bit controls whether data write accesses are enabled to generate MPU debug events. When disabled, no data write access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. If such a match occurs, no MPU debug event is generated, and no access protection is performed. When enabled, data write accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for data write accesses 1 MPU debug events are enabled for data write accesses
26	IDEN	Instruction Debug Enable This bit controls whether instruction accesses are enabled to generate MPU debug events. When disabled, no instruction access will generate an MPU debug event, regardless of whether an access match occurs to a valid MPU region descriptor with DEBUG = 1. When enabled, instruction accesses are not blocked from generating MPU debug events. 0 MPU debug events are disabled for instruction accesses 1 MPU debug events are enabled for instruction accesses
27	—	Reserved
28	TIDCTL	TID usage Control When TIDCTL is set to 1, the TID match is disabled (forced match) in Supervisor mode. 0 TID comparisons performed normally 1 No TID comparisons are performed for matching while in Supervisor mode
29	—	Reserved
30	MPUFI	MPU flash invalidate When written to a 1, a MPU invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. MPU invalidation operations require 3 cycles to complete. 0 No flash invalidate 1 MPU1 invalidation operation **Note:** Entries that have the IPROT bit set will not be invalidated.
31	MPUEN	MPU Enable This bit enables operation of the MPU. When enabled, access addresses are compared to each entry in the MPU for a match condition. If no match condition occurs, and the access type is not enabled to bypass the MPU protections, then an ISI or DSI condition is signaled for the access. Note that this bit is ignored for matching entries with DEBUG = 1 if in EDM and hardware owns MPU Debug entries. 0 MPU is disabled 1 MPU is enabled

8.9.9 MPU Assist Registers (MAS)

The core uses four special-purpose registers (MAS0, MAS1, MAS2, and MAS3) to facilitate reading and writing MPU entries. The MAS registers can be read or written using the **mfspr** and **mtspr** instructions.

The MAS0 register is shown in Figure 8-19. Fields are defined in Table 8-43.



SPR - 624; Read/ Write; Reset - Unaffected

Figure 8-19. MPU Assist Register 0 (MAS0)

Note

[1] This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1

Table 8-43. MAS0—MPU Read/Write and Replacement Control

Bit	Name	Comments, or Function when Set
0	VALID	MPU Entry Valid 0 = This MPU entry is invalid 1 = This MPU entry is valid
1	IPROT	Invalidation Protect 0 = Entry is not protected from invalidation 1 = Entry is protected from invalidation
2:3	SEL	Selects MPU for access: 00 = Reserved, no access 01 = Reserved, no access 10 = Select MPU 11 = Reserved, no access
4	—	Reserved
5	RO	Read-Only When set to 1, the entry is no longer writable by software. Once set, this bit will remain set until a processor reset occurs. 0 = This MPU entry is writable 1 = This MPU entry is Read-only

Table continues on the next page...

Table 8-43. MAS0—MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
6	DEBUG	Debug Control for Entry This bit is used to assign an entry for debug event use instead of normal access protection use. When used for debug purposes, an MPU debug event is generated when a hit to the entry occurs and the access permissions allow the access. 0 = This MPU entry is used for access protections and allows accesses when matched based on protection attributes 1 = This MPU entry is used for generating a debug event when a match occurs and the access protections would allow access
7	INST	Instruction Entry When SHD = 0, this bit is used to select the INST entry portion of the region descriptor table for **mpure** and **mpuwe** instruction operations. When SHD = 1, this bit is used to indicate whether the entry in the Shared portion of the region descriptor table is assigned as an entry for either instruction access or data access matching. Only one of these access types is monitored for matching by a given shared region descriptor entry. 0 = This MPU entry is used for matching data accesses only 1 = This MPU entry is used for matching instruction accesses only
8	SHD	Shared Entry Select This bit is used to control selection of the Shared portion of the region descriptor table. 0 = The shared portion of the region descriptor table is not accessed on a **mpure** or **mpuwe** operation. Either the instruction portion or the data portion is accessed based on the setting of INST. The entry within the selected portion is based on ESEL. 1 = The shared portion of the region descriptor table is accessed on a **mpure** or **mpuwe** operation. The instruction and data portions are not accessed. The INST bit is used to indicate whether the entry in the Shared portion of the region descriptor table is assigned as an entry for instruction access or data access matching. Only one of these access types is monitored for matching by a given shared region descriptor entry. ESEL defines which shared entry is selected.
9:11	—	Reserved
12:15	ESEL	Entry select for MPU. This field is used to select an entry for reading or writing in conjunction with the settings of SHD and INST. Only valid entry numbers should be used in this field, otherwise the result of an operation is boundedly undefined.
16	—	Reserved

Table continues on the next page...

Table 8-43. MAS0—MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
17:19	UAMSK	Upper Address Mask Control This field is used to support masking of upper address bits before performing range comparisons. Masked bits will be forced to 0 for the purposes of the range compare. Range upper and lower bounds should be set accordingly. 000 = No upper address bits are masked, address matching uses all 32 address bits for accesses 001 = The most significant access address bit is forced to zero before matching 010 = The two most significant access address bits are forced to zero before matching 011 = The three most significant access address bits are forced to 0 before matching 100 = The four most significant access address bits are forced to 0 before matching 101 = The upper five access address bits are forced to zero before matching 11x = Reserved, do not use
20	UW	User Mode Write PermissionDetermines User mode Write (W) permission when INST = 0. Ignored when INST = 1. 0 = No User mode Write permission 1 = User mode has Write permission
21	SW	Supervisor Mode Write / Read Permission Determines Supervisor mode Write (W) permission when INST = 0. Ignored when INST = 1. 0 = No Supervisor mode Write permission 1 = Supervisor mode has Write permission
22	UX/UR	User Mode Execute / Read Permission Determines User mode Execute (X) permission when INST = 1, or User mode Read (R) permission when INST = 0. 0 = No User mode Execute/ Read permission 1 = User mode has Execute/ Read permission
23	SX/SR	Supervisor Mode Execute / Read Permission Determines Supervisor mode Execute (X) permission when INST = 1, or Supervisor mode Read (R) permission when INST = 0. 0 = No Supervisor mode Execute/ Read permission 1 = Supervisor mode has Execute/ Read permission
24	IOVR	Cache-Inhibit attribute OverrideDetermines whether this matching entry overrides the I bit settings of other matching entries and the cache-inhibited region configuration signals 0 = No override of I attribute by entry 1 = Entry I bit overrides I attribute

Table continues on the next page...

Table 8-43. MAS0—MPU Read/Write and Replacement Control (continued)

Bit	Name	Comments, or Function when Set
25	GOVR/—	G attribute Override Determines whether this entry overrides the G bit settings of other matching entries and the Guarded region configuration signals This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1. 0 = No override of G attribute by entry 1 = Entry G bit overrides G attribute
26	—	Reserved
27	—	Reserved
28	I	Cache Inhibited This attribute may be overridden by the cache-inhibited region configuration input settings. 0 = This region is considered cacheable 1 = This region is considered cache-inhibited
29	—	Reserved
30	G/—	Guarded This attribute may be overridden by the guarded region configuration input settings.This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1. 0 = Accesses to this region are not guarded, and can be performed before it is known if they are required by the sequential execution model 1 = All loads and stores to this region are performed without speculation (i.e. they are known to be required)
31	—	Reserved

The MAS1 register is shown in Figure 8-20. Fields are defined in Table 8-44.

0	TID	0	TIDMSK

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

SPR - 625; Read/Write; Reset - Unaffected

Figure 8-20. MPU Assist Register 1 (MAS1)

Table 8-44. MAS1—Descriptor Context and Configuration Control

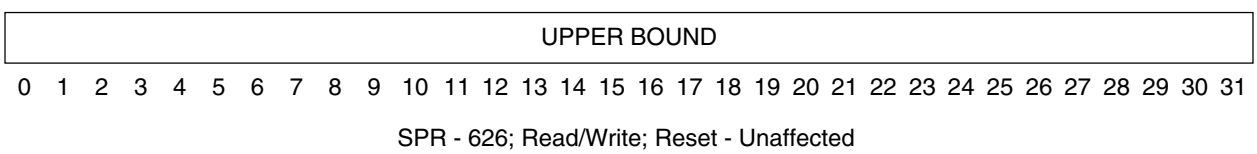
Bit	Name	Comments, or Function when Set
0:7	—	Reserved
8:15	TID	Region ID bits This field is compared with the current process ID of the access address. A TID value of 0 defines an entry as global and matches with all process IDs.
16:23	—	Reserved

Table continues on the next page...

Table 8-44. MAS1—Descriptor Context and Configuration Control (continued)

Bit	Name	Comments, or Function when Set
24:31	TIDMSK	Region ID mask 0xxxxxxx = TID[0] comparison to PID0[0] not masked 1xxxxxxx = TID[0] comparison to PID0[0] is masked x0xxxxxx = TID[1] comparison to PID0[1] not masked x1xxxxxx = TID[1] comparison to PID0[1] is masked xx0xxxxx = TID[2] comparison to PID0[2] not masked xx1xxxxx = TID[2] comparison to PID0[2] is masked xxx0xxxx = TID[3] comparison to PID0[3] not masked xxx1xxxx = TID[3] comparison to PID0[3] is masked xxxx0xxx = TID[4] comparison to PID0[4] not masked xxxx1xxx = TID[4] comparison to PID0[4] is masked xxxxx0xx = TID[5] comparison to PID0[5] not masked xxxxx1xx = TID[5] comparison to PID0[5] is masked xxxxxx0x = TID[6] comparison to PID0[6] not masked xxxxxx1x = TID[6] comparison to PID0[6] is masked xxxxxxx0 = TID[7] comparison to PID0[7] not masked xxxxxxx1 = TID[7] comparison to PID0[7] is masked This field controls whether bits within the TID to PID0 comparison are masked for determining a range match. When a bit is masked, the value of that TID and PID0 bit is ignored for matching purposes.

The MAS2 register is shown in Figure 8-21. Fields are defined in Table 8-45.



SPR - 626; Read/Write; Reset - Unaffected

Figure 8-21. MPU Assist Register 2 (MAS2)

Table 8-45. MAS2—Upper Bound Control

Bit	Name	Comments, or Function when Set
0:31	UPPER BOUND	Upper bound of address range covered by entry. Entry match requires LOWER_BOUND <= EA <= UPPER_BOUND

The MAS3 register is shown in Figure 8-22. Fields are defined in Table 8-46.

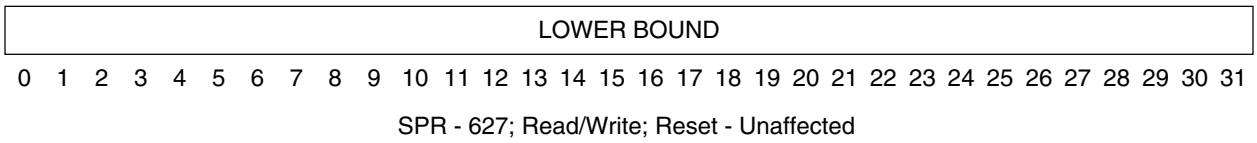


Figure 8-22. MPU Assist Register 3 (MAS3)

Table 8-46. MAS3—Lower Bound Control

Bit	Name	Comments, or Function when Set
0:31	LOWER BOUND	Lower bound of address range covered by entry. Entry match requires LOWER_BOUND <= EA <= UPPER_BOUND

8.9.10 MAS Registers Summary

The MAS registers are summarized in the following figure.

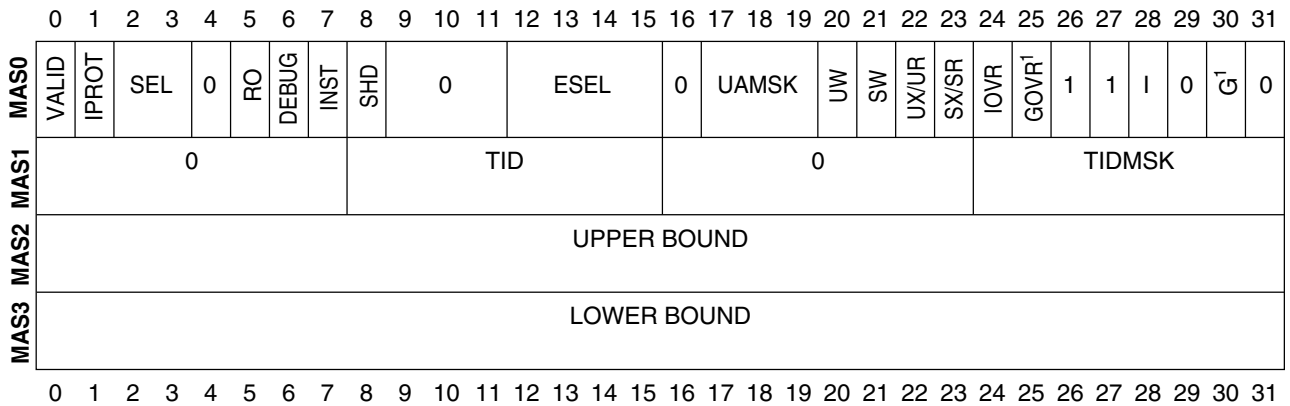


Figure 8-23. MPU Assist Registers Summary

Note

[1] This bit is not implemented in dedicated instruction entries, and is ignored in shared entries with INST = 1

8.10 Local memories

8.10.1 Local Instruction and Data Memory Overview

In order to provide for low latency memory access for critical instruction routines and data operands, local instruction (IMEM) and data (DMEM) memory capabilities have been added. This provides low latency access (zero wait-states) similar to an instruction or data cache, but does not suffer from the overhead of cache miss or cache writethrough operations. Instead, it provides a large-capacity, "tightly coupled" storage capability.

The IMEM and DMEM provide a dedicated port to the CPU's instruction fetch unit and load/store unit respectively. Access via this port is refered to as "frontdoor" access. Each local memory occupies a portion of the address space of the processor, and is conditionally accessed in parallel with the corresponding cache (if present) when a CPU request is made, based on an address decode comparison early in the access pipeline.

In addition to the CPU port, a shared slave port is provided to each local memory to allow external access. Access via this port is refered to as "backdoor" access. This slave port is implemented as an AHB 2.v6 slave interface. External masters use this port to access information from the IMEM and DMEM. The slave port accesses are arbitrated with the CPU port in a round-robin fashion, such that starvation is precluded from occurring on either port.

IMEM frontdoor accesses are always expected during instruction fetches from this local memory. However, when AHB Masters execute read/write accesses from/to IMEM, backdoor AHB slave accesses are expected.

NOTE

DMEM, by default, is always accessed through the frontdoor CPU interface, regardless of whether the program uses Decorated Access, Non-decorated Cache Bypass, Atomic Load and Reserve instructions, as well as regular load/store instructions (stw, lwz, for example). The only exception would be when the Core's DMEMCTL0 register is set to bypass the type of DMEM access used onto AHB backdoor slave interface. When regular load/store instructions are used, they will always access DMEM through the frontdoor CPU interface since no bypass control is available for these kinds of instructions.

8.10.2 Local memory control and configuration

DMEM and IMEM local memory configuration information is provided by a set of privileged read-only SPRs that are accessed using **mfspr** instructions. DMEM and IMEM local memory operation is controlled by a set of privileged device control registers

(DCRs) that are accessed using the **mfdcr** and **mtdcr** instructions. These registers and a description of the operation of various features are described in the following subsections.

8.10.2.1 DMEM Configuration Register 0 (DMEMCFG0)

The DMEM Configuration Register 0 (DMEMCFG0) is a 32-bit read-only register. DMEMCFG0 provides information about the configuration of the e200z425n3 local data memory design. The contents of the DMEMCFG0 register can be read using a **mfspr** instruction. The SPR number for DMEMCFG0 is 694 in decimal. The DMEMCFG0 register is shown in the following figure.

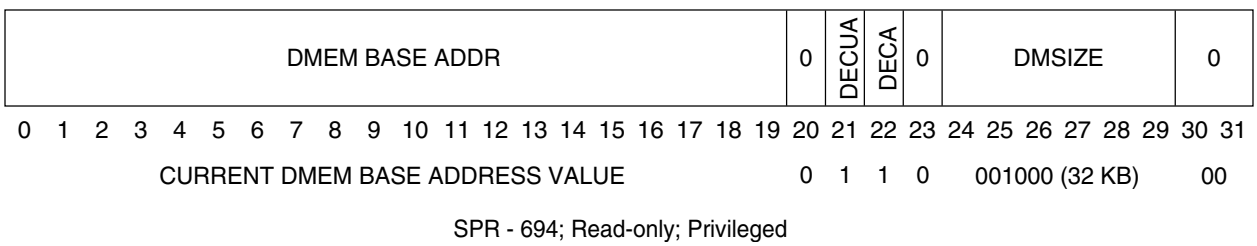


Figure 8-24. DMEM Configuration Register 0 (DMEMCFG0)

The DMEMCFG0 fields are described in the following table.

Table 8-47. DMEMCFG0 field descriptions

Bits	Name	Description
0:19	DMEM BASE ADDR	DMEM BASE ADDRESS (CPU Port) This field defines the current Base Address value being used for the CPU port to the DMEM. This field reflects the value of the external DMEM base address port inputs when the external base address port inputs are enabled via the DBAPD control bit, or the value of the BASE ADDRESS field of DMEMCTL0 when the external base address port inputs are disabled via the DBAPD control bit. **Note:** Low order bits of this field are driven to 0s when the DMEM size is > 4 KB
20	—	Reserved
21	DECUA	DMEM Error Correction Update Available DECUA indicates an error scrubbing function for the DMEM is available. 0 Error Correction Update is not available. 1 Error Correction Update is available for correction of a correctable single-bit error detected on a read access.
22	DECA	DMEM Error Correction Available DECA indicates an error correction function for the DMEM is available. 0 Error Correction is not available. 1 Error Correction is available.
23	—	Reserved

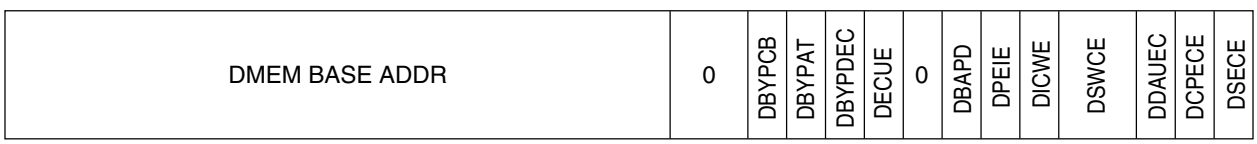
Table continues on the next page...

Table 8-47. DMEMCFG0 field descriptions (continued)

Bits	Name	Description
24:29	DMSIZE	DMEM Size 001000 The size of the DMEM is 32 KB.
30:31	—	Reserved

8.10.2.2 DMEM Control Register 0 (DMEMCTL0)

The DMEM Control Register 0 (DMEMCTL0) controls operation of certain functions of the DMEM logic.



DCR - 496; Read/Write; Reset - 19‘b0 || p_dmem_rstcfg[8,6:7,0] || 4‘b0 || p_dmem_rstcfg[1:5]; Privileged

Figure 8-25. DMEM Control Register 0 (DMEMCTL0)

Table 8-48. DMEMCTL0 field descriptions

Bits	Name	Description
0:16	DMEM BASEADDR	DMEM BASE ADDRESS Field (CPU Port) This field defines the Base Address used for the CPU port to the DMEM when the external base address port inputs are disabled via the DBAPD control bit.
17:18	—	Reserved
19	DBYPCB	DMEM Bypass Cache Bypass CPU accesses DBYPCB can be used to force Non-decorated Cache Bypass loads and Store with Writethrough (**lbcbx**, **lhcbx**, **lwcbx**, and **stwwtx**, **sthwtx**, **stbwtx**) accesses that target the DMEM address space to be presented to the external bus. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses do not bypass the DMEM CPU port. 1 Non-decorated Cache Bypass loads and Store with Writethrough CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface.

Table continues on the next page...

Table 8-48. DMEMCTL0 field descriptions (continued)

Bits	Name	Description
20	DBYPAT	DMEM Bypass Atomic CPU accesses DBYPAT can be used to force atomic (load and reserve, store conditional) accesses to the DMEM address space to be presented to external reservation hardware in systems that support reservation and/or decoration functionality, since no internal reservation hardware is present in the CPU for the DMEM. In general, software must not rely on normal CPU write accesses to clear a reservation, since they are not monitored by reservation hardware; instead only store conditional accesses should be used for this purpose. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Atomic CPU accesses do not bypass the DMEM CPU port. 1 Atomic CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface.
21	DBYPDEC	DMEM Bypass Decorated CPU accesses DBYPDEC can be used to force decorated accesses (**lbdx, lhdx, lwdx, lbdcbx, lhdcbx, lwdcbx, stbdx, sthdx, stwdx, stbdcbx, sthdcbx, stwdcbx, dsn, dsncb**) to the DMEM address space to be presented to external decoration hardware for systems that support this functionality, since no internal decoration hardware is present in the CPU. Note that the protection settings in DMEMCTL1 are still applied to these accesses. 0 Decorated Load and Store CPU accesses do not bypass the DMEM CPU port. 1 Decorated Load and Store CPU accesses bypass the DMEM CPU port and are routed out through the BIU to the DAHB external interface.
22	DECUE	DMEM Error Correction Update Enable DECUE can be used in conjunction with DCPECE and DSECE to provide an error scrubbing function for the DMEM. 0 Error Correction Update is disabled. 1 Error Correction Update is enabled. A correctable single-bit error detected on a read access from either the CPU or the slave port will cause the DMEM to be rewritten with the corrected data if the corresponding error checking enable bit is set (DCPECE for CPU accesses, DSECE for slave port accesses). (Write accesses perform this correction automatically during partial-width writes when error checking is enabled.)
23	—	Reserved
24	DBAPD	DMEM Base Address Port Disable This bit controls usage of the external Base Address port to define the base address of the DMEM for CPU port accesses. It has no effect on DMEM slave port accesses. 0 The external DMEM base address port is used to define the base address of the DMEM for CPU accesses. 1 The external DMEM base address port is disabled for use, and instead the BASE ADDR field value is used to define the base address of the DMEM for CPU accesses.
25	DPEIE	DMEM Processor Error Injection Enable DPEIE will cause injection of errors regardless of the setting of DCPECE, although reporting of errors to the CPU will be masked while DCPECE = 0. 0 Error Injection is disabled. 1 A double-bit error will be injected on each CPU port write into the DMEM data array by inverting the two uppermost parity check bits.

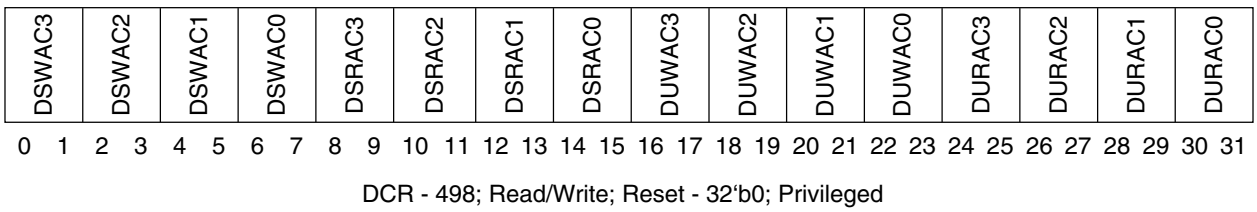
Table continues on the next page...

Table 8-48. DMEMCTL0 field descriptions (continued)

Bits	Name	Description
26	DICWE	DMEM Imprecise CPU Write Enable DICWE may allow for increased partial-width write performance when set. 0 Imprecise writes by the CPU are disabled. All partial-width write ECC errors are reported precisely (error report machine check). 1 Imprecise writes by the CPU are enabled. In certain cases, partial-width write errors may be reported imprecisely (async machine check only).
27:28	DSWCE	DMEM Slave port Write Check/Correct Enable 00 Slave write data is not checked or corrected for errors. 01 Slave write data is checked but not corrected for errors. Detected errors generate a slave port ERROR response and no Write is performed to the DMEM. 10 Slave write data is checked and corrected for errors on all writes. Single-bit correctable errors do not automatically generate an ERROR response, but are instead corrected. 11 Slave write data is checked and corrected for errors on partial-width (1, 2, 3, 5, 6, or 7 byte, or misaligned 4-byte writes) writes. Single-bit correctable errors on partial-width writes do not automatically generate an ERROR response, but are instead corrected. Aligned word or doubleword writes are not checked or corrected for errors.
29	DDAUEC	DMEM Disable Address Use in Error Check This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portions are not used for checkbit/syndrome generation for DMEM CPU or slave port accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation.
30	DCPECE	DMEM CPU Port ECC Enable (CPU port) 0 End-to-End ECC is disabled for the CPU interface. No checking of read data is performed by the CPU. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the CPU interface. Error checking of read data is performed with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a CPU port access cause a machine check to be generated.
31	DSECE	DMEM Slave port Error Checking Enable (Slave port) 0 End-to-End ECC checking logic is disabled for the Slave interface. No checking of read data is performed during read-modify-write operations for Slave port partial-width writes. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the Slave Port interface. Error checking of read data is performed with single-bit error correction during read-modify-write operations for partial-width writes. Detection of uncorrectable single- or multi-bit errors on a Slave port partial-width write access causes a bus error ERROR response to be generated.

8.10.2.3 DMEM Control Register 1 (DMEMCTL1)

The DMEM Control Register 1 (DMEMCTL1) provides protection functions for the DMEM. Separate Supervisor-mode and User-mode read and write access controls allow accesses to be granted, denied, or conditionally granted independently for four equally sized blocks of DMEM. Conditional granting of access permissions causes the MPU memory protection functions to be employed. All other settings override MPU settings. The DMEMCTL1 settings are applied to all accesses that target DMEM address range, regardless of whether access may bypass the DMEM based on settings in DMEMCTL0$_{DBYPCB, DBYPATF, DBYPDEC}$.



DCR - 498; Read/Write; Reset - 32'b0; Privileged

Figure 8-26. DMEM Control Register 1 (DMEMCTL1)

Table 8-49. DMEMCTL1 field descriptions

Bits	Name	Description
0:1	DSWAC3	DMEM Supervisor Write Access Control for Quadrant 3 00 Supervisor-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC3 is used when these two bits = 2'b11
2:3	DSWAC2	DMEM Supervisor Write Access Control for Quadrant 2 00 Supervisor-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 2 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC2 is used when these two bits = 2'b10

Table continues on the next page...

Table 8-49. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
4:5	DSWAC1	DMEM Supervisor Write Access Control for Quadrant 1 00 Supervisor-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC1 is used when these two bits = 2'b01
6:7	DSWAC0	DMEM Supervisor Write Access Control for Quadrant 0 00 Supervisor-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 Supervisor-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 Supervisor-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSWAC0 is used when these two bits = 2'b00
8:9	DSRAC3	DMEM Supervisor Read Access Control for Quadrant 3 00 Supervisor-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC3 is used when these two bits = 2'b11
10:11	DSRAC2	DMEM Supervisor Read Access Control for Quadrant 2 00 Supervisor-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC2 is used when these two bits = 2'b10

Table continues on the next page...

Table 8-49. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
12:13	DSRAC1	DMEM Supervisor Read Access Control for Quadrant 1 00 Supervisor-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC1 is used when these two bits = 2'b01
14:15	DSRAC0	DMEM Supervisor Read Access Control for Quadrant 0 00 Supervisor-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 Supervisor-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 Supervisor-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DSRAC0 is used when these two bits = 2'b00
16:17	DUWAC3	DMEM User Write Access Control for Quadrant 3 00 User-mode CPU write accesses are allowed to quadrant 3 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 3 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC3 is used when these two bits = 2'b11.
18:19	DUWAC2	DMEM User Write Access Control for Quadrant 2 00 User-mode CPU write accesses are allowed to quadrant 2 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 2 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC2 is used when these two bits = 2'b10

Table continues on the next page...

Table 8-49. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
20:21	DUWAC1	DMEM User Write Access Control for Quadrant 1 00 = User-mode CPU write accesses are allowed to quadrant 1 of DMEM 01 = User-mode CPU write accesses are not allowed to quadrant 1 of DMEM 10 = User-mode CPU write accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 = Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC1 is used when these two bits = 2'b01
22:23	DUWAC0	DMEM User Write Access Control for Quadrant 0 00 User-mode CPU write accesses are allowed to quadrant 0 of DMEM 01 User-mode CPU write accesses are not allowed to quadrant 0 of DMEM 10 User-mode CPU write accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DUWAC0 is used when these two bits = 2'b00
24:25	DURAC3	DMEM User Read Access Control for Quadrant 3 00 User-mode CPU read accesses are allowed to quadrant 3 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 3 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 3 of DMEM based on memory protection logic 11 Reserved **Note:** Note: Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC3 is used when these two bits = 2'b11.
26:27	DURAC2	DMEM User Read Access Control for Quadrant 2 00 User-mode CPU read accesses are allowed to quadrant 2 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 2 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 2 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC2 is used when these two bits = 2'b10

Table continues on the next page...

Table 8-49. DMEMCTL1 field descriptions (continued)

Bits	Name	Description
28:29	DURAC1	DMEM User Read Access Control for Quadrant 1 00 User-mode CPU read accesses are allowed to quadrant 1 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 1 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 1 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC1 is used when these two bits = 2'b01
30:31	DURAC0	DMEM User Read Access Control for Quadrant 0 00 User-mode CPU read accesses are allowed to quadrant 0 of DMEM 01 User-mode CPU read accesses are not allowed to quadrant 0 of DMEM 10 User-mode CPU read accesses are conditionally allowed to quadrant 0 of DMEM based on memory protection logic 11 Reserved **Note:** Individual control fields are provided for four equal sized sections of DMEM regardless of DMEM size. A single control field is used for each access based on the upper two address bits used to index into the DMEM. DURAC0 is used when these two bits = 2'b00

8.10.2.4 IMEM Configuration Register 0 (IMEMCFG0)

The IMEM Configuration Register 0 (IMEMCFG0) is a 32-bit read-only register. IMEMCFG0 provides information about the configuration of the e200z425n3 local instruction memory design. The contents of the IMEMCFG0 register can be read using a **mfspr** instruction. The SPR number for IMEMCFG0 is 695 in decimal. The IMEMCFG0 register is shown in the following figure.

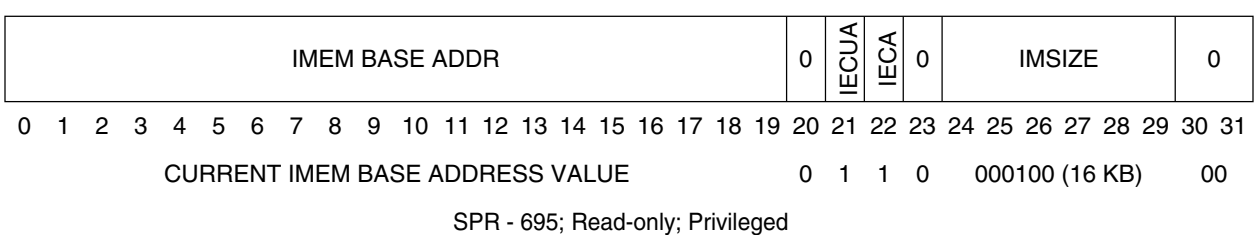


Figure 8-27. IMEM Configuration Register 0 (IMEMCFG0)

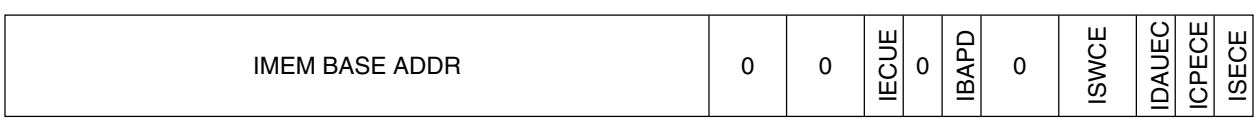
The IMEMCFG0 fields are described in the following table.

Table 8-50. IMEMCFG0 field descriptions

Bits	Name	Description
0:19	IMEM BASE ADDR	IMEM BASE ADDRESS (CPU Port) This field defines the current Base Address value being used for the CPU port to the IMEM. This field reflects the value of the external IMEM base address port inputs when the external base address port inputs are enabled via the DBAPD control bit, or the value of the BASE ADDRESS field of IMEMCTL0 when the external base address port inputs are disabled via the DBAPD control bit. **Note:** Low order bits of this field are driven to 0's when the IMEM size is > 4 KB
20	—	Reserved
21	IECUA	IMEM Error Correction Update Available IECUA indicates an error scrubbing function for the IMEM is available. 0 Error Correction Update is not available 1 Error Correction Update is available for correction of a correctable single-bit error detected on a read access
22	IECA	IMEM Error Correction Available IECA indicates an error correction function for the IMEM is available. 0 Error Correction is not available 1 Error Correction is available
23	—	Reserved
24:29	IMEM Size	IMSIZE - IMEM Size 000100 The size of the IMEM is 16 KB
30:31	—	Reserved

8.10.2.5 IMEM Control Register 0 (IMEMCTL0)

The IMEM Control Register 0 (IMEMCTL0) controls operation of certain functions of the IMEM logic.



DCR - 497; Read/Write; Reset - 22‘b0 ||| p_imem_rstcfg[0] || 4‘b0 || p_imem_rstcfg[1:5]; Privileged

Figure 8-28. IMEM Control Register 0 (IMEMCTL0)

Table 8-51. IMEMCTL0 field descriptions

Bits	Name	Description
0:17	IMEM BASE ADDR	IMEM BASE ADDRESS Field (CPU Port) This field defines the Base Address used for the CPU port to the IMEM when the external base address port inputs are disabled via the BAPD control bit. **Note:** Changes to this value and to the BAPD control bit must be performed carefully by software to ensure coherency is maintained. Specifically, software must ensure that the BTB is invalidated, and that no cached entries are present in the I-Cache for the memory space to be occupied by the IMEM.
18:19	—	Reserved for Base Address extension
20:21	—	Reserved
22	IECUE	IMEM Error Correction Update Enable IECUE can be used to provide an error scrubbing function. 0 Error Correction Update is disabled 1 Error Correction Update is enabled. A correctable single-bit error detected on CPU instruction fetches or slave port read cycles will cause the IMEM to be re-written with the corrected data if the corresponding error checking enable bit is set (ICPECE for CPU accesses, ISECE for slave port accesses). IECUE can be used in conjunction with ICPECE and ISECE to provide an error scrubbing function.
23	—	Reserved
24	IBAPD	IMEM Base Address Port Disable This bit controls usage of the external Base Address port to define the base address of the IMEM for CPU port accesses. It has no effect on IMEM slave port accesses. 0 The external IMEM base address port is used to define the base address of the IMEM for CPU accesses. 1 The external IMEM base address port is disabled for use, and instead the BASE ADDR field value is used to define the base address of the IMEM for CPU accesses.
25:26	—	Reserved
27:28	ISWCE	IMEM Slave port Write Check/Correct Enable 00 Slave write data is not checked or corrected for errors. 01 Slave write data is checked but not corrected for errors. Detected errors generate a slave port ERROR response and no Write is performed to the IMEM. 10 Slave write data is checked and corrected for errors on all writes. Single-bit correctable errors do not automatically generate an ERROR response, but are instead corrected. 11 Slave write data is checked and corrected for errors on partial-width (1, 2, 3, 4, 5, 6, or 7 byte) writes. Single-bit correctable errors on partial-width writes do not automatically generate an ERROR response, but are instead corrected.
29	IDAUEC	IMEM Disable Address Use in Error Check This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portions are not used. for checkbit/syndrome generation for IMEM CPU or slave port accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation.

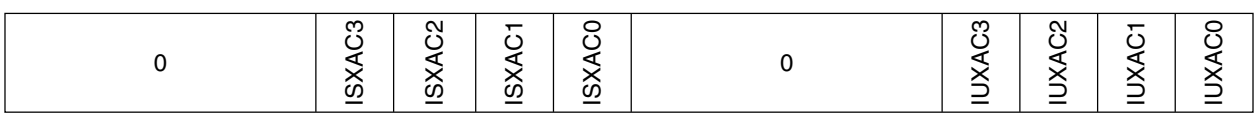
Table continues on the next page...

Table 8-51. IMEMCTL0 field descriptions (continued)

Bits	Name	Description
30	ICPECE	IMEM CPU Port ECC Enable (CPU port) 0 End-to-End ECC is disabled for the CPU interface. No checking of read data is performed by the CPU. Writes will still generate the proper check bit values to be written. 1End-to-End ECC is enabled for performing ECC on the CPU interface. Error checking of read data is performed with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a CPU port access cause a machine check to be generated.
31	ISECE	IMEM Slave port Error Checking Enable (Slave port) 0 End-to-End ECC checking logic is disabled for the Slave interface. No checking of read data is performed during read-modify-write operations for partial-width writes. Writes will still generate the proper check bit values to be written. 1 End-to-End ECC is enabled for performing ECC on the Slave Port interface. Error checking of read data is performed during read-modify-write operations for partial-width writes with single-bit error correction. Detection of uncorrectable single- or multi-bit errors on a Slave port access causes a bus error ERROR response to be generated.

8.10.2.6 IMEM Control Register 1 (IMEMCTL1)

The IMEM Control Register 1 (IMEMCTL1) provides protection functions for the IMEM. Separate Supervisor-mode and User-mode instruction fetch access controls allow accesses to be granted, denied, or conditionally granted independently for four equally sized blocks of IMEM. Conditional granting of access permissions causes the MPU memory protection functions to be employed, all other settings override MPU settings.



DCR - 499; Read/Write; Reset - 32‘b0; Privileged

Figure 8-29. IMEM Control Register 1 (IMEMCTL1)

Table 8-52. IMEMCTL1 field descriptions

Bits	Name	Description
0:7	—	Reserved

Table continues on the next page...

Table 8-52. IMEMCTL1 field descriptions (continued)

Bits	Name	Description
8:9	ISXAC3	IMEM Supervisor Instruction Fetch Access Control for Quadrant 3 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 3 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 3 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 3 of IMEM based on memory protection logic 11 Reserved Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC3 is used when these two bits = 2'b11
10:11	ISXAC2	IMEM Supervisor Instruction Fetch Access Control for Quadrant 2 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 2 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 2 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 2 of IMEM based on memory protection logic 11 Reserved Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC2 is used when these two bits = 2'b10
12:13	ISXAC1	IMEM Supervisor Instruction Fetch Access Control for Quadrant 1 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 1 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 1 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 1 of IMEM based on memory protection logic 11 Reserved Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC1 is used when these two bits = 2'b01
14:15	ISXAC0	IMEM Supervisor Instruction Fetch Access Control for Quadrant 0 00 Supervisor-mode CPU instruction fetch accesses are allowed to quadrant 0 of IMEM 01 Supervisor-mode CPU instruction fetch accesses are not allowed to quadrant 0 of IMEM 10 Supervisor-mode CPU instruction fetch accesses are conditionally allowed to quadrant 0 of IMEM based on memory protection logic 11 Reserved Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. ISXAC0 is used when these two bits = 2'b00

Table continues on the next page...

Table 8-52. IMEMCTL1 field descriptions (continued)

Bits	Name	Description
16:23	—	Reserved
24:25	IUXAC3	IMEM User Instruction Fetch Access Control for Quadrant 3 00 User-mode CPU instruction fetch accesses are allowed to quadrant 3 of IMEM 01 User-mode CPU instruction fetch accesses are not allowed to quadrant 3 of IMEM 10 User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 3 of IMEM based on memory protection logic 11 Reserved Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC3 is used when these two bits = 2'b11
26:27	IUXAC2	IMEM User Instruction Fetch Access Control for Quadrant 2 00 User-mode CPU instruction fetch accesses are allowed to quadrant 2 of IMEM 01 User-mode CPU instruction fetch accesses are not allowed to quadrant 2 of IMEM 10 User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 2 of IMEM based on memory protection logic 11 Reserved Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC2 is used when these two bits = 2'b10
28:29	IUXAC1	MEM User Instruction Fetch Access Control for Quadrant 1 00 User-mode CPU instruction fetch accesses are allowed to quadrant 1 of IMEM 01 User-mode CPU instruction fetch accesses are not allowed to quadrant 1 of IMEM 10 User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 1 of IMEM based on memory protection logic 11 Reserved Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC1 is used when these two bits = 2'b01
30:31	IUXAC0	IMEM User Instruction Fetch Access Control for Quadrant 0 00 = User-mode CPU instruction fetch accesses are allowed to quadrant 0 of IMEM 01 = User-mode CPU instruction fetch accesses are not allowed to quadrant 0 of IMEM 10 = User-mode CPU instruction fetch accesses are conditionally allowed to quadrant 0 of IMEM based on memory protection logic 11 = Reserved Note: Individual control fields are provided for four equal sized sections of IMEM regardless of IMEM size. A single control field is used for each access based on the upper two address bits used to index into the IMEM. IUXAC0 is used when these two bits = 2'b00

8.11 End-to-End ECC Support

This section describes end-to-end error detection and correction capability (e2eECC) enhancements to address additional diagnostic capabilities for the next generation of embedded designs.

8.11.1 End-to-End ECC control and configuration

e2eECC is controlled by a set of privileged device control registers (DCRs) accessed using the **mfdcr** and **mtdcr** instructions. These registers and the operation of various features are described in the following subsections.

8.11.1.1 End-to-End ECC Control Register 0 (E2ECTL0)

The End-to-End ECC Control Register 0 (E2ECTL0) controls operation of the e2eECC logic.

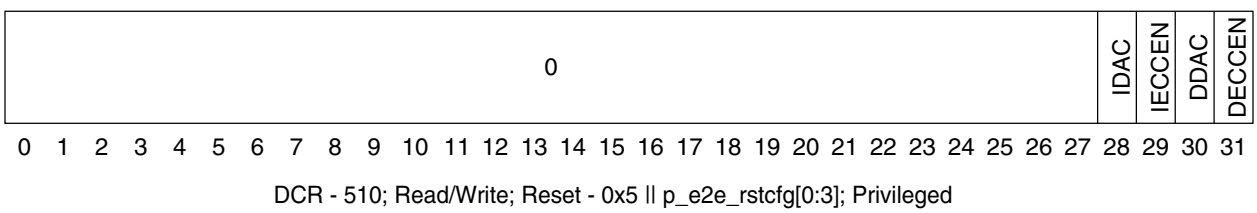


Figure 8-30. e2eECC Control Register 0 (E2ECTL0)

Table 8-53. E2ECTL0 field descriptions

Bits	Name	Description
0:27	—	Reserved
28	IDAC	Instruction Disable Address Checking This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion is not used for checkbit/syndrome generation for Instruction AHB accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation.

Table continues on the next page...

Table 8-53. E2ECTL0 field descriptions (continued)

Bits	Name	Description
29	IECCEN	Instruction ECC Enable Field 0 End-to-End ECC is disabled for the Instruction interface. No checking of instruction fetch data is performed. 1 End-to-End ECC is enabled for performing error detection and correction on the Instruction interface. Checking of instruction fetch data is performed with correction of single-bit data errors. Detection of any address errors or uncorrectable multi-bit data errors cause a machine check to be generated if the instruction fetch information is attempted to be used for instruction decoding and execution.
30	DDAC	Data Disable Address Checking This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion is not used for checkbit/syndrome generation for Data AHB accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation.
31	DECCEN	Data ECC Enable Field 0 End-to-End ECC is disabled for the Data interface. No checking of read data is performed. Writes will still generate the proper check bit values to be supplied to external storage devices. 1 End-to-End ECC is enabled for performing error detection and correction on the Data interface. Checking of read data is performed with correction of single-bit data errors. Detection of any address errors, or uncorrectable multi-bit data errors cause a machine check to be generated. Writes generate the proper check bit values to be supplied to external storage devices.

e2eECC is enabled by setting the [I,D]ECCEN bit of the E2ECTL0 DCR to a 1. On reset, e2eECC operation may be disabled from detecting or correcting errors based on reset configuration inputs, and no exceptions will be signaled for nonzero calculated syndrome values. This allows for the proper initialization of stored checkbits in any volatile storage by the CPU without causing error conditions due to uninitialized storage. Following the initialization, software may enable e2eECC operation by writing the appropriate values to the E2ECTL0[IECCEN] and E2ECTL0[DECCEN] control bits.

The $E2ECTL0_{[I,D]DAC}$ control bits can be used to remove the address checking component of the ECC logic for the Instruction AHB and Data AHB when address inclusion is not desired.

8.11.1.2 End-to-End ECC error generation by software

Error generation provides a way to test error recovery and portions of the error detection/ correction logic by intentionally injecting parity errors into the driven checkbit information on the external bus during write operations. No provision is made to generate

internal errors on read data due to timing issues; instead, software may intentionally generate errors in the checkbit values to emulate data, address, or checkbit error on external bus write cycles, and read back the written data to cause an error to be detected.

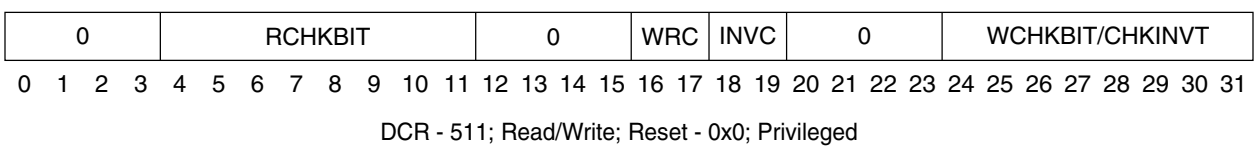
The End-to-End ECC Error Control/Status Register (E2EECSR0) controls operation of the e2eECC error injection logic. It allows for generation of a single error injection operation on the next CPU data write to the external bus only, or for a series of error injection write operations to the external bus, using the WRC and INVC control fields. The WRCHKBIT/CHKINVT field controls which of the outputs are affected.

Control is provided to allow for either inverting the value(s) of one or more of the checkbit outputs in hardware on one or more external write accesses, or for software to directly supply checkbit values to be used for the external write access(es).

The End-to-End ECC Error Control/Status Register (E2EECSR0) also supports access to raw checkbit values via the RCHKBIT status field. This field is updated with checkbit values received on each CPU data read operation to either the DMEM or to the external bus.

Note that Nexus 3 accesses do not cause error injection or the capture of read checkbits, regardless of the settings of the E2EECSR0 register.

The following figure shows the layout of E2EECSR0.



DCR - 511; Read/Write; Reset - 0x0; Privileged

Figure 8-31. e2eECC Error Control/Status Register 0 (E2EECSR0)

Table 8-54. E2EECSR0 field descriptions

Bits	Name	Description
0:3	—	Reserved
4:11	RCHKBIT	Read Checkbits This field provides the raw checkbits received on the last CPU data access to the DMEM or to external memory via the Data BIU. Software may use this information to determine the stored checkbits of external memories or the DMEM by performing CPU load operations and then accessing this field prior to the next load operation (assuming interrupts are masked).
12:15	—	Reserved

Table continues on the next page...

Table 8-54. E2EECSR0 field descriptions (continued)

Bits	Name	Description
16:17	WRC	Write Control 00 No write checkbit substitution is performed by this control bit 01 Checkbit substitution is performed for the next CPU external write access (only). Software must clear this field before rewriting to 01 in order to generate a subsequent checkbit substitution operation. 10 Checkbit substitution is performed for each CPU external write access while this field remains set to 10. This field must be cleared by software to discontinue further checkbit substitution. 11 Reserved Note: Checkbit substitution has priority over Error injection
18:19	INVC	Invert Control 00 No error injection is performed by this control bit 01 Error injection is performed for the next CPU external write access (only). Software must clear this field before rewriting to 01 in order to generate a subsequent error injection operation. 10 Error injection is performed for each CPU external write access while this field remains set to 10. This field must be cleared by software to discontinue further error generation. 11 Reserved Note: Checkbit substitution has priority over Error injection
20:23	—	Reserved
24:31	WCHKBIT/ CHKINVT	Write Checkbits / Checkbit Invert Mask This field provides the checkbit substitution values when performing checkbit substitution via the WRC control field, and controls which checkbits are inverted when performing error injection via the INVC control field. For Checkbit inversion operations via error injection: 0 Checkbit is driven normally 1 Checkbit is inverted before being driven out when error injection occurs.

Chapter 9
System Integration Unit Lite2 (SIUL2)

9.1 System Integration Unit Lite2 (SIUL2) chip-specific module information

9.1.1 SIUL2 interrupts

The SIUL2 supports 11 external interrupts on this device.

The 11 external interrupt sources are grouped into 2 sets, one with 8 sources and one with 3 sources. Each of the groups is assigned one interrupt vector and mapped to the INTC. Each external interrupt source can be masked, and its status can be read individually. Glitch filters are implemented for each external interrupt source.

9.1.2 Reset values of MIDRn

The reset value of MIDR1 is 0x5746_xxxx. The values represented by x vary by orderable part number. See the register's detailed description for the values corresponding to your part.

For the MPC5746R, the reset value of MIDR2 is 0x4000_5200. For devices in the family with different memory sizes, see the register's detailed description for the values corresponding to the device.

9.2 Introduction

9.2.1 Overview

The System Integration Unit Lite2 provides control over all the electrical pin controls and up to 32 ports with 16 bits of bidirectional, general-purpose input and output signals. One of the most important functions of the SIUL2 is that it enables the user to select the functions and electrical characteristics that appear on external device pins. It also controls the multiplexing of internal signals from one module to another and controls device I/O. It supports 11 external interrupts with trigger event configuration. The following figure is the block diagram of SIUL2 and its interfaces to other system components.

The module provides dedicated pad control to general-purpose pads that can be configured as either inputs or outputs. The SIUL2 module provides registers that enable user software to read values from GPIO pads configured as inputs, and write values to GPIO pads configured as outputs:

- When configured as output, you can write to an internal register to control the state driven on the associated output pad.
- When configured as input, you can detect the state of the associated pad by reading the value from an internal register.
- When configured as input and output, the pad value can be read back, which can be used as a method of checking if the written value appeared on the pad.

To assist software development, GPIO data registers can be accessed using various mechanisms. These differing mechanisms allow support for port access or for bit manipulation without the need to use read-modify-write operations or access with Power Architecture reservation.

- Access to two 16-bit ports in one access
- Read/write access to a single bit
- A 16-bit port write with a bit mask, using single 32-bit access.

The external interrupt sources can be configured at the MCU level to be used with any chip pad. The 11 interrupt sources can be configured to have a digital filter to reject short glitches on the inputs. The external interrupt/DMA requests map to the REQ pins in the device packages.

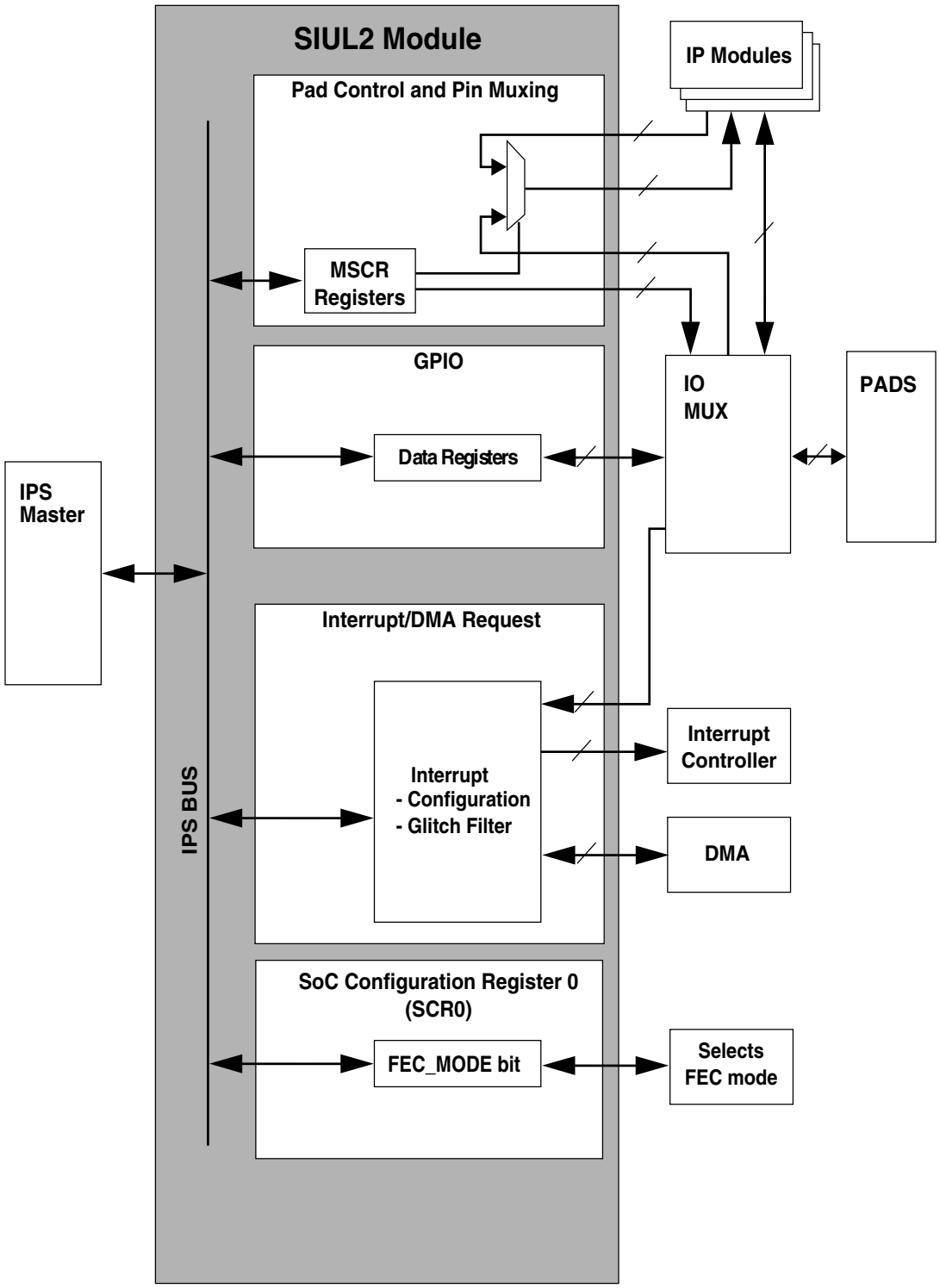


Figure 9-1. System Integration Unit Lite2 block diagram

9.2.2 Features

The SIUL2 supports the following features:

- 1 to 32 GPIO ports with data control
  - Drive data to as many as 16 independent I/O channels
  - Sample data from as many as 16 independent I/O channels

Two 16-bit registers can be read/written with one access for a 32-bit port, if needed.

External interrupt/DMA request support with:

- Two interrupt vectors for 11 interrupt sources with independent interrupt masks
- 11 programmable digital glitch filters, one for each REQ pin
- 8 DMA request channels for 8 REQ pins
- Edge detection

NOTE

SIUL2 supports 11 interrupt requests and 8 DMA requests. The 8 DMA request corresponds to REQ 0 - REQ 7 which are controlled by bits DIRSR0-DIRSR7 in register SIUL2 DMA/ Interrupt Request Select Register(SIUL2_DIRSR0).

Additionally the SIUL2 contains the Multiplexed Signal Configuration Registers that configure the electrical parameters and settings for 180 functional pads. The number of these registers that is actually implemented varies by device. These registers configure the following pad features:

- Drive strength
- Output impedance control
- Open drain/source output enable
- Slew rate control
- Hysteresis control
- Internal weak pull control and pull selection
- Pin function assignment
- Control of analog path switches
- Safe mode behavior configuration

9.2.3 Register protection

The System Integration Unit Lite2 uses the Register Protection Scheme to protect the individual registers from accidental writes. The following registers can be protected:

- SIUL2 DMA/Interrupt Request Enable Register 0 (SIUL2_DIRER0)
- SIUL2 DMA/Interrupt Request Select Register 0 (SIUL2_DIRSR0)
- SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)
- SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)
- SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)
- SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR0 to SIUL2_IFMCR31)
- SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)
- SIUL2 Multiplexed Single Configuration Register (SIUL2_MSCR0 to SIUL2_MSCR1023): Only the specific MSCRs that are actually implemented on a device are protected.

The Register Protection chapter provides a complete list of protected registers for this device.

9.3 Memory map and register description

This section provides a detailed description of all registers accessible in the SIUL2 module in address order. Each description includes a standard register diagram. Details of register bit and field function follow the register diagrams, in bit order.

The following table lists the implemented SIUL2 registers.

NOTE

Undocumented register spaces in the SIUL2 memory map, including addresses shown as "blank," are reserved. Reserved registers read as 0, and writes to these registers have no effect. Transfer errors are generated for attempts to access a completely reserved register space.

9.3.1 Memory map

Table 9-1. SIUL2 memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
4	SIUL2 MCU ID Register #1 (SIUL2_MIDR1)	32	R	See section
8	SIUL2 MCU ID Register #2 (SIUL2_MIDR2)	32	R	See section
10	SIUL2 DMA/Interrupt Status Flag Register0 (SIUL2_DISR0)	32	w1c	0000_0000h
18	SIUL2 DMA/Interrupt Request Enable Register0 (SIUL2_DIRER0)	32	R/W	0000_0000h
20	SIUL2 DMA/Interrupt Request Select Register0 (SIUL2_DIRSR0)	32	R/W	0000_0000h
28	SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)	32	R/W	0000_0000h
30	SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)	32	R/W	0000_0000h
38	SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)	32	R/W	0000_0000h
40	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR0)	32	R/W	0000_0000h
44	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR1)	32	R/W	0000_0000h
48	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR2)	32	R/W	0000_0000h
4C	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR3)	32	R/W	0000_0000h
50	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR4)	32	R/W	0000_0000h
54	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR5)	32	R/W	0000_0000h
58	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR6)	32	R/W	0000_0000h
5C	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR7)	32	R/W	0000_0000h
60	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR8)	32	R/W	0000_0000h
64	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR9)	32	R/W	0000_0000h
68	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR10)	32	R/W	0000_0000h
C0	SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)	32	R/W	0000_0000h
100	SIUL2 SoC Configuration Register0 (SIUL2_SCR0)	16	R/W	0000h
240	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR0)	32	R/W	See section
244	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR1)	32	R/W	See section
248	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR2)	32	R/W	See section
24C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR3)	32	R/W	See section
250	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR4)	32	R/W	See section
254	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR5)	32	R/W	See section
258	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR6)	32	R/W	See section

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
25C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR7)	32	R/W	See section
260	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR8)	32	R/W	See section
264	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR9)	32	R/W	See section
268	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR10)	32	R/W	See section
26C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR11)	32	R/W	See section
270	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR12)	32	R/W	See section
274	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR13)	32	R/W	See section
280	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR16)	32	R/W	See section
284	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR17)	32	R/W	See section
2C0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR32)	32	R/W	See section
2C4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR33)	32	R/W	See section
2C8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR34)	32	R/W	See section
2CC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR35)	32	R/W	See section
2D0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR36)	32	R/W	See section
2D4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR37)	32	R/W	See section
2D8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR38)	32	R/W	See section
2DC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR39)	32	R/W	See section
2E0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR40)	32	R/W	See section
2E4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR41)	32	R/W	See section
2E8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR42)	32	R/W	See section
2EC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR43)	32	R/W	See section
2F0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR44)	32	R/W	See section
2F4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR45)	32	R/W	See section
300	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR48)	32	R/W	See section

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
304	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR49)	32	R/W	See section
308	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR50)	32	R/W	See section
30C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR51)	32	R/W	See section
310	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR52)	32	R/W	See section
314	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR53)	32	R/W	See section
318	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR54)	32	R/W	See section
31C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR55)	32	R/W	See section
320	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR56)	32	R/W	See section
324	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR57)	32	R/W	See section
328	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR58)	32	R/W	See section
32C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR59)	32	R/W	See section
330	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR60)	32	R/W	See section
334	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR61)	32	R/W	See section
338	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR62)	32	R/W	See section
33C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR63)	32	R/W	See section
340	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR64)	32	R/W	See section
380	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR80)	32	R/W	See section
384	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR81)	32	R/W	See section
388	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR82)	32	R/W	See section
38C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR83)	32	R/W	See section
390	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR84)	32	R/W	See section
394	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR85)	32	R/W	See section

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
398	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR86)	32	R/W	See section
39C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR87)	32	R/W	See section
3A0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR88)	32	R/W	See section
3A4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR89)	32	R/W	See section
3A8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR90)	32	R/W	See section
3AC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR91)	32	R/W	See section
3B0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR92)	32	R/W	See section
3B4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR93)	32	R/W	See section
3C4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR97)	32	R/W	See section
3C8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR98)	32	R/W	See section
3CC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR99)	32	R/W	See section
3D0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR100)	32	R/W	See section
3D4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR101)	32	R/W	See section
3D8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR102)	32	R/W	See section
3DC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR103)	32	R/W	See section
3E4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR105)	32	R/W	See section
3E8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR106)	32	R/W	See section
3EC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR107)	32	R/W	See section
3F0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR108)	32	R/W	See section
3F4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR109)	32	R/W	See section
3F8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR110)	32	R/W	See section
3FC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR111)	32	R/W	See section

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
400	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR112)	32	R/W	See section
404	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR113)	32	R/W	See section
408	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR114)	32	R/W	See section
40C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR115)	32	R/W	See section
410	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR116)	32	R/W	See section
414	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR117)	32	R/W	See section
418	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR118)	32	R/W	See section
41C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR119)	32	R/W	See section
420	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR120)	32	R/W	See section
424	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR121)	32	R/W	See section
428	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR122)	32	R/W	See section
42C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR123)	32	R/W	See section
430	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR124)	32	R/W	See section
434	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR125)	32	R/W	See section
438	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR126)	32	R/W	See section
43C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR127)	32	R/W	See section
440	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR128)	32	R/W	See section
444	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR129)	32	R/W	See section
448	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR130)	32	R/W	See section
44C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR131)	32	R/W	See section
450	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR132)	32	R/W	See section
454	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR133)	32	R/W	See section

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
480	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR144)	32	R/W	See section
484	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR145)	32	R/W	See section
488	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR146)	32	R/W	See section
48C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR147)	32	R/W	See section
490	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR148)	32	R/W	See section
494	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR149)	32	R/W	See section
498	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR150)	32	R/W	See section
49C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR151)	32	R/W	See section
4A0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR152)	32	R/W	See section
4A4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR153)	32	R/W	See section
4A8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR154)	32	R/W	See section
4AC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR155)	32	R/W	See section
4B0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR156)	32	R/W	See section
4B4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR157)	32	R/W	See section
4B8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR158)	32	R/W	See section
4BC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR159)	32	R/W	See section
4C0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR160)	32	R/W	See section
4C4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR161)	32	R/W	See section
4C8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR162)	32	R/W	See section
4D0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR164)	32	R/W	See section
4D4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR165)	32	R/W	See section
4DC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR167)	32	R/W	See section

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
4E0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR168)	32	R/W	See section
4E4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR169)	32	R/W	See section
4E8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR170)	32	R/W	See section
4EC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR171)	32	R/W	See section
4F0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR172)	32	R/W	See section
4F4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR173)	32	R/W	See section
4F8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR174)	32	R/W	See section
500	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR176)	32	R/W	See section
504	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR177)	32	R/W	See section
540	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR192)	32	R/W	See section
544	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR193)	32	R/W	See section
548	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR194)	32	R/W	See section
54C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR195)	32	R/W	See section
580	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR208)	32	R/W	See section
584	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR209)	32	R/W	See section
588	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR210)	32	R/W	See section
58C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR211)	32	R/W	See section
590	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR212)	32	R/W	See section
594	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR213)	32	R/W	See section
598	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR214)	32	R/W	See section
59C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR215)	32	R/W	See section
5A0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR216)	32	R/W	See section

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
5A4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR217)	32	R/W	See section
5A8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR218)	32	R/W	See section
5AC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR219)	32	R/W	See section
5B0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR220)	32	R/W	See section
5B4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR221)	32	R/W	See section
5B8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR222)	32	R/W	See section
5BC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR223)	32	R/W	See section
5C0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR224)	32	R/W	See section
5C4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR225)	32	R/W	See section
5C8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR226)	32	R/W	See section
5CC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR227)	32	R/W	See section
5D0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR228)	32	R/W	See section
5D4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR229)	32	R/W	See section
5D8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR230)	32	R/W	See section
5DC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR231)	32	R/W	See section
5E0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR232)	32	R/W	See section
5E4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR233)	32	R/W	See section
5E8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR234)	32	R/W	See section
5EC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR235)	32	R/W	See section
5F0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR236)	32	R/W	See section
5F4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR237)	32	R/W	See section
5F8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR238)	32	R/W	See section

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
5FC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR239)	32	R/W	See section
600	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR240)	32	R/W	See section
604	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR241)	32	R/W	See section
608	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR242)	32	R/W	See section
60C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR243)	32	R/W	See section
610	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR244)	32	R/W	See section
614	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR245)	32	R/W	See section
618	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR246)	32	R/W	See section
61C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR247)	32	R/W	See section
620	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR248)	32	R/W	See section
624	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR249)	32	R/W	See section
628	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR250)	32	R/W	See section
62C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR251)	32	R/W	See section
630	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR252)	32	R/W	See section
634	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR253)	32	R/W	See section
638	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR254)	32	R/W	See section
63C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR255)	32	R/W	See section
A40	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR512)	32	R/W	See section
A44	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR513)	32	R/W	See section
A48	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR514)	32	R/W	See section
A4C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR515)	32	R/W	See section
A50	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR516)	32	R/W	See section

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
A54	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR517)	32	R/W	See section
A58	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR518)	32	R/W	See section
A5C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR519)	32	R/W	See section
A60	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR520)	32	R/W	See section
A64	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR521)	32	R/W	See section
A68	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR522)	32	R/W	See section
A6C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR523)	32	R/W	See section
A70	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR524)	32	R/W	See section
A74	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR525)	32	R/W	See section
A78	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR526)	32	R/W	See section
A7C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR527)	32	R/W	See section
A80	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR528)	32	R/W	See section
A84	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR529)	32	R/W	See section
A88	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR530)	32	R/W	See section
A8C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR531)	32	R/W	See section
A90	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR532)	32	R/W	See section
A94	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR533)	32	R/W	See section
A98	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR534)	32	R/W	See section
A9C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR535)	32	R/W	See section
AA0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR536)	32	R/W	See section
AA4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR537)	32	R/W	See section
AA8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR538)	32	R/W	See section

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
AAC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR539)	32	R/W	See section
AB0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR540)	32	R/W	See section
AB4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR541)	32	R/W	See section
AB8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR542)	32	R/W	See section
ABC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR543)	32	R/W	See section
AC0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR544)	32	R/W	See section
AC4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR545)	32	R/W	See section
AC8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR546)	32	R/W	See section
ACC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR547)	32	R/W	See section
AD0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR548)	32	R/W	See section
AD4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR549)	32	R/W	See section
AD8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR550)	32	R/W	See section
ADC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR551)	32	R/W	See section
AE0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR552)	32	R/W	See section
AE4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR553)	32	R/W	See section
AE8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR554)	32	R/W	See section
AEC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR555)	32	R/W	See section
AF0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR556)	32	R/W	See section
AF4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR557)	32	R/W	See section
AF8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR558)	32	R/W	See section
AFC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR559)	32	R/W	See section
B00	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR560)	32	R/W	See section

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
B04	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR561)	32	R/W	See section
B08	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR562)	32	R/W	See section
B0C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR563)	32	R/W	See section
B10	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR564)	32	R/W	See section
B14	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR565)	32	R/W	See section
B18	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR566)	32	R/W	See section
B1C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR567)	32	R/W	See section
B20	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR568)	32	R/W	See section
B24	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR569)	32	R/W	See section
B28	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR570)	32	R/W	See section
B2C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR571)	32	R/W	See section
B30	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR572)	32	R/W	See section
B34	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR573)	32	R/W	See section
B38	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR574)	32	R/W	See section
B3C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR575)	32	R/W	See section
B40	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR576)	32	R/W	See section
B44	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR577)	32	R/W	See section
B48	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR578)	32	R/W	See section
B4C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR579)	32	R/W	See section
B50	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR580)	32	R/W	See section
B54	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR581)	32	R/W	See section
B58	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR582)	32	R/W	See section

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
B5C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR583)	32	R/W	See section
B60	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR584)	32	R/W	See section
B64	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR585)	32	R/W	See section
B78	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR590)	32	R/W	See section
B7C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR591)	32	R/W	See section
B80	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR592)	32	R/W	See section
B84	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR593)	32	R/W	See section
B88	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR594)	32	R/W	See section
B8C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR595)	32	R/W	See section
B90	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR596)	32	R/W	See section
B94	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR597)	32	R/W	See section
B98	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR598)	32	R/W	See section
B9C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR599)	32	R/W	See section
BA0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR600)	32	R/W	See section
BA4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR601)	32	R/W	See section
BA8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR602)	32	R/W	See section
BAC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR603)	32	R/W	See section
BB0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR604)	32	R/W	See section
BB4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR605)	32	R/W	See section
BB8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR606)	32	R/W	See section
BBC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR607)	32	R/W	See section
BC0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR608)	32	R/W	See section

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
BC4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR609)	32	R/W	See section
BE0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR616)	32	R/W	See section
BE4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR617)	32	R/W	See section
BE8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR618)	32	R/W	See section
BEC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR619)	32	R/W	See section
BF0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR620)	32	R/W	See section
BF4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR621)	32	R/W	See section
BF8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR622)	32	R/W	See section
BFC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR623)	32	R/W	See section
C00	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR624)	32	R/W	See section
C04	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR625)	32	R/W	See section
C08	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR626)	32	R/W	See section
C0C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR627)	32	R/W	See section
C10	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR628)	32	R/W	See section
C14	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR629)	32	R/W	See section
C18	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR630)	32	R/W	See section
C1C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR631)	32	R/W	See section
C20	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR632)	32	R/W	See section
C24	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR633)	32	R/W	See section
C28	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR634)	32	R/W	See section
C2C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR635)	32	R/W	See section
C40	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR640)	32	R/W	See section

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
C44	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR641)	32	R/W	See section
C48	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR642)	32	R/W	See section
C4C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR643)	32	R/W	See section
C50	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR644)	32	R/W	See section
C54	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR645)	32	R/W	See section
C58	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR646)	32	R/W	See section
C5C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR647)	32	R/W	See section
C60	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR648)	32	R/W	See section
C64	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR649)	32	R/W	See section
C68	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR650)	32	R/W	See section
C6C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR651)	32	R/W	See section
C70	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR652)	32	R/W	See section
C74	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR653)	32	R/W	See section
C78	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR654)	32	R/W	See section
C7C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR655)	32	R/W	See section
C80	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR656)	32	R/W	See section
C84	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR657)	32	R/W	See section
C88	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR658)	32	R/W	See section
C8C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR659)	32	R/W	See section
C90	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR660)	32	R/W	See section
C94	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR661)	32	R/W	See section
C98	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR662)	32	R/W	See section

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
C9C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR663)	32	R/W	See section
CA0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR664)	32	R/W	See section
CA4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR665)	32	R/W	See section
CA8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR666)	32	R/W	See section
CAC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR667)	32	R/W	See section
CB0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR668)	32	R/W	See section
CB4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR669)	32	R/W	See section
CB8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR670)	32	R/W	See section
CBC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR671)	32	R/W	See section
CC0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR672)	32	R/W	See section
CC4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR673)	32	R/W	See section
CC8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR674)	32	R/W	See section
CCC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR675)	32	R/W	See section
CD0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR676)	32	R/W	See section
CD4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR677)	32	R/W	See section
CD8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR678)	32	R/W	See section
CDC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR679)	32	R/W	See section
CE0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR680)	32	R/W	See section
CE4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR681)	32	R/W	See section
CE8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR682)	32	R/W	See section
CEC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR683)	32	R/W	See section
CF0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR684)	32	R/W	See section

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
CF4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR685)	32	R/W	See section
CF8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR686)	32	R/W	See section
CFC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR687)	32	R/W	See section
D00	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR688)	32	R/W	See section
D04	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR689)	32	R/W	See section
D08	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR690)	32	R/W	See section
D0C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR691)	32	R/W	See section
D10	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR692)	32	R/W	See section
D14	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR693)	32	R/W	See section
D18	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR694)	32	R/W	See section
D30	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR700)	32	R/W	See section
D34	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR701)	32	R/W	See section
D38	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR702)	32	R/W	See section
D3C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR703)	32	R/W	See section
D60	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR712)	32	R/W	See section
D64	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR713)	32	R/W	See section
D68	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR714)	32	R/W	See section
D6C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR715)	32	R/W	See section
D70	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR716)	32	R/W	See section
D74	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR717)	32	R/W	See section
E00	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR752)	32	R/W	See section
E04	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR753)	32	R/W	See section

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
E08	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR754)	32	R/W	See section
E0C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR755)	32	R/W	See section
E10	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR756)	32	R/W	See section
E14	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR757)	32	R/W	See section
E48	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR770)	32	R/W	See section
E4C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR771)	32	R/W	See section
E50	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR772)	32	R/W	See section
E54	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR773)	32	R/W	See section
E5C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR775)	32	R/W	See section
E60	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR776)	32	R/W	See section
E64	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR777)	32	R/W	See section
E68	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR778)	32	R/W	See section
ECC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR803)	32	R/W	See section
EE4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR809)	32	R/W	See section
EF0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR812)	32	R/W	See section
EF4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR813)	32	R/W	See section
EFC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR815)	32	R/W	See section
F00	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR816)	32	R/W	See section
F60	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR840)	32	R/W	See section
F64	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR841)	32	R/W	See section
F68	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR842)	32	R/W	See section
F6C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR843)	32	R/W	See section

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
F70	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR844)	32	R/W	See section
F74	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR845)	32	R/W	See section
F78	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR846)	32	R/W	See section
F7C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR847)	32	R/W	See section
F80	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR848)	32	R/W	See section
F84	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR849)	32	R/W	See section
F88	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR850)	32	R/W	See section
F8C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR851)	32	R/W	See section
F90	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR852)	32	R/W	See section
F94	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR853)	32	R/W	See section
F98	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR854)	32	R/W	See section
F9C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR855)	32	R/W	See section
FA0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR856)	32	R/W	See section
FA4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR857)	32	R/W	See section
FA8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR858)	32	R/W	See section
FAC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR859)	32	R/W	See section
FB0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR860)	32	R/W	See section
FB4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR861)	32	R/W	See section
FB8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR862)	32	R/W	See section
FBC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR863)	32	R/W	See section
FC0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR864)	32	R/W	See section
FC4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR865)	32	R/W	See section

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
FC8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR866)	32	R/W	See section
FCC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR867)	32	R/W	See section
FD0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR868)	32	R/W	See section
FD4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR869)	32	R/W	See section
FD8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR870)	32	R/W	See section
FDC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR871)	32	R/W	See section
FE0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR872)	32	R/W	See section
FE4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR873)	32	R/W	See section
FE8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR874)	32	R/W	See section
FEC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR875)	32	R/W	See section
FF0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR876)	32	R/W	See section
FF4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR877)	32	R/W	See section
FF8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR878)	32	R/W	See section
FFC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR879)	32	R/W	See section
1000	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR880)	32	R/W	See section
1004	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR881)	32	R/W	See section
1008	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR882)	32	R/W	See section
100C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR883)	32	R/W	See section
1010	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR884)	32	R/W	See section
1014	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR885)	32	R/W	See section
1018	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR886)	32	R/W	See section
101C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR887)	32	R/W	See section

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
1020	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR888)	32	R/W	See section
1024	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR889)	32	R/W	See section
1028	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR890)	32	R/W	See section
102C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR891)	32	R/W	See section
1030	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR892)	32	R/W	See section
1034	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR893)	32	R/W	See section
1038	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR894)	32	R/W	See section
103C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR895)	32	R/W	See section
1040	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR896)	32	R/W	See section
1044	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR897)	32	R/W	See section
1048	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR898)	32	R/W	See section
104C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR899)	32	R/W	See section
1050	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR900)	32	R/W	See section
1054	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR901)	32	R/W	See section
1058	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR902)	32	R/W	See section
105C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR903)	32	R/W	See section
1060	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR904)	32	R/W	See section
1064	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR905)	32	R/W	See section
1068	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR906)	32	R/W	See section
106C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR907)	32	R/W	See section
1070	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR908)	32	R/W	See section
1074	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR909)	32	R/W	See section

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
1078	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR910)	32	R/W	See section
107C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR911)	32	R/W	See section
1080	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR912)	32	R/W	See section
1084	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR913)	32	R/W	See section
1088	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR914)	32	R/W	See section
108C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR915)	32	R/W	See section
1090	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR916)	32	R/W	See section
1094	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR917)	32	R/W	See section
1098	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR918)	32	R/W	See section
109C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR919)	32	R/W	See section
10A0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR920)	32	R/W	See section
10A4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR921)	32	R/W	See section
10A8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR922)	32	R/W	See section
10AC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR923)	32	R/W	See section
10B0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR924)	32	R/W	See section
10B4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR925)	32	R/W	See section
10B8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR926)	32	R/W	See section
10BC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR927)	32	R/W	See section
10C0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR928)	32	R/W	See section
10C4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR929)	32	R/W	See section
10C8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR930)	32	R/W	See section
10CC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR931)	32	R/W	See section

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
10D0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR932)	32	R/W	See section
10D4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR933)	32	R/W	See section
10D8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR934)	32	R/W	See section
10DC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR935)	32	R/W	See section
1140	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR960)	32	R/W	See section
1144	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR961)	32	R/W	See section
1148	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR962)	32	R/W	See section
1160	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR968)	32	R/W	See section
1164	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR969)	32	R/W	See section
1168	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR970)	32	R/W	See section
11C0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR992)	32	R/W	See section
11C4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR993)	32	R/W	See section
11C8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR994)	32	R/W	See section
11CC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR995)	32	R/W	See section
11D0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR996)	32	R/W	See section
11D8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR998)	32	R/W	See section
1240	General Purpose Register 0 (GPR0)	32	R/W	See section
1244	General Purpose Register 1 (GPR1)	32	R/W	See section
1248	General Purpose Register 2 (GPR2)	32	R/W	See section
1300	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO0)	8	R/W	00h
1301	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO1)	8	R/W	00h
1302	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO2)	8	R/W	00h
1303	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO3)	8	R/W	00h
1304	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO4)	8	R/W	00h
1305	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO5)	8	R/W	00h
1306	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO6)	8	R/W	00h
1307	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO7)	8	R/W	00h

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
1308	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO8)	8	R/W	00h
1309	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO9)	8	R/W	00h
130A	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO10)	8	R/W	00h
130B	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO11)	8	R/W	00h
130C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO12)	8	R/W	00h
130D	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO13)	8	R/W	00h
1310	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO16)	8	R/W	00h
1311	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO17)	8	R/W	00h
1320	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO32)	8	R/W	00h
1321	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO33)	8	R/W	00h
1322	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO34)	8	R/W	00h
1323	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO35)	8	R/W	00h
1324	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO36)	8	R/W	00h
1325	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO37)	8	R/W	00h
1326	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO38)	8	R/W	00h
1327	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO39)	8	R/W	00h
1328	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO40)	8	R/W	00h
1329	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO41)	8	R/W	00h
132A	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO42)	8	R/W	00h
132B	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO43)	8	R/W	00h
132C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO44)	8	R/W	00h
132D	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO45)	8	R/W	00h
1330	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO48)	8	R/W	00h
1331	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO49)	8	R/W	00h
1332	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO50)	8	R/W	00h
1333	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO51)	8	R/W	00h
1334	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO52)	8	R/W	00h
1335	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO53)	8	R/W	00h
1336	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO54)	8	R/W	00h
1337	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO55)	8	R/W	00h
1338	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO56)	8	R/W	00h
1339	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO57)	8	R/W	00h
133A	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO58)	8	R/W	00h
133B	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO59)	8	R/W	00h
133C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO60)	8	R/W	00h
133D	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO61)	8	R/W	00h
133E	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO62)	8	R/W	00h
133F	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO63)	8	R/W	00h
1340	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO64)	8	R/W	00h

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
1350	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO80)	8	R/W	00h
1351	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO81)	8	R/W	00h
1352	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO82)	8	R/W	00h
1353	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO83)	8	R/W	00h
1354	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO84)	8	R/W	00h
1355	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO85)	8	R/W	00h
1356	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO86)	8	R/W	00h
1357	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO87)	8	R/W	00h
1358	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO88)	8	R/W	00h
1359	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO89)	8	R/W	00h
135A	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO90)	8	R/W	00h
135B	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO91)	8	R/W	00h
135C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO92)	8	R/W	00h
135D	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO93)	8	R/W	00h
1361	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO97)	8	R/W	00h
1362	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO98)	8	R/W	00h
1363	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO99)	8	R/W	00h
1364	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO100)	8	R/W	00h
1365	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO101)	8	R/W	00h
1366	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO102)	8	R/W	00h
1367	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO103)	8	R/W	00h
1369	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO105)	8	R/W	00h
136A	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO106)	8	R/W	00h
136B	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO107)	8	R/W	00h
136C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO108)	8	R/W	00h
136D	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO109)	8	R/W	00h
136E	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO110)	8	R/W	00h
136F	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO111)	8	R/W	00h
1370	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO112)	8	R/W	00h
1371	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO113)	8	R/W	00h
1372	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO114)	8	R/W	00h
1373	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO115)	8	R/W	00h
1374	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO116)	8	R/W	00h
1375	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO117)	8	R/W	00h
1376	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO118)	8	R/W	00h
1377	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO119)	8	R/W	00h
1378	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO120)	8	R/W	00h
1379	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO121)	8	R/W	00h
137A	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO122)	8	R/W	00h

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
137B	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO123)	8	R/W	00h
137C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO124)	8	R/W	00h
137D	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO125)	8	R/W	00h
137E	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO126)	8	R/W	00h
137F	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO127)	8	R/W	00h
1380	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO128)	8	R/W	00h
1381	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO129)	8	R/W	00h
1382	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO130)	8	R/W	00h
1383	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO131)	8	R/W	00h
1384	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO132)	8	R/W	00h
1385	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO133)	8	R/W	00h
1390	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO144)	8	R/W	00h
1391	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO145)	8	R/W	00h
1392	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO146)	8	R/W	00h
1393	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO147)	8	R/W	00h
1394	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO148)	8	R/W	00h
1395	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO149)	8	R/W	00h
1396	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO150)	8	R/W	00h
1397	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO151)	8	R/W	00h
1398	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO152)	8	R/W	00h
1399	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO153)	8	R/W	00h
139A	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO154)	8	R/W	00h
139B	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO155)	8	R/W	00h
139C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO156)	8	R/W	00h
139D	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO157)	8	R/W	00h
139E	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO158)	8	R/W	00h
139F	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO159)	8	R/W	00h
13A0	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO160)	8	R/W	00h
13A1	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO161)	8	R/W	00h
13A2	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO162)	8	R/W	00h
13A4	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO164)	8	R/W	00h
13A5	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO165)	8	R/W	00h
13A7	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO167)	8	R/W	00h
13A8	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO168)	8	R/W	00h
13A9	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO169)	8	R/W	00h
13AA	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO170)	8	R/W	00h
13AB	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO171)	8	R/W	00h
13AC	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO172)	8	R/W	00h
13AD	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO173)	8	R/W	00h

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
13AE	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO174)	8	R/W	00h
13B0	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO176)	8	R/W	00h
13B1	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO177)	8	R/W	00h
13C0	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO192)	8	R/W	00h
13C1	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO193)	8	R/W	00h
13C2	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO194)	8	R/W	00h
13C3	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO195)	8	R/W	00h
13D0	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO208)	8	R/W	00h
13D1	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO209)	8	R/W	00h
13D2	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO210)	8	R/W	00h
13D3	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO211)	8	R/W	00h
13D4	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO212)	8	R/W	00h
13D5	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO213)	8	R/W	00h
13D6	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO214)	8	R/W	00h
13D7	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO215)	8	R/W	00h
13D8	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO216)	8	R/W	00h
13D9	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO217)	8	R/W	00h
13DA	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO218)	8	R/W	00h
13DB	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO219)	8	R/W	00h
13DC	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO220)	8	R/W	00h
13DD	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO221)	8	R/W	00h
13DE	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO222)	8	R/W	00h
13DF	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO223)	8	R/W	00h
13E0	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO224)	8	R/W	00h
13E1	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO225)	8	R/W	00h
13E2	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO226)	8	R/W	00h
13E3	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO227)	8	R/W	00h
13E4	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO228)	8	R/W	00h
13E5	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO229)	8	R/W	00h
13E6	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO230)	8	R/W	00h
13E7	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO231)	8	R/W	00h
13E8	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO232)	8	R/W	00h
13E9	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO233)	8	R/W	00h
13EA	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO234)	8	R/W	00h
13EB	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO235)	8	R/W	00h
13EC	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO236)	8	R/W	00h
13ED	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO237)	8	R/W	00h
13EE	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO238)	8	R/W	00h
13EF	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO239)	8	R/W	00h

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
13F0	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO240)	8	R/W	00h
13F1	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO241)	8	R/W	00h
13F2	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO242)	8	R/W	00h
13F3	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO243)	8	R/W	00h
13F4	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO244)	8	R/W	00h
13F5	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO245)	8	R/W	00h
13F6	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO246)	8	R/W	00h
13F7	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO247)	8	R/W	00h
13F8	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO248)	8	R/W	00h
13F9	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO249)	8	R/W	00h
13FA	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO250)	8	R/W	00h
13FB	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO251)	8	R/W	00h
13FC	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO252)	8	R/W	00h
13FD	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO253)	8	R/W	00h
13FE	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO254)	8	R/W	00h
13FF	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO255)	8	R/W	00h
1500	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI0)	8	R	00h
1501	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI1)	8	R	00h
1502	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI2)	8	R	00h
1503	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI3)	8	R	00h
1504	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI4)	8	R	00h
1505	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI5)	8	R	00h
1506	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI6)	8	R	00h
1507	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI7)	8	R	00h
1508	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI8)	8	R	00h
1509	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI9)	8	R	00h
150A	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI10)	8	R	00h
150B	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI11)	8	R	00h
150C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI12)	8	R	00h
150D	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI13)	8	R	00h
1510	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI16)	8	R	00h
1511	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI17)	8	R	00h
1520	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI32)	8	R	00h
1521	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI33)	8	R	00h
1522	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI34)	8	R	00h
1523	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI35)	8	R	00h
1524	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI36)	8	R	00h
1525	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI37)	8	R	00h
1526	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI38)	8	R	00h

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
1527	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI39)	8	R	00h
1528	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI40)	8	R	00h
1529	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI41)	8	R	00h
152A	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI42)	8	R	00h
152B	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI43)	8	R	00h
152C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI44)	8	R	00h
152D	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI45)	8	R	00h
1530	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI48)	8	R	00h
1531	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI49)	8	R	00h
1532	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI50)	8	R	00h
1533	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI51)	8	R	00h
1534	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI52)	8	R	00h
1535	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI53)	8	R	00h
1536	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI54)	8	R	00h
1537	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI55)	8	R	00h
1538	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI56)	8	R	00h
1539	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI57)	8	R	00h
153A	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI58)	8	R	00h
153B	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI59)	8	R	00h
153C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI60)	8	R	00h
153D	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI61)	8	R	00h
153E	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI62)	8	R	00h
153F	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI63)	8	R	00h
1540	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI64)	8	R	00h
1550	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI80)	8	R	00h
1551	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI81)	8	R	00h
1552	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI82)	8	R	00h
1553	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI83)	8	R	00h
1554	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI84)	8	R	00h
1555	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI85)	8	R	00h
1556	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI86)	8	R	00h
1557	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI87)	8	R	00h
1558	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI88)	8	R	00h
1559	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI89)	8	R	00h
155A	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI90)	8	R	00h
155B	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI91)	8	R	00h
155C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI92)	8	R	00h
155D	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI93)	8	R	00h
1561	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI97)	8	R	00h

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
1562	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI98)	8	R	00h
1563	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI99)	8	R	00h
1564	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI100)	8	R	00h
1565	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI101)	8	R	00h
1566	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI102)	8	R	00h
1567	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI103)	8	R	00h
1569	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI105)	8	R	00h
156A	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI106)	8	R	00h
156B	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI107)	8	R	00h
156C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI108)	8	R	00h
156D	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI109)	8	R	00h
156E	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI110)	8	R	00h
156F	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI111)	8	R	00h
1570	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI112)	8	R	00h
1571	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI113)	8	R	00h
1572	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI114)	8	R	00h
1573	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI115)	8	R	00h
1574	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI116)	8	R	00h
1575	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI117)	8	R	00h
1576	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI118)	8	R	00h
1577	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI119)	8	R	00h
1578	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI120)	8	R	00h
1579	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI121)	8	R	00h
157A	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI122)	8	R	00h
157B	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI123)	8	R	00h
157C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI124)	8	R	00h
157D	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI125)	8	R	00h
157E	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI126)	8	R	00h
157F	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI127)	8	R	00h
1580	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI128)	8	R	00h
1581	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI129)	8	R	00h
1582	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI130)	8	R	00h
1583	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI131)	8	R	00h
1584	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI132)	8	R	00h
1585	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI133)	8	R	00h
1590	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI144)	8	R	00h
1591	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI145)	8	R	00h
1592	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI146)	8	R	00h
1593	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI147)	8	R	00h

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
1594	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI148)	8	R	00h
1595	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI149)	8	R	00h
1596	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI150)	8	R	00h
1597	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI151)	8	R	00h
1598	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI152)	8	R	00h
1599	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI153)	8	R	00h
159A	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI154)	8	R	00h
159B	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI155)	8	R	00h
159C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI156)	8	R	00h
159D	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI157)	8	R	00h
159E	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI158)	8	R	00h
159F	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI159)	8	R	00h
15A0	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI160)	8	R	00h
15A1	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI161)	8	R	00h
15A2	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI162)	8	R	00h
15A4	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI164)	8	R	00h
15A5	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI165)	8	R	00h
15A7	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI167)	8	R	00h
15A8	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI168)	8	R	00h
15A9	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI169)	8	R	00h
15AA	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI170)	8	R	00h
15AB	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI171)	8	R	00h
15AC	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI172)	8	R	00h
15AD	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI173)	8	R	00h
15AE	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI174)	8	R	00h
15B0	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI176)	8	R	00h
15B1	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI177)	8	R	00h
15C0	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI192)	8	R	00h
15C1	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI193)	8	R	00h
15C2	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI194)	8	R	00h
15C3	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI195)	8	R	00h
15D0	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI208)	8	R	00h
15D1	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI209)	8	R	00h
15D2	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI210)	8	R	00h
15D3	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI211)	8	R	00h
15D4	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI212)	8	R	00h
15D5	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI213)	8	R	00h
15D6	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI214)	8	R	00h
15D7	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI215)	8	R	00h

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
15D8	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI216)	8	R	00h
15D9	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI217)	8	R	00h
15DA	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI218)	8	R	00h
15DB	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI219)	8	R	00h
15DC	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI220)	8	R	00h
15DD	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI221)	8	R	00h
15DE	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI222)	8	R	00h
15DF	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI223)	8	R	00h
15E0	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI224)	8	R	00h
15E1	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI225)	8	R	00h
15E2	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI226)	8	R	00h
15E3	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI227)	8	R	00h
15E4	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI228)	8	R	00h
15E5	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI229)	8	R	00h
15E6	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI230)	8	R	00h
15E7	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI231)	8	R	00h
15E8	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI232)	8	R	00h
15E9	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI233)	8	R	00h
15EA	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI234)	8	R	00h
15EB	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI235)	8	R	00h
15EC	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI236)	8	R	00h
15ED	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI237)	8	R	00h
15EE	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI238)	8	R	00h
15EF	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI239)	8	R	00h
15F0	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI240)	8	R	00h
15F1	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI241)	8	R	00h
15F2	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI242)	8	R	00h
15F3	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI243)	8	R	00h
15F4	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI244)	8	R	00h
15F5	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI245)	8	R	00h
15F6	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI246)	8	R	00h
15F7	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI247)	8	R	00h
15F8	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI248)	8	R	00h
15F9	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI249)	8	R	00h
15FA	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI250)	8	R	00h
15FB	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI251)	8	R	00h
15FC	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI252)	8	R	00h
15FD	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI253)	8	R	00h
15FE	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI254)	8	R	00h

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
15FF	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI255)	8	R	00h
1700	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO0)	16	R/W	0000h
1702	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO1)	16	R/W	0000h
1704	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO2)	16	R/W	0000h
1706	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO3)	16	R/W	0000h
1708	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO4)	16	R/W	0000h
170A	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO5)	16	R/W	0000h
170C	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO6)	16	R/W	0000h
170E	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO7)	16	R/W	0000h
1710	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO8)	16	R/W	0000h
1712	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO9)	16	R/W	0000h
1714	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO10)	16	R/W	0000h
1716	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO11)	16	R/W	0000h
1718	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO12)	16	R/W	0000h
171A	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO13)	16	R/W	0000h
171C	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO14)	16	R/W	0000h
171E	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO15)	16	R/W	0000h
1740	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI0)	16	R	0000h
1742	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI1)	16	R	0000h
1744	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI2)	16	R	0000h
1746	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI3)	16	R	0000h
1748	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI4)	16	R	0000h
174A	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI5)	16	R	0000h
174C	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI6)	16	R	0000h
174E	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI7)	16	R	0000h
1750	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI8)	16	R	0000h
1752	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI9)	16	R	0000h
1754	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI10)	16	R	0000h
1756	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI11)	16	R	0000h
1758	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI12)	16	R	0000h
175A	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI13)	16	R	0000h
175C	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI14)	16	R	0000h
175E	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI15)	16	R	0000h
1780	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO0)	32	W	0000_0000h
1784	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO1)	32	W	0000_0000h
1788	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO2)	32	W	0000_0000h

Table continues on the next page...

Table 9-1. SIUL2 memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value
178C	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO3)	32	W	0000_0000h
1790	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO4)	32	W	0000_0000h
1794	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO5)	32	W	0000_0000h
1798	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO6)	32	W	0000_0000h
179C	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO7)	32	W	0000_0000h
17A0	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO8)	32	W	0000_0000h
17A4	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO9)	32	W	0000_0000h
17A8	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO10)	32	W	0000_0000h
17AC	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO11)	32	W	0000_0000h
17B0	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO12)	32	W	0000_0000h
17B4	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO13)	32	W	0000_0000h
17B8	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO14)	32	W	0000_0000h
17BC	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO15)	32	W	0000_0000h

The following table lists the implemented SIUL2 registers.

NOTE

Undocumented register spaces in the SIUL2 memory map, including addresses shown as "blank," are reserved. Reserved registers read as 0, and writes to these registers have no effect. Transfer errors are generated for attempts to access a completely reserved register space.

9.3.2 SIUL2 MCU ID Register #1 (SIUL2_MIDR1)

This register holds identification information about the device.

NOTE

This register supports only 32-bit accesses. Byte and half-word write accesses are not supported.

NOTE

The reset value depends on the chip's package and mask version.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PARTNUM															
W																
Reset	0	1	0	1	0	1	1	1	0	1	0	0	0	1	1	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ED	PKG					0		MAJOR_MASK				MINOR_MASK			
W																
Reset	0	*	*	*	*	*	0	0	*	*	*	*	*	*	*	*

* Notes:
- MINOR_MASK field: Value is set at the factory and increments with each revision of the device.
- MAJOR_MASK field: Value is set at factory and increments with each revision of the device.
- PKG field: Values are set at factory and cannot be modified.

SIUL2_MIDR1 field descriptions

Field	Description
0–15 PARTNUM	MCU Part Number 0x5746 0x5745 0x5743 0x5742 For the full part number, this field must be combined with MIDR2.PARTNUM[23:16]
16 ED	Always reads 0 in the Production Device. This bit will be set when the Production Device is packaged with a Buddy Die in the Emulation Device.
17–21 PKG	Package Settings Can by read by software to determine the package type that is used for the particular device: 0b01000: 292-ball BGA 0b01101: 144-pin LQFP 0b01110: 252-pin BGA 0b10001: 176-pin LQFP All other bit combinations are reserved.
22–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIUL2_MIDR1 field descriptions (continued)

Field	Description
24–27 MAJOR_MASK	Major Mask Revision. 0x1 for Cut2.0B of this device.
28–31 MINOR_MASK	Minor Mask Revision. 0x1 for Cut2.0B of this device.

9.3.3 SIUL2 MCU ID Register #2 (SIUL2_MIDR2)

NOTE

This register supports only 32-bit accesses. Byte and half-word write accesses are not supported.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SF	FLASH_SIZE_1				FLASH_SIZE_2				0						
W																
Reset	0	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PARTNUM								0							
W																
Reset	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0	0

* Notes:
- PARTNUM field: Value is set at factory and cannot be changed.
- FLASH_SIZE_2 field: Value is set at factory and cannot be changed.
- FLASH_SIZE_1 field: Value is set at factory and cannot be changed.

SIUL2_MIDR2 field descriptions

Field	Description
0 SF	Manufacturer 0 Freescale Semiconductor, Inc. 1 Reserved
1–4 FLASH_SIZE_1	Coarse granularity for flash memory size Needs to be combined with FLASH_SIZE_2 to calculate the actual memory size of 4 MB. 0b0110: 1 MB 0b0111: 2 MB 0b1000: 4 MB All other values are reserved.

Table continues on the next page...

SIUL2_MIDR2 field descriptions (continued)

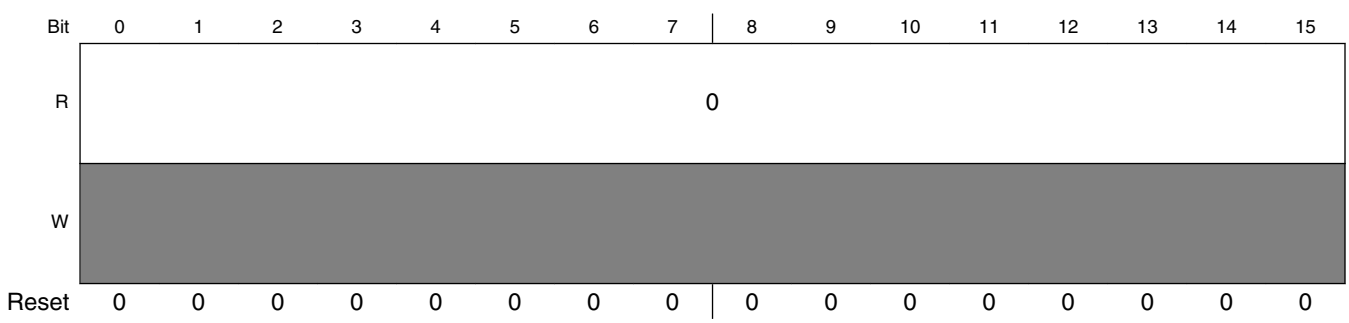
Field	Description
5–8 FLASH_SIZE_2	Fine granularity for flash memory size Needs to be combined with FLASH_SIZE_1 to calculate the actual memory size: 0b0000: 0 x (FLASH_SIZE_1 / 8) 0b0100: 4 x (FLASH_SIZE_1 / 8) All other values are reserved.
9–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 PARTNUM	ASCII character in MCU Part Number 0x52 R All other values are reserved.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

9.3.4 SIUL2 DMA/Interrupt Status Flag Register0 (SIUL2_DISR0)

The DMA/Interrupt Status Register contains flag bits that record an event on the external IRQ pins. When an event as defined in IRQ Rising-Edge Event Enable Register (SIUL2_IREER0) and IRQ Falling-Edge Event Enable Register (SIUL2_IFEER0) occurs, the corresponding flag bit is set. The IRQ Flag bit is set regardless of the state of the corresponding DMA/Interrupt Request Enable bit in DMA/Interrupt Request Enable Register (SIUL2_DIRER0) - Standard. The IRQ Flag bit remains set until cleared by software or through the servicing of a DMA request. The IRQ Flag bits are cleared by writing a '1' to the bits. A write of '0' has no effect. The number of bits implemented is specified by the parameter that defines the number external IRQ pins.

This register supports 8-, 16-, and 32-bit accesses.

Address: 0h base + 10h offset = 10h



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					EIF10	EIF9	EIF8	EIF7	EIF6	EIF5	EIF4	EIF3	EIF2	EIF1	EIF0
W						w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_DISR0 field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 EIF10	External Interrupt Status Flag 10 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER10), EIF10 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER10 and SIUL2_IFEER10 has occurred
22 EIF9	External Interrupt Status Flag 9 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER9), EIF9 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER9 and SIUL2_IFEER9 has occurred
23 EIF8	External Interrupt Status Flag 8 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER8), EIF8 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER8 and SIUL2_IFEER8 has occurred
24 EIF7	External Interrupt Status Flag 7 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER7), EIF7 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER7 and SIUL2_IFEER7 has occurred
25 EIF6	External Interrupt Status Flag 6 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER6), EIF6 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER6 and SIUL2_IFEER6 has occurred
26 EIF5	External Interrupt Status Flag 5 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER5), EIF5 causes an interrupt or DMA request.

Table continues on the next page...

SIUL2_DISR0 field descriptions (continued)

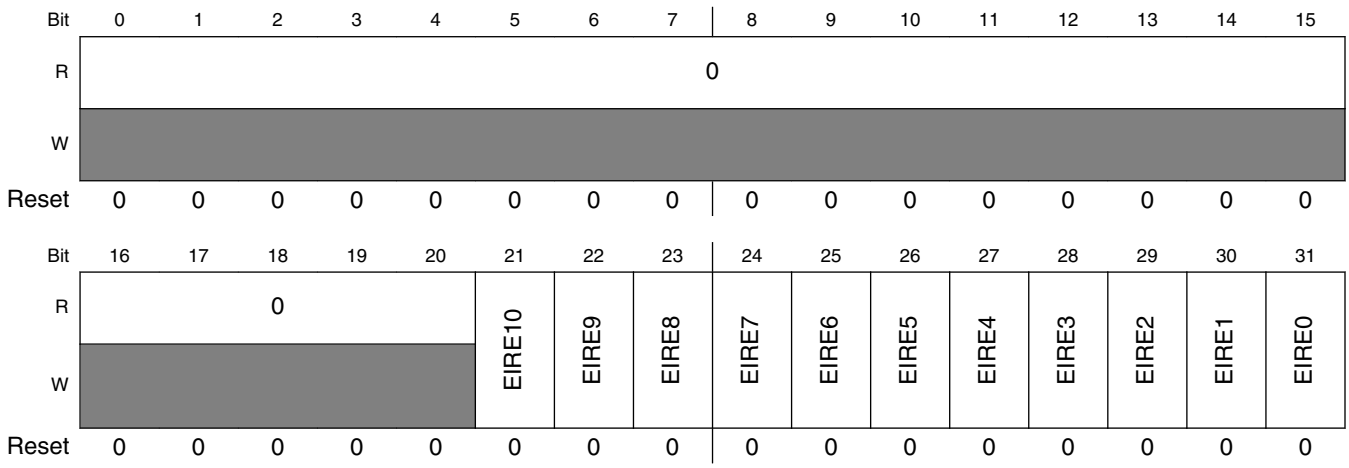
Field	Description
	0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER5 and SIUL2_IFEER5 has occurred
27 EIF4	External Interrupt Status Flag 4 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER4), EIF4 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER4 and SIUL2_IFEER4 has occurred
28 EIF3	External Interrupt Status Flag 3 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER3), EIF3 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER3 and SIUL2_IFEER3 has occurred
29 EIF2	External Interrupt Status Flag 2 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER2), EIF2 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER2 and SIUL2_IFEER2 has occurred
30 EIF1	External Interrupt Status Flag 1 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER1), EIF1 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER1 and SIUL2_IFEER1 has occurred
31 EIF0	External Interrupt Status Flag 0 This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER0), EIF0 causes an interrupt or DMA request. 0 No interrupt or DMA event has occurred on the pad 1 An interrupt or DMA event as defined by SIUL2_IREER0 and SIUL2_IFEER0 has occurred

9.3.5 SIUL2 DMA/Interrupt Request Enable Register0 (SIUL2_DIRER0)

The DMA/Interrupt Request Enable Register enables the assertion of DMA or interrupt request if the corresponding External IRQ Flag bit is set in SIUL2 DMA/Interrupt Status Flag Register0 (SIUL2_DISR0). The type of request enabled is determined by the corresponding DMA/Interrupt Request Select bit in SIUL2 DMA/Interrupt Request Select Register0 (SIUL2_DIRSR0). The number of bits implemented is specified by the parameter that defines the number of external IRQ pins. This register is used to enable the interrupt messaging to the interrupt controller.

This register supports 8-, 16-, and 32-bit accesses.

Address: 0h base + 18h offset = 18h



SIUL2_DIRER0 field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 EIRE10	External Interrupt or DMA Request Enable 10 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
22 EIRE9	External Interrupt or DMA Request Enable 9 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
23 EIRE8	External Interrupt or DMA Request Enable 8 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
24 EIRE7	External Interrupt or DMA Request Enable 7 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
25 EIRE6	External Interrupt or DMA Request Enable 6 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
26 EIRE5	External Interrupt or DMA Request Enable 5 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
27 EIRE4	External Interrupt or DMA Request Enable 4 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
28 EIRE3	External Interrupt or DMA Request Enable 3

Table continues on the next page...

SIUL2_DIRER0 field descriptions (continued)

Field	Description
	0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
29 EIRE2	External Interrupt or DMA Request Enable 2 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
30 EIRE1	External Interrupt or DMA Request Enable 1 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register
31 EIRE0	External Interrupt or DMA Request Enable 0 0 Interrupt or DMA requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes either a DMA or an interrupt request depending on DIRSR register

9.3.6 SIUL2 DMA/Interrupt Request Select Register0 (SIUL2_DIRSR0)

The DIRSR selects between the DMA or interrupt request. If the corresponding bits are set in SIUL2 DMA/Interrupt Status Flag Register0 (SIUL2_DISR0) and SIUL2 DMA/Interrupt Request Select Register0 (SIUL2_DIRSR0), then the DMA/Interrupt Request Select bit determines whether DMA or a interrupt request is asserted. The number of bits implemented is specified by the parameter that defines the number of DMA request outputs.

This register supports 8-, 16-, and 32-bit accesses.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					DIRSR10	DIRSR9	DIRSR8	DIRSR7	DIRSR6	DIRSR5	DIRSR4	DIRSR3	DIRSR2	DIRSR1	DIRSR0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_DIRSR0 field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 DIRSR10	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
22 DIRSR9	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
23 DIRSR8	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
24 DIRSR7	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
25 DIRSR6	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
26 DIRSR5	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
27 DIRSR4	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
28 DIRSR3	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
29 DIRSR2	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected
30 DIRSR1	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin.

Table continues on the next page...

SIUL2_DIRSR0 field descriptions (continued)

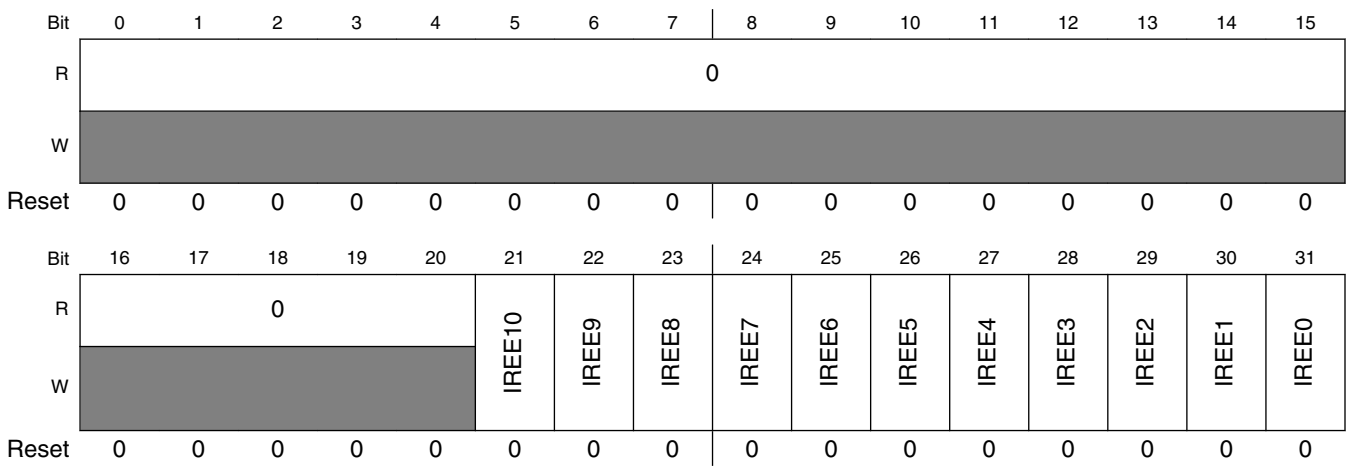
Field	Description
	0 Interrupt request is selected 1 DMA request is selected
31 DIRSR0	DMA/Interrupt Request Select Register-Selects between DMA request or external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 DMA request is selected

9.3.7 SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)

This register is used to enable the rising-edge triggered events on the corresponding interrupt pads.

This register supports 8-, 16-, and 32-bit accesses.

Address: 0h base + 28h offset = 28h



SIUL2_IREER0 field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 IREE10	Enable rising-edge events to cause the EIF10 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
22 IREE9	Enable rising-edge events to cause the EIF9 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled

Table continues on the next page...

SIUL2_IREER0 field descriptions (continued)

Field	Description
23 IREE8	Enable rising-edge events to cause the EIF8 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
24 IREE7	Enable rising-edge events to cause the EIF7 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
25 IREE6	Enable rising-edge events to cause the EIF6 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
26 IREE5	Enable rising-edge events to cause the EIF5 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
27 IREE4	Enable rising-edge events to cause the EIF4 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
28 IREE3	Enable rising-edge events to cause the EIF3 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
29 IREE2	Enable rising-edge events to cause the EIF2 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
30 IREE1	Enable rising-edge events to cause the EIF1 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
31 IREE0	Enable rising-edge events to cause the EIF0 bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled

9.3.8 SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)

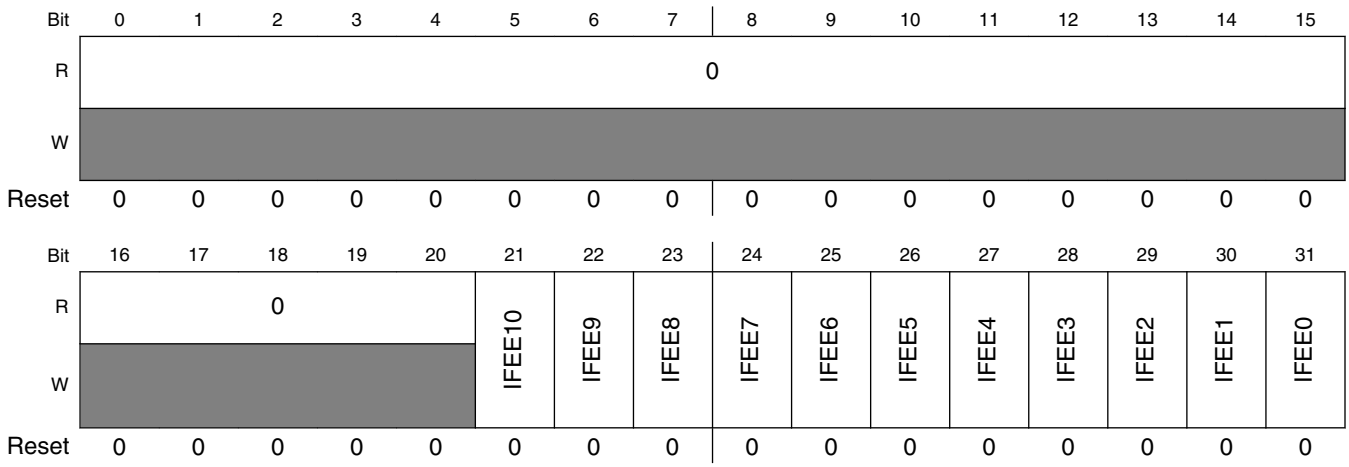
This register is used to enable falling-edge triggered events on the corresponding interrupt pads.

This register supports 8-, 16-, and 32-bit accesses.

NOTE

If both the IREE and the IFEE bits are cleared for the same interrupt source, the interrupt status flag for the corresponding external interrupt will never be set.

Address: 0h base + 30h offset = 30h



SIUL2_IFEER0 field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 IFEE10	Enable falling-edge events to cause the EIF10 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
22 IFEE9	Enable falling-edge events to cause the EIF9 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
23 IFEE8	Enable falling-edge events to cause the EIF8 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
24 IFEE7	Enable falling-edge events to cause the EIF7 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
25 IFEE6	Enable falling-edge events to cause the EIF6 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
26 IFEE5	Enable falling-edge events to cause the EIF5 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled

Table continues on the next page...

SIUL2_IFEER0 field descriptions (continued)

Field	Description
27 IFEE4	Enable falling-edge events to cause the EIF4 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
28 IFEE3	Enable falling-edge events to cause the EIF3 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
29 IFEE2	Enable falling-edge events to cause the EIF2 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
30 IFEE1	Enable falling-edge events to cause the EIF1 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
31 IFEE0	Enable falling-edge events to cause the EIF0 bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled

9.3.9 SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)

This register is used to enable a digital filter counter on the corresponding interrupt pads to filter out glitches on the inputs.

This register supports 8-, 16-, and 32-bit accesses.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					IFE10	IFE9	IFE8	IFE7	IFE6	IFE5	IFE4	IFE3	IFE2	IFE1	IFE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_IFER0 field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 IFE10	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
22 IFE9	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
23 IFE8	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
24 IFE7	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
25 IFE6	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
26 IFE5	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
27 IFE4	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
28 IFE3	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
29 IFE2	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
30 IFE1	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
31 IFE0	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled

9.3.10 SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR*n*)

These registers are used to configure the filter counter associated with each digital glitch filter.

NOTE

This register supports only 32-bit accesses. Byte and half-word write accesses are not supported.

Address: 0h base + 40h offset + (4d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																												MAXCNT			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_IFMCR*n* field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 MAXCNT	Maximum Interrupt Filter Counter setting MAXCNT can be 0d to 15d. • For a MAXCNT value of 0d, 1d, or 2d, the filter behaves as an ALL PASS filter. • For a MAXCNT value of 3d to 15d, the filter period is TCK*MAXCNT+n*TCK, where n is 0, 1, 2, 3, or 4. TCK is the prescaled filter clock period, which is the IRC clock prescaled to the IFCP value, which is specified in SIUL2_IFCPR. TIRC is the basic filter clock period: 62.5 ns (f = 16 MHz). **NOTE:** The filter delay is 2 TCK clock cycles more than the filter period. In general, TFILTER and TDELAY are not integer multiples of TCK because DIN is asynchronous whereas DOUT is synchronous with respect to the CK clock. The factor n accounts for this uncertainty in filter period calculation.

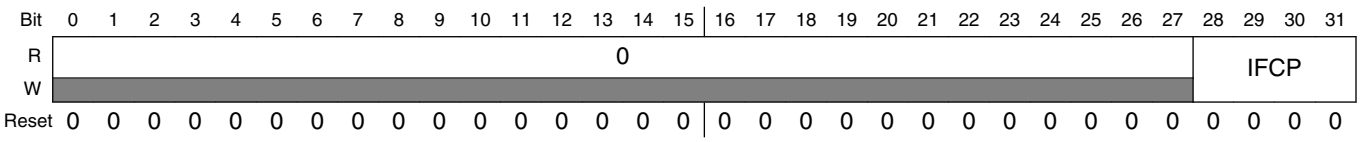
9.3.11 SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)

This register is used to configure a clock prescaler that selects the clock for all digital filters. A prescaler is applied to the input clock to the SIUL2, which is the system AIPS clock counter in the SIUL2.

NOTE

This register supports only 32-bit accesses. Byte and half-word write accesses are not supported.

Address: 0h base + C0h offset = C0h



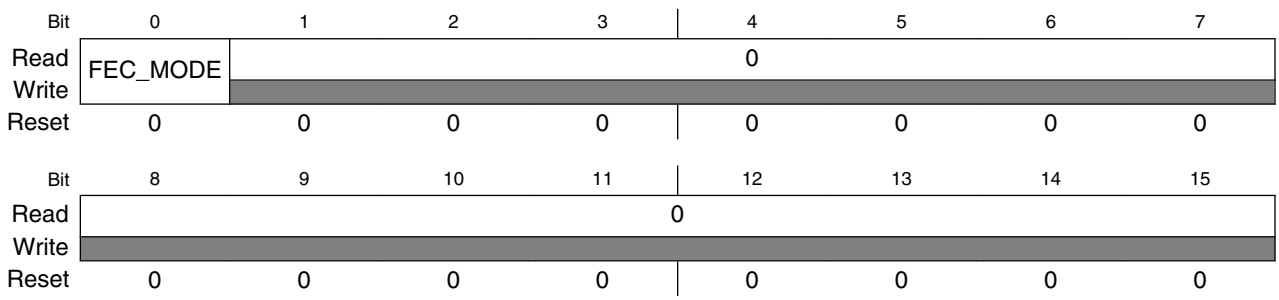
SIUL2_IFCPR field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 IFCP	Interrupt Filter Clock Prescaler setting Prescaled Filter Clock Period = T(IRC) x (IFCP + 1) T(IRC) is the internal oscillator period. IFCP can be 0 to 15

9.3.12 SoC Configuration Register 0 (SIUL2_SCR0*n*)

This register allows selecting between the Media Independent Interface (MII) and the Reduced Media Independent Interface (RMII) mode for the Fast Ethernet Controller (FEC).

Address: 0h base + 100h offset + (1d × i), where i=0d to 0d



SIUL2_SCR0*n* field descriptions

Field	Description
0 FEC_MODE	FEC Mode Setting 0 RMII is selected 1 MII-Lite is selected
1–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

9.3.13 SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR*n*)

The SIUL2_MSCR*n* register selects which source signal is connected to the register's associated destination, which is one of the following:

- A chip pin can be configured as an output
- An internal module port can be configured as an input

For the associated chip-pin destination, the register also specifies the pin's electrical properties.

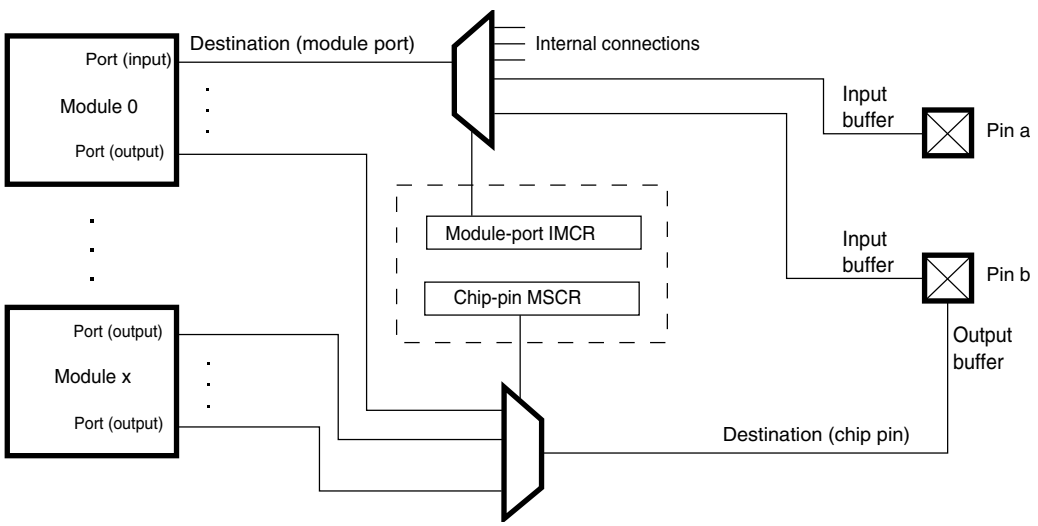


Figure 9-47. MSCR pin connection

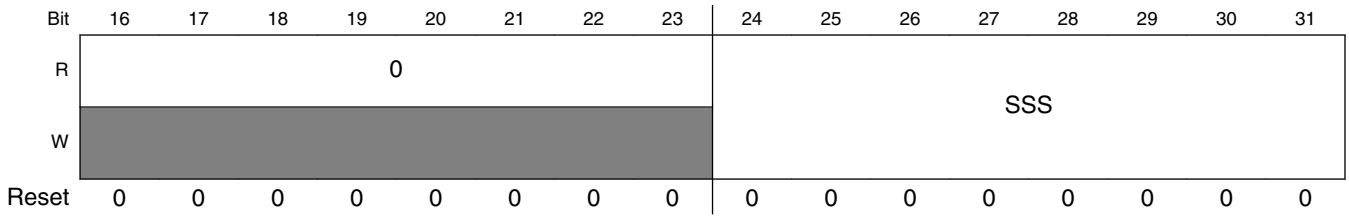
The fields in an MSCR vary depending on its associated destination (chip pin).

This register supports only 32-bit accesses. 8- and 16-bit accesses generate transfer errors.

For chip-pin MSCR assignments, pin types, APC support, and SSS values, see the pin-description table for your chip.

Address: 0h base + 240h offset + (4d × i), where i=0d to 255d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0		OERC		0		ODC		SMC	APC	ILS		IBE	HYS	WPDE	WPUE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1



SIUL2_MSCR*n* field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–3 OERC	Output Edge Rate Control Specifies the drive strength and slew rate of the associated pin. Valid for pad_sr_hv pads only. 00 :Half drive strength with slew rate control 01 : Full drive strength with slew rate control 10 : Half drive strength without slew rate control 11 : Full drive strength without slew rate control Refer to table "GPIO pad output buffer electrical characteristics" in the Data Sheet for electrical characteristics corresponding to these OERC bits.
4–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 ODC	Output Drive Control Used only when the associated destination is a chip pin. Specifies the type of output drive control for the associated pin. 00 Disabled (high impedance for turning off the pin or using it as an input) 01 Open-drain 10 Push-pull 11 Open-source
8 SMC	Safe Mode Control Used only when the associated destination is a chip pin. Specifies whether the chip disables the pin's output buffer when the chip enters Safe mode. 0 Disable (the output buffer returns to its previous state when the chip leaves Safe mode) 1 Don't disable
9 APC	Analog Pad Control Used only when the associated destination is a chip pin that supports analog I/O. Enables the pin's analog-input-path switch. 0 Disable (the switch is off) 1 Enable (another module can control the state of the switch)
10–11 ILS	Input Level Selection Used only when the associated destination is a chip pin. Specifies the logic family for the associated pin, which determines its logic switching levels.

Table continues on the next page...

SIUL2_MSCR*n* field descriptions (continued)

Field	Description
	NOTE: Supported input levels vary according to pad types. Pad type "pad_sr_hv" supports only the CMOS input level, while pad type "pad_isatww_st_hv" supports TTL and CMOS levels. Refer to the attached IO spreadsheet for the pad type of each pin. 00 Reserved 01 TTL 10 LVDS 11 CMOS
12 IBE	Input Buffer Enable Used only when the associated destination is a chip pin. Enables the associated pin's input buffer. 0 Disabled 1 Enabled
13 HYS	Input Hysteresis Used only when the associated destination is a chip pin. Enables input hysteresis for the associated pin. 0 Disabled 1 Enabled
14 WPDE	Weak Pulldown Enable Used only when the associated destination is a chip pin. Enables the associated pin's weak pulldown resistor. **NOTE:** If WPDE and WPUE are both set, the pull-up and pull-down are both disabled. 0 Disabled 1 Enabled
15 WPUE	Weak Pullup Enable Used only when the associated destination is a chip pin. Enables the associated pin's weak pullup resistor. For most of the pads, the WPUE reset value is 1. See the Signal Description IOMUX details for the reset values of the this bit. **NOTE:** If WPDE and WPUE are both set, the pull-up and pull-down are both disabled. 0 Disabled 1 Enabled
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 SSS	Source Signal Select Selects which source signal is connected to the associated destination (chip pin). This field has a variable width depending on the number of alternate functions available for that pin. For a chip pin, the source signals are outputs from module ports.

9.3.14 SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_MSCR*n*)

This register selects which source signal is connected to the register's associated destination, which is an internal module port that can be configured as an input.

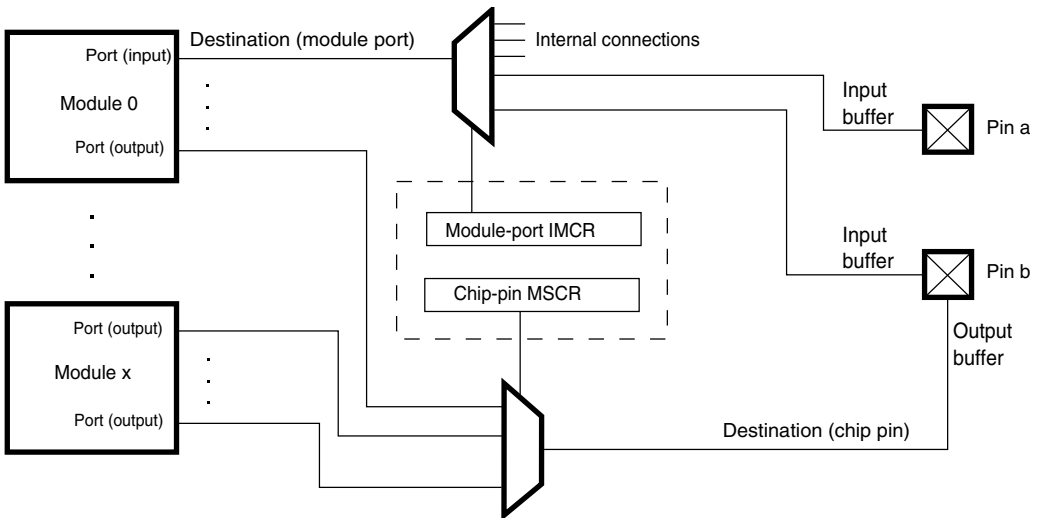


Figure 9-561. MSCR module-port connection

The fields in an MSCR vary depending on its associated destination (module port).

For MSCR assignments and SSS values, see the pin-description table.

Address: 0h base + A40h offset + (4d × i), where i=0d to 483d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								SSS							
W																																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- See the Signal Description details for the reset values of the MSCRs.

SIUL2_MSCR*n* field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 SSS	Source Signal Select Selects which source signal is connected to the associated destination (module port). This field has a variable width depending on the number of alternate functions available for that port. For a module port, the source signals are either outputs from module ports or inputs from chip pins. **NOTE:** This field can be up to 8 bits wide, depending on the number of multiplexed options. Please refer to the IOMUX sheet for the multiplexing details.

9.3.15 General Purpose Register 0 (SIUL2_GPR0)

The fields in the SIUL_GPR0, SIUL_GPR1, and SIUL_GPR2 registers, in combination, set the function of the corresponding pads as follows:

Table 9-789. SIUL_GPR0/1/2 function

STR1	STR0	SEL1	SEL0	Pad Function
x	x	0	0	Pull disabled (Hi-Z)
0	0	0	1	Pull disabled (Hi-Z)
0	1	0	1	200K Ohm Pull-Up
1	0	0	1	100K Ohm Pull-Up
1	1	0	1	5K Ohm Pull-Up
0	0	1	0	Pull disabled (Hi-Z)
0	1	1	0	200K Ohm Pull-Down
1	0	1	0	100K Ohm Pull-Down
1	1	1	0	5K Ohm Pull-Down
0	0	1	1	Pull disabled (Hi-Z)
0	1	1	1	200K Ohm Pull-Up/ Down Both
1	0	1	1	100K Ohm Pull-Up/ Down Both
1	1	1	1	5K Ohm Pull-Up/Down Both

Address: 0h base + 1240h offset = 1240h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	GPI243_STR1	GPI243_STR0	GPI243_SEL1	GPI243_SEL0	GPI242_STR1	GPI242_STR0	GPI242_SEL1	GPI242_SEL0	GPI241_STR1	GPI241_STR0	GPI241_SEL1	GPI241_SEL0	GPI240_STR1	GPI240_STR0	GPI240_SEL1	GPI240_SEL0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GPI227_STR1	GPI227_STR0	GPI227_SEL1	GPI227_SEL0	GPI226_STR1	GPI226_STR0	GPI226_SEL1	GPI226_SEL0	GPI225_STR1	GPI225_STR0	GPI225_SEL1	GPI225_SEL0	GPI224_STR1	GPI224_STR0	GPI224_SEL1	GPI224_SEL0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_GPR0 field descriptions

Field	Description
31 GPI243_STR1	ipp_str_1 control for GPI_243
30 GPI243_STR0	ipp_str_0 control for GPI_243
29 GPI243_SEL1	ipp_sel_1 control for GPI_243
28 GPI243_SEL0	ipp_sel_0 control for GPI_243
27 GPI242_STR1	ipp_str_1 control for GPI_242
26 GPI242_STR0	ipp_str_0 control for GPI_242
25 GPI242_SEL1	ipp_sel_1 control for GPI_242
24 GPI242_SEL0	ipp_sel_0 control for GPI_242
23 GPI241_STR1	ipp_str_1 control for GPI_241
22 GPI241_STR0	ipp_str_0 control for GPI_241
21 GPI241_SEL1	ipp_sel_1 control for GPI_241
20 GPI241_SEL0	ipp_sel_0 control for GPI_241
19 GPI240_STR1	ipp_str_1 control for GPI_240
18 GPI240_STR0	ipp_str_0 control for GPI_240
17 GPI240_SEL1	ipp_sel_1 control for GPI_240
16 GPI240_SEL0	ipp_sel_0 control for GPI_240
15 GPI227_STR1	ipp_str_1 control for GPI_227
14 GPI227_STR0	ipp_str_0 control for GPI_227
13 GPI227_SEL1	ipp_sel_1 control for GPI_227
12 GPI227_SEL0	ipp_sel_0 control for GPI_227
11 GPI226_STR1	ipp_str_1 control for GPI_226
10 GPI226_STR0	ipp_str_0 control for GPI_226
9 GPI226_SEL1	ipp_sel_1 control for GPI_226

Table continues on the next page...

SIUL2_GPR0 field descriptions (continued)

Field	Description
8 GPI226_SEL0	ipp_sel_0 control for GPI_226
7 GPI225_STR1	ipp_str_1 control for GPI_225
6 GPI225_STR0	ipp_str_0 control for GPI_225
5 GPI225_SEL1	ipp_sel_1 control for GPI_225
4 GPI225_SEL0	ipp_sel_0 control for GPI_225
3 GPI224_STR1	ipp_str_1 control for GPI_224
2 GPI224_STR0	ipp_str_0 control for GPI_224
1 GPI224_SEL1	ipp_sel_1 control for GPI_224
0 GPI224_SEL0	ipp_sel_0 control for GPI_224

9.3.16 General Purpose Register 1 (SIUL2_GPR1)

For the field settings corresponding to the pad functions, see the table "SIUL_GPR0/1/2 function" in the SIUL_GPR0 register section.

Address: 0h base + 1244h offset = 1244h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPI251_STR1	GPI251_STR0	GPI251_SEL1	GPI251_SEL0	GPI250_STR1	GPI250_STR0	GPI250_SEL1	GPI250_SEL0	GPI249_STR1	GPI249_STR0	GPI249_SEL1	GPI249_SEL0	GPI248_STR1	GPI248_STR0	GPI248_SEL1	GPI248_SEL0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPI247_STR1	GPI247_STR0	GPI247_SEL1	GPI247_SEL0	GPI246_STR1	GPI246_STR0	GPI246_SEL1	GPI246_SEL0	GPI245_STR1	GPI245_STR0	GPI245_SEL1	GPI245_SEL0	GPI244_STR1	GPI244_STR0	GPI244_SEL1	GPI244_SEL0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_GPR1 field descriptions

Field	Description
31 GPI251_STR1	ipp_str_1 control for GPI_251
30 GPI251_STR0	ipp_str_0 control for GPI_251
29 GPI251_SEL1	ipp_sel_1 control for GPI_251
28 GPI251_SEL0	ipp_sel_0 control for GPI_251
27 GPI250_STR1	ipp_str_1 control for GPI_250
26 GPI250_STR0	ipp_str_0 control for GPI_250
25 GPI250_SEL1	ipp_sel_1 control for GPI_250
24 GPI250_SEL0	ipp_sel_0 control for GPI_250
23 GPI249_STR1	ipp_str_1 control for GPI_249
22 GPI249_STR0	ipp_str_0 control for GPI_249
21 GPI249_SEL1	ipp_sel_1 control for GPI_249
20 GPI249_SEL0	ipp_sel_0 control for GPI_249
19 GPI248_STR1	ipp_str_1 control for GPI_248
18 GPI248_STR0	ipp_str_0 control for GPI_248
17 GPI248_SEL1	ipp_sel_1 control for GPI_248
16 GPI248_SEL0	ipp_sel_0 control for GPI_248
15 GPI247_STR1	ipp_str_1 control for GPI_247
14 GPI247_STR0	ipp_str_0 control for GPI_247
13 GPI247_SEL1	ipp_sel_1 control for GPI_247
12 GPI247_SEL0	ipp_sel_0 control for GPI_247
11 GPI246_STR1	ipp_str_1 control for GPI_246
10 GPI246_STR0	ipp_str_0 control for GPI_246
9 GPI246_SEL1	ipp_sel_1 control for GPI_246

Table continues on the next page...

SIUL2_GPR1 field descriptions (continued)

Field	Description
8 GPI246_SEL0	ipp_sel_0 control for GPI_246
7 GPI245_STR1	ipp_str_1 control for GPI_245
6 GPI245_STR0	ipp_str_0 control for GPI_245
5 GPI245_SEL1	ipp_sel_1 control for GPI_245
4 GPI245_SEL0	ipp_sel_0 control for GPI_245
3 GPI244_STR1	ipp_str_1 control for GPI_244
2 GPI244_STR0	ipp_str_0 control for GPI_244
1 GPI244_SEL1	ipp_sel_1 control for GPI_244
0 GPI244_SEL0	ipp_sel_0 control for GPI_244

9.3.17 General Purpose Register 2 (SIUL2_GPR2)

For the field settings corresponding to the pad functions, see the table "SIUL_GPR0/1/2 function" in the SIUL_GPR0 register section.

Address: 0h base + 1248h offset = 1248h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GPI255_STR1	GPI255_STR0	GPI255_SEL1	GPI255_SEL0	GPI254_STR1	GPI254_STR0	GPI254_SEL1	GPI254_SEL0	GPI253_STR1	GPI253_STR0	GPI253_SEL1	GPI253_SEL0	GPI252_STR1	GPI252_STR0	GPI252_SEL1	GPI252_SEL0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_GPR2 field descriptions

Field	Description
31–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 GPI255_STR1	ipp_str_1 control for GPI_255
14 GPI255_STR0	ipp_str_0 control for GPI_255
13 GPI255_SEL1	ipp_sel_1 control for GPI_255
12 GPI255_SEL0	ipp_sel_0 control for GPI_255
11 GPI254_STR1	ipp_str_1 control for GPI_254
10 GPI254_STR0	ipp_str_0 control for GPI_254
9 GPI254_SEL1	ipp_sel_1 control for GPI_254
8 GPI254_SEL0	ipp_sel_0 control for GPI_254
7 GPI253_STR1	ipp_str_1 control for GPI_253
6 GPI253_STR0	ipp_str_0 control for GPI_253
5 GPI253_SEL1	ipp_sel_1 control for GPI_253
4 GPI253_SEL0	ipp_sel_0 control for GPI_253
3 GPI252_STR1	ipp_str_1 control for GPI_252
2 GPI252_STR0	ipp_str_0 control for GPI_252
1 GPI252_SEL1	ipp_sel_1 control for GPI_252
0 GPI252_SEL0	ipp_sel_0 control for GPI_252

9.3.18 SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO*n*)

These registers can be used to set or clear a single GPIO pad with a byte access.

These registers can be accessed one, two, or four at a time using 8-, 16- or 32-bit accesses at the 32-bit aligned address.

Address: 0h base + 1300h offset + (1d × i), where i=0d to 255d

Bit	0	1	2	3	4	5	6	7
Read	0							PDO
Write								
Reset	0	0	0	0	0	0	0	0

SIUL2_GPDO*n* field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 PDO	Pad Data Out This bit stores the data to be driven out on the external GPIO pad controlled by this register. 0 Logic low value is driven on the corresponding GPIO pad when the pad is configured as an output 1 Logic high value is driven on the corresponding GPIO pad when the pad is configured as an output

9.3.19 SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI*n*)

These registers can be used to read the GPIO pad data with a byte access.

These registers can be accessed one, two, or four at a time using 8-, 16- or 32-bit accesses at the 32-bit aligned address.

Address: 0h base + 1500h offset + (1d × i), where i=0d to 255d

Bit	0	1	2	3	4	5	6	7
Read	0							PDI
Write								
Reset	0	0	0	0	0	0	0	0

SIUL2_GPDI*n* field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 PDI	Pad Data In This bit stores the value of the external GPIO pad associated with this register. 0 The value of the data in signal for the corresponding GPIO pad is logic low 1 The value of the data in signal for the corresponding GPIO pad is logic high

9.3.20 SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO*n*)

These registers are used to set or clear the respective pads of the device. Parallel port registers for input (PGPDI) and output (PGPDO) are provided to allow a complete port to be written or read in one operation, dependent on the individual pad configuration. The difference between PGPDO and GPDO is that PGPDO registers can be used to set the values of all output pins assigned to a device port with a single 16-bit register write vs. the GPDO registers, which are used to set the value on a specific pin with a byte write.

The SIUL2_PGPDO registers access the same physical resource as the SIUL2_PDO and SIUL2_MPGPDO address locations. Some examples of the mapping:

- PPDO[0][0] = PDO[0]
- PPDO[2][0] = PDO[32]
- PPDO[31][15] = PDO[511]

These registers support 8-, 16-, and 32-bit accesses.

Address: 0h base + 1700h offset + (2d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	PPDO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_PGPDO*n* field descriptions

Field	Description
0–15 PPDO	Parallel Pad Data Out Write or read the data register that stores the value to be driven on the pad in output mode. Access to this register location are coherent with access to the bit-wise GPIO Pad Data Output Registers (SIUL2_GPDO).The x and bit index define which PPDO register bit is equivalent to which PDO register bit according to the following equation: PPDO[x][y] = PDO[(x*16)+y]

9.3.21 SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI*n*)

These registers hold the synchronized input value from the pads. Parallel port registers for input (PGPDI) and output (PGPDO) are provided to allow a complete port to be written or read in one operation, dependent on the individual pad configuration.The difference between PGPDI and GPDI registers is that PGPDI registers can be used to read the values of all input pins assigned to a device port with a single 16-bit register read vs. the GPDI registers, which are used to read the value on a specific pin with a byte read.

These registers support 8-, 16-, and 32-bit accesses.

Address: 0h base + 1740h offset + (2d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	PPDI															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_PGPDI*n* field descriptions

Field	Description
0–15 PPDI	Parallel Pad Data In Reads the current pad value. Access to this register location are coherent with access to the bit-wise GPIO Pad Data Input Registers (SIUL2_GPDI). The x and bit index define which PPDI register bit is equivalent to which PDI register bit according to the following equation: PPDI[x][y] = PDI[(x*16)+y]

9.3.22 SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO*n*)

This register can be used to selectively modify the pad values associated to PPDO[x][0:15]. The MPGPDO[x] register may only be accessed with 32-bit writes. 8-bit or 16-bit writes will not modify any bits in the register and cause a transfer error response by the module. Read access will return 0.

NOTE

These registers support only 32-bit accesses. Byte and half-word write accesses are not supported.

Address: 0h base + 1780h offset + (4d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																0															
W	MASK																MPPDO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_MPGPDO*n* field descriptions

Field	Description
0–15 MASK	Mask Field Each bit corresponds to one data bit in the MPPDO[x] register at the same bit location. 0b The associated bit value in the MPPDO[x] field is ignored 1b The associated bit value in the MPPDO[x] field is written
16–31 MPPDO	Masked Parallel Pad Data Out Write the data register that stores the value to be driven on the pad in

Table continues on the next page...

SIUL2_MPGPDO*n* field descriptions (continued)

Field	Description
	output mode. Access to this register location are coherent with access to the bit-wise GPIO Pad Data Output Registers (PDO).The x and bit index define which MPPDO register bit is equivalent to which PDO register bit according to the following equation: MPPDO[x][y] = PDO[(x*16)+y]

9.4 Functional description

9.4.1 General

This section provides a complete functional description of the System Integration Unit Lite2.

9.4.2 Pad Control

The SIUL2 is capable of controlling the electrical characteristic of up to 512 pads. It provides a consistent interface for all pads, both on a by-port and a by-bit basis. The following describes the pad characteristics that can be supported by the SIUL2. Not all of these features will be supported on all devices or on all pads as this is dependent on the specific application needs an the pads implemented on the device.

The required controls are:

- Slew rate control
  - Offers improved EMC performance.
  - Support for a fast and a slow slew rate.
- Drive strength control
  - Offers improved EMC performance.
  - Up to 4 different drive strength levels can be supported.
- Pull capability

    - Offers flexibility in the device configuration and the elimination of external hardware in some cases.
    - The configuration of the pull on any pad should be independently controlled to be either pull-up, pull-down or no-pull enabled.
- Input and Output Buffer control
- Hysteresis
- Open drain
    - Needed to support different pads muxed (e.g. mux IIC, with non open drain pads)
- Pad Output Assignment
    - This setting defines which chip function has control over the output of the pad.

The setting of each pad out of reset is fixed per MCU, but can be configured individually. In this way it is possible to select special pull settings or peripheral pad ownership per design.

It is possible for the user software to configure each pad independently of all other pads on the device or other pads grouped within a single port. This allows different pad types to be grouped together in ports and allows the necessary flexibility needed for the pads individual operation. This is achieved by grouping all of the above functions into a single register for each pad on the device, and allows each pad to be configured with a single write to one register and allowing simplified duplication of software for each pad with indexed changes for each pad.

9.4.3 General Purpose Input and Output Pads (GPIO)

The SIUL2 allows each pad to be configured as either a General Purpose Input or Output pad (GPIO), and as one or more alternate functions (input or output), the function of which is determined by the peripheral that will use the pad.

GPIO pads can also optionally be implemented without any alternate function.

The SIUL2 is designed to manage as many as 512 GPIO pads organized as ports that can be recognized and addressed with read/write operations. The GPDI, GPDO, PGPDO, and PGPDI registers can be accessed with 8-, 16-, or 32-bit reads and writes. The MSCR registers can only be accessed with 32-bit reads and writes.

The SIUL2 has separate data input and data output registers for all pads, allowing the possibility of reading back an input or output value of a pad directly. This supports the ability to validate what is present on the pad rather than simply confirming the value that was written to the data register by accessing the data input registers.

The data output registers support both read and write operations to be performed.

The data input registers support read access only.

When the pad is configured to use one of its alternate functions, the data input value reflect the respective value of the pad. If a write operation is performed to the data output register for a pad configured as an alternate function (non GPIO), this write will not be reflected by the pad value until reconfigured to GPIO. All general purpose pads are implemented as bidirectional.

Note

In case the bidirectional operation impacts some performances (e.g. ADC accuracy), or is not required for a specific pad function it is acceptable to limit the functionality of some pads to "Input Only."

9.4.4 External Interrupts/DMA Requests (REQ Pins)

The SIUL2 supports 11 external interrupts, which can be allocated to the EIRQ pins.

For the assignments of the external interrupts, see the Signal Description details.

The SIUL2 supports 2 interrupt vectors, which are multiplexed to provide 3 and 8 external interrupt sources to the interrupt controller of the MCU. Each interrupt vector can support up to 8 external interrupt sources from the device pads.

All the external interrupt pads within a single group (maximum 8 pads) have equal priority. It is the responsibility of the user software to search through the group of sources in the most appropriate way for their application.

The priority of the vectors used by the external interrupt pads is fixed based on the platform and the interrupt controller and its priority levels, but the allocation of pads to each group of interrupts can be independently configured by the MCU.

The external interrupt lines can have digital glitch filters applied to them. The supported range is 1 to 32. The glitch filters need a running internal oscillator clock to work. If no such clock is available, external interrupts will be effectively disabled when the glitch filter is enabled on a interrupt line.

9.4.4.1 External interrupt initialization

When an external interrupt pin is first enabled, it is possible to get a false interrupt flag. To prevent a false interrupt request, during pin interrupt initialization, the user must do the following:

- Mask interrupts by clearing the EIREn bits in DIRER0.
- Select the pin polarity by setting the appropriate IREEn bits in IREER0 and the appropriate IFEEn bits in IFEER0 as desired.
- Configure the appropriate bits in the MSCR[0:511] register for the external interrupt pin(s) desired as follows:
  - Clear the ODC bits to disable output.
  - If using the internal weak pullup/pulldown, configure the appropriate WPUE and WPDE fields.

Note

MSCR[SSS] bits do not need to be changed for either MSCR[0:511] or IMCR[0:511] because the external interrupt pin input is directly connected to the SIUL for EIRQ pins.

External interrupt pins should never be configured as outputs (MSCR[ODC] bits are not zeros) when external interrupt inputs are desired because false interrupts could be detected (such as from a GPIO configuration).

- Select the request desired between DMA or Interrupt by writing the appropriate DIRSn bits in DIRSR0.
- Select the desired glitch filter setup for the pins by writing the following:
  - Write the Filter Counter setting to the desired value by writing the MAXCNT[3:0] bits in the IFMCn register for the respective external interrupt that is being used.
  - Set the Filter Clock Prescaler setting from 0 to 15 to the desired value by writing the IFCP[3:0] bits in the IFCPR register.
  - Enable the glitch filter for the desired external interrupt pins by setting the appropriate IFEn bits in IFER0.

- Write to EIFn bits in DISR0 to as desired to clear any flags.
- Enable the interrupt pins by setting the appropriate EIREn bits in DIREER0.

9.4.4.2 External interrupt management

The user can enable or disable each interrupt independently using a single rolled up register, SIUL2_DIRER0.

A pad defined as an external interrupt can be configured by the user to recognize interrupts with an active rising edge, an active falling edge, or both edges being active. A setting of having both edge events disabled is reserved and should not be configured. The user controls the active IRQ edge through the registers SIUL2_IREER and SIUL2_IFEER.

Each external interrupt supports an individual flag which is held in the Flag register (DISR0). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register.

Refer to the following figure for an overview of the external interrupt implementation.

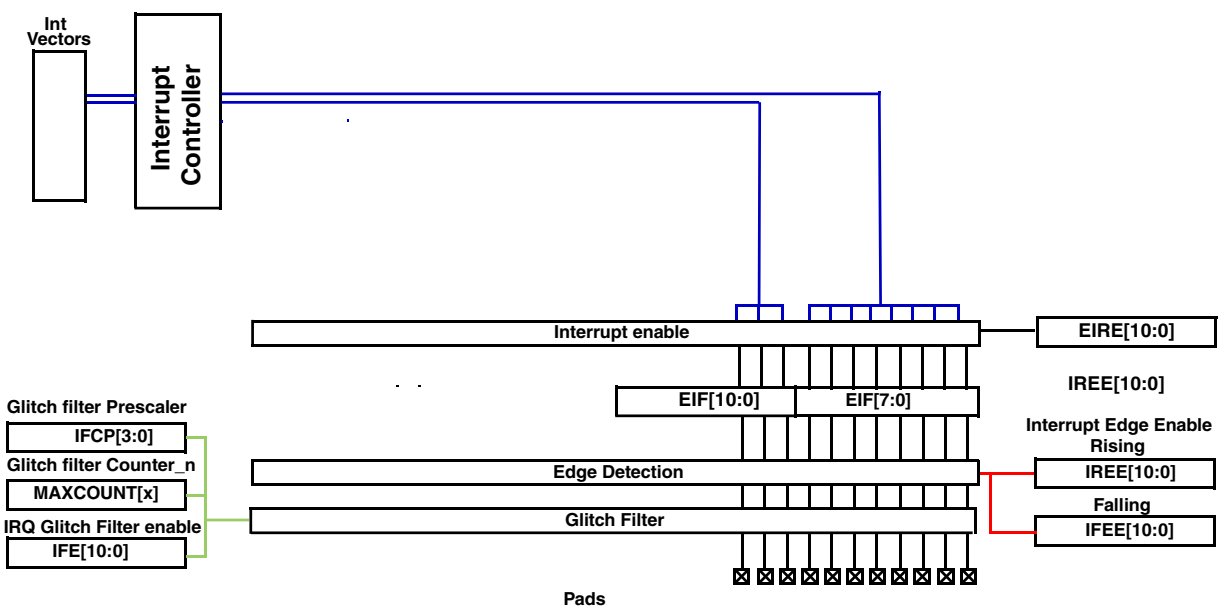


Figure 9-2098. External interrupt pad diagram

9.4.4.3 External Interrupt request

The REQ input pins on the device are sources for interrupt or DMA requests. The system has two interrupt vectors available for the REQ pins in the SIUL2. The mapping of 11 interrupt request sources to vectors and channels appears in the following table.

Table 9-1357. Interrupt source mapping to SIUL2 Interrupt request output for 11 interrupt sources

Vector/Channel #	Interrupt Vector Source
0	REQ[07] | REQ[06] | REQ[05] | REQ[04] | REQ[03] | REQ[02] | REQ[01] | REQ[00]
1	REQ[10] | REQ[09] | REQ[08]
2	Reserved
3	Reserved

9.4.4.4 DMA requests

The REQ pins on the device map to independent DMA request channels in the DMA controller. The eight DMA request pins are mapped to independent DMA request channels in the DMA controller. See the DMA request mapping table in the DMACHMUX chapter.

Chapter 10
Crossbar Switch (AXBS)

10.1 Crossbar switch configuration

This chip contains two crossbar switch modules:

- AXBS0
  - Computational shell crossbar switch
  - 64-bit data bus
- AXBS1
  - Peripheral shell crossbar switch
  - 32-bit data bus

The following sections describe how each is configured for this chip.

10.1.1 Computational shell crossbar switch configuration

This section summarizes how the module has been configured in the chip.

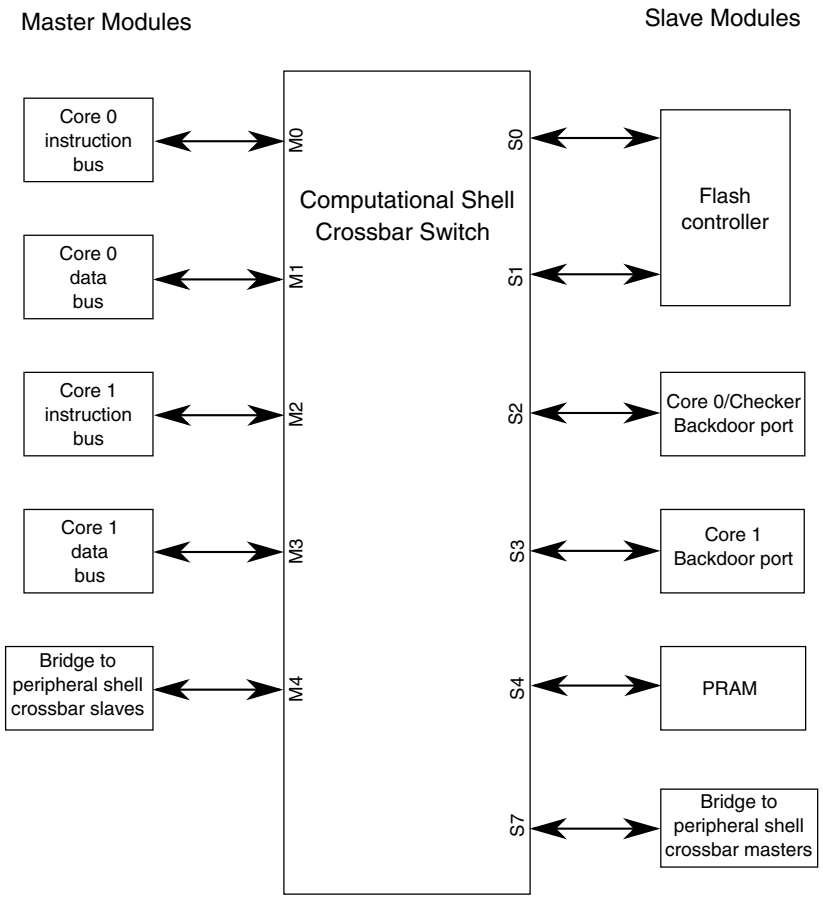


Figure 10-1. Computational shell crossbar switch integration

10.1.1.1 Computational Shell Crossbar Switch Master Assignments

This device contains five masters connected to the computational-shell crossbar switch.

The master assignment is as follows:

Master module	Master port number
Core 0 instruction bus	0
Core 0 data bus	1
Core 1 instruction bus	2
Core 1 data bus	3
Intelligent bridging bus gasket (peripheral-shell crossbar switch)	4

10.1.1.2 Computational Shell Crossbar Switch Slave Assignments

This device contains six slaves connected to the computational shell's crossbar switch.

The slave assignment is as follows:

Slave module	Slave port number
Flash memory controller port 0	0
Flash memory controller port 1	1
Core 0/Checker Core Unified Backdoor I/F	2
Core 1 Unified Backdoor I/F	3
PRAM controller	4
Intelligent Bridging Bus Gasket (peripheral-shell crossbar switch)	7

10.1.1.3 Crossbar Master and Slave Connections

The masters and slaves of the crossbar switches can be connected as follows:

Table 10-1. XBAR0 master and slave connections

	Slave	Flash memory controller port 0	Flash memory controller port 1	Core 0/ Checker Core Unified Backdoor I/F	Core 1 Unified Backdoor I/F	PRAM controller	XBAR0 to 1 interconnect
Master	Port No.	0	1	2	3	4	7
Core 0 instruction bus	0	X	—	X	X	X	X
Core 0 data bus	1	—	X	X	X	X	X
Core 1 instruction bus	2	—	X	X	X	X	X
Core 1 data bus	3	X	—	X	X	X	X
XBAR1 to 0 interconnect	4	X	—	X	X	X	—

Table 10-2. XBAR1 master and slave connections

	Slave	XBAR1 to 0 interconnect	Peripheral bridge (AIPS) PBRIDGE_1	Peripheral bridge (AIPS) PBRIDGE_0
Master	Port No.	0	2	3
DMA	1	X	X	X
FEC and LFAST/SIPI	2	X	X	X
XBAR0 to 1 interconnect	3	—	X	X

"X" signifies a possible connection.

10.1.1.4 Hardware Implementation Dependent Register (HID1)

HID1 is the core register (SPR 1009) that controls AXBS interrupt priority elevation.

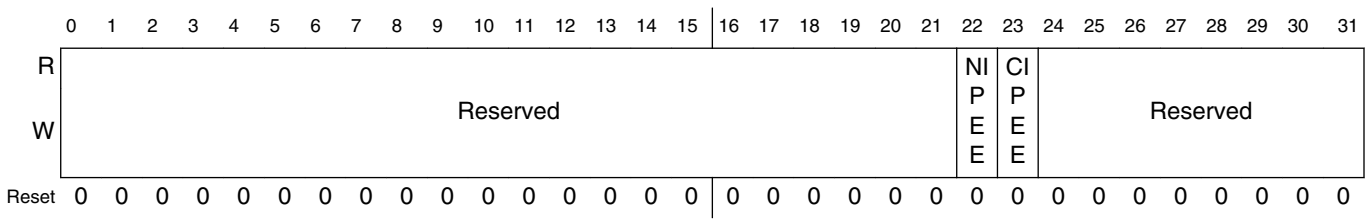


Table 10-4. Hardware Implementation Dependent Register Field Description

Field	Name	Description
0 - 21	Reserved	This field is reserved.
22	NIPEE	Normal Interrupt Priority Elevation Enable 0 Disables the platform's crossbar highpriority elevation for normal, external interrupts 1 Enables the platform's crossbar highpriority elevation for normal, external interrupts
23	CIPEE	Critical Interrupt Priority Elevation Enable 0 Disables the platform's crossbar highpriority elevation for NMI and critical interrupts 1 Enables the platform's crossbar highpriority elevation for NMI and critical interrupts
24-31	Reserved	This field is reserved.

10.1.1.5 Computational shell (XBAR0) memory map

Table 10-5. XBAR0 memory map

Absolute address (hex)	Register name	Width (bits)	Access	Reset value
FC004000	Priority Registers Slave (XBAR0_PRS0)	32	R/W	0000_1324h
FC004010	Control Register (XBAR0_CRS0)	32	R/W	001F_0000h
FC004100	Priority Registers Slave (XBAR0_PRS1)	32	R/W	0000_1324h
FC004110	Control Register (XBAR0_CRS1)	32	R/W	001F_0000h
FC004200	Priority Registers Slave (XBAR0_PRS2)	32	R/W	0000_1324h
FC004210	Control Register (XBAR0_CRS2)	32	R/W	001F_0000h
FC004300	Priority Registers Slave (XBAR0_PRS3)	32	R/W	0000_1324h
FC004310	Control Register (XBAR0_CRS3)	32	R/W	001F_0000h
FC004400	Priority Registers Slave (XBAR0_PRS4)	32	R/W	0000_1324h

Table continues on the next page...

Table 10-5. XBAR0 memory map (continued)

Absolute address (hex)	Register name	Width (bits)	Access	Reset value
FC004410	Control Register (XBAR0_CRS4)	32	R/W	001F_0000h
FC004700	Priority Registers Slave (XBAR0_PRS7)	32	R/W	0000_1324h
FC004710	Control Register (XBAR0_CRS7)	32	R/W	001F_0000h
FC004800	Master General Purpose Control Register (XBAR0_MGPCR0)	32	R/W	0000_0000h
FC004900	Master General Purpose Control Register (XBAR0_MGPCR1)	32	R/W	0000_0000h
FC004A00	Master General Purpose Control Register (XBAR0_MGPCR2)	32	R/W	0000_0000h
FC004B00	Master General Purpose Control Register (XBAR0_MGPCR3)	32	R/W	0000_0000h
FC004C00	Master General Purpose Control Register (XBAR0_MGPCR4)	32	R/W	0000_0000h

10.1.2 Peripheral Shell Crossbar Switch Configuration

This section summarizes how the peripheral-shell crossbar switch has been configured in the chip.

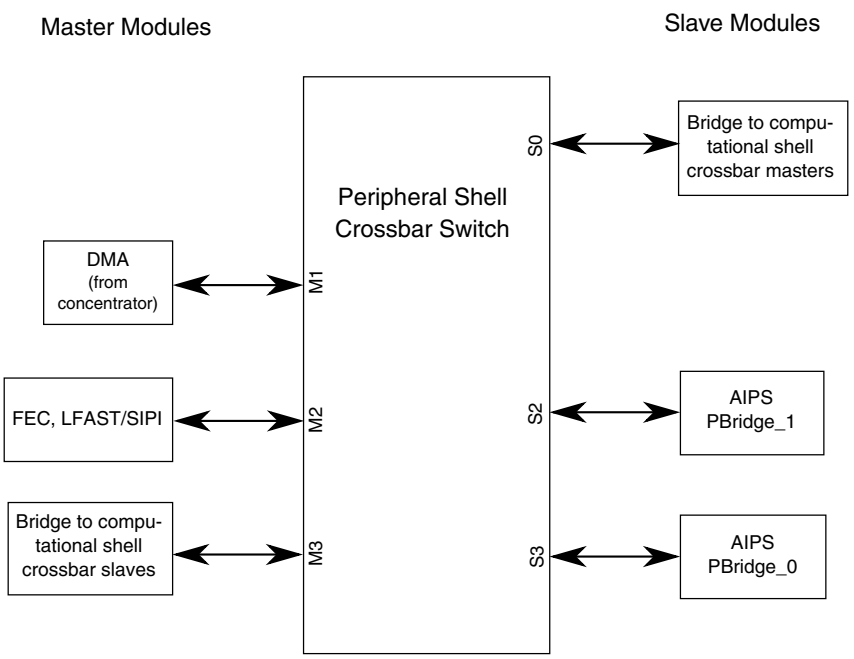


Figure 10-2. Peripheral shell crossbar switch integration

10.1.2.1 Peripheral Shell Crossbar Switch Master Assignments

This device contains three masters connected to the peripheral-shell crossbar switch.

The master assignment is as follows:

Master module	Master port number
DMA	1
FEC and LFAST/SIPI (Zipwire)	2
Intelligent Bridging Bus Gasket (computational-shell crossbar switch)	3

10.1.2.2 Peripheral Shell Crossbar Switch Slave Assignments

This device contains three slaves connected to the peripheral-shell crossbar switch.

The slave assignment is as follows:

Slave module	Slave port number
Intelligent Bridging Bus Gasket (computational-shell crossbar switch)	0
Peripheral bridge (AIPS) PBRIDGE_1	2
Peripheral bridge (AIPS) PBRIDGE_0	3

10.1.2.3 Peripheral shell (XBAR1) memory map

Table 10-6. XBAR1 memory map

Absolute address (hex)	Register name	Width (bits)	Access	Reset value
FC008000	Priority Registers Slave (XBAR1_PRS0)	32	R/W	0000_2100h
FC008010	Control Register (XBAR1_CRS0)	32	R/W	000E_0000h
FC008200	Priority Registers Slave (XBAR1_PRS2)	32	R/W	0000_2100h
FC008210	Control Register (XBAR1_CRS2)	32	R/W	000E_0000h
FC008300	Priority Registers Slave (XBAR1_PRS3)	32	R/W	0000_2100h
FC008310	Control Register (XBAR1_CRS3)	32	R/W	000E_0000h
FC008900	Master General Purpose Control Register (XBAR1_MGPCR1)	32	R/W	0000_0000h
FC008A00	Master General Purpose Control Register (XBAR1_MGPCR2)	32	R/W	0000_0000h
FC008B00	Master General Purpose Control Register (XBAR1_MGPCR3)	32	R/W	0000_0000h

10.2 Introduction

The information found here provides information on the layout, configuration, and programming of the crossbar switch.

The crossbar switch connects bus masters and bus slaves using a crossbar switch structure. This structure allows all bus masters to access different bus slaves simultaneously, while providing arbitration among the bus masters when they access the same slave. A variety of bus arbitration methods and attributes may be programmed on a slave-by-slave basis.

10.2.1 Features

The crossbar switch includes these features:

- Symmetric crossbar bus switch implementation
  - Allows concurrent accesses from different masters to different slaves
  - Slave arbitration attributes configured on a slave-by-slave basis
- Up to single-clock 64-bit transfer
- Support for burst transfers of up to 16 beats of data
- Low-Power Park mode support

10.3 Memory Map / Register Definition

Each slave port of the crossbar switch contains configuration registers. Read- and write-transfers require two bus clock cycles. The registers can be read from and written to only in supervisor mode. Additionally, these registers can be read from or written to only by 32-bit accesses.

A bus error response is returned if an unimplemented location is accessed within the crossbar switch.

The CRS*n* and PRS*n* registers can be programmed to be read-only to prevent changes to their configuration. After being read-only protected, future writes to them will terminate with an error.

NOTE

This section shows the registers for all eight master and slave ports. If a master or slave is not used on this particular device, then unexpected results occur when writing to its registers. See the chip configuration details for the exact master/slave assignments for your device.

All references to the crossbar switch registers are based on the physical port connections.

XBAR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Priority Registers Slave (XBAR_PRS0)	32	R/W	See section	10.3.1/319
10	Control Register (XBAR_CRS0)	32	R/W	See section	10.3.2/321
100	Priority Registers Slave (XBAR_PRS1)	32	R/W	See section	10.3.1/319
110	Control Register (XBAR_CRS1)	32	R/W	See section	10.3.2/321
200	Priority Registers Slave (XBAR_PRS2)	32	R/W	See section	10.3.1/319
210	Control Register (XBAR_CRS2)	32	R/W	See section	10.3.2/321
300	Priority Registers Slave (XBAR_PRS3)	32	R/W	See section	10.3.1/319
310	Control Register (XBAR_CRS3)	32	R/W	See section	10.3.2/321
400	Priority Registers Slave (XBAR_PRS4)	32	R/W	See section	10.3.1/319
410	Control Register (XBAR_CRS4)	32	R/W	See section	10.3.2/321
500	Priority Registers Slave (XBAR_PRS5)	32	R/W	See section	10.3.1/319
510	Control Register (XBAR_CRS5)	32	R/W	See section	10.3.2/321
600	Priority Registers Slave (XBAR_PRS6)	32	R/W	See section	10.3.1/319
610	Control Register (XBAR_CRS6)	32	R/W	See section	10.3.2/321
700	Priority Registers Slave (XBAR_PRS7)	32	R/W	See section	10.3.1/319
710	Control Register (XBAR_CRS7)	32	R/W	See section	10.3.2/321
800	Master General Purpose Control Register (XBAR_MGPCR0)	32	R/W	0000_0000h	10.3.3/323
900	Master General Purpose Control Register (XBAR_MGPCR1)	32	R/W	0000_0000h	10.3.3/323
A00	Master General Purpose Control Register (XBAR_MGPCR2)	32	R/W	0000_0000h	10.3.3/323
B00	Master General Purpose Control Register (XBAR_MGPCR3)	32	R/W	0000_0000h	10.3.3/323
C00	Master General Purpose Control Register (XBAR_MGPCR4)	32	R/W	0000_0000h	10.3.3/323
D00	Master General Purpose Control Register (XBAR_MGPCR5)	32	R/W	0000_0000h	10.3.3/323
E00	Master General Purpose Control Register (XBAR_MGPCR6)	32	R/W	0000_0000h	10.3.3/323
F00	Master General Purpose Control Register (XBAR_MGPCR7)	32	R/W	0000_0000h	10.3.3/323

10.3.1 Priority Registers Slave (XBAR_PRS*n*)

The priority registers (PRSn) set the priority of each master port on a per slave port basis and reside in each slave port. The priority register can be accessed only with 32-bit accesses. After the CRSn[RO] bit is set, the PRSn register can only be read; attempts to write to it have no effect on PRSn and result in a bus-error response to the master initiating the write.

Two available masters must not be programmed with the same priority level. Attempts to program two or more masters with the same priority level result in a bus-error response and the PRSn is not updated.

NOTE

Valid values for the M*n* priority fields depend on which masters are available on the chip. This information can be found in the chip-specific information for the crossbar.

- If the chip contains less than five masters, values 0 to 3 are valid. Writing other values will result in an error.
- If the chip contains five or more masters, valid values are 0 to n-1, where n is the number of masters attached to the Crossbar Switch. Other values will result in an error.

NOTE

See the chip-specific crossbar information for the reset value of this register.

Address: 0h base + 0h offset + (256d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	Reserved			0	Reserved			0	Reserved			0	M4		
W		—				—				—						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	M3			0	M2			0	M1			0	M0		
W																
Reset	0	0*	0*	0*	0	0*	0*	0*	0	0*	0*	0*	0	0*	0*	0*

* Notes:
- See the chip-specific crossbar information for the reset value of this register.

XBAR_PRS*n* field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 Reserved	This field is reserved.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 Reserved	This field is reserved.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–11 Reserved	This field is reserved.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 M4	Master 4 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–19 M3	Master 3 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–23 M2	Master 2 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port.

Table continues on the next page...

XBAR_PRS*n* field descriptions (continued)

Field	Description
	110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–27 M1	Master 1 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 M0	Master 0 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.

10.3.2 Control Register (XBAR_CRS*n*)

These registers control several features of each slave port and must be accessed using 32-bit accesses. After CRSn[RO] is set, the PRSn can only be read; attempts to write to it have no effect and result in an error response.

NOTE

See the chip-specific crossbar information for the reset value of this register.

Address: 0h base + 10h offset + (256d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RO	0	0													
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						ARB		0		PCTL		0	PARK		
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- See the chip-specific crossbar information for the reset value of this register.

XBAR_CRS*n* field descriptions

Field	Description
0 RO	Read Only Forces the slave port's CSRn and PRSn registers to be read-only. After set, only a hardware reset clears it. 0 The slave port's registers are writeable 1 The slave port's registers are read-only and cannot be written. Attempted writes have no effect on the registers and result in a bus error response.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 ARB	Arbitration Mode Selects the arbitration policy for the slave port. 00 Fixed priority 01 Round-robin, or rotating, priority 10 Reserved 11 Reserved
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–27 PCTL	Parking Control Determines the slave port's parking control. The low-power park feature results in an overall power savings if the slave port is not saturated. However, this forces an extra latency clock when any master tries to access the slave port while not in use because it is not parked on any master. 00 When no master makes a request, the arbiter parks the slave port on the master port defined by the PARK field 01 When no master makes a request, the arbiter parks the slave port on the last master to be in control of the slave port 10 When no master makes a request, the slave port is not parked on a master and the arbiter drives all outputs to a constant safe state 11 Reserved
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 PARK	Park Determines which master port the current slave port parks on when no masters are actively making requests and the PCTL bits are cleared.

Table continues on the next page...

XBAR_CRS*n* field descriptions (continued)

Field	Description
	NOTE: Select only master ports that are present on the chip. Otherwise, undefined behavior might occur. 000 Park on master port M0 001 Park on master port M1 010 Park on master port M2 011 Park on master port M3 100 Park on master port M4 101 Park on master port M5 110 Park on master port M6 111 Park on master port M7

10.3.3 Master General Purpose Control Register (XBAR_MGPCR*n*)

The MGPCR controls only whether the master's undefined length burst accesses are allowed to complete uninterrupted or whether they can be broken by requests from higher priority masters. The MGPCR can be accessed only in Supervisor mode with 32-bit accesses.

NOTE

If there are fewer than eight master ports, only the registers associated with those masters are present. Register addresses associated with master ports that are not present are reserved. Please see this module's chip-specific information for which master ports are present on this module.

Address: 0h base + 800h offset + (256d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																													AULB		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBAR_MGPCR*n* field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 AULB	Arbitrates On Undefined Length Bursts Determines whether, and when, the crossbar switch arbitrates away the slave port the master owns when the master is performing undefined length burst accesses. 000 No arbitration is allowed during an undefined length burst 001 Arbitration is allowed at any time during an undefined length burst 010 Arbitration is allowed after four beats of an undefined length burst 011 Arbitration is allowed after eight beats of an undefined length burst

Table continues on the next page...

XBAR_MGPCR*n* field descriptions (continued)

Field	Description
	100 Arbitration is allowed after 16 beats of an undefined length burst 101 Reserved 110 Reserved 111 Reserved

10.4 Functional Description

10.4.1 General operation

When a master accesses the crossbar switch, the access is immediately taken. If the targeted slave port of the access is available, then the access is immediately presented on the slave port. Single-clock or zero-wait-state accesses are possible through the crossbar. If the targeted slave port of the access is busy or parked on a different master port, the requesting master simply sees wait states inserted until the targeted slave port can service the master's request. The latency in servicing the request depends on each master's priority level and the responding slave's access time.

Because the crossbar switch appears to be just another slave to the master device, the master device has no knowledge of whether it actually owns the slave port it is targeting. While the master does not have control of the slave port it is targeting, it simply waits.

A master is given control of the targeted slave port only after a previous access to a different slave port completes, regardless of its priority on the newly targeted slave port. This prevents deadlock from occurring when:

- A higher priority master has:
  - An outstanding request to one slave port that has a long response time and
  - A pending access to a different slave port, and
- A lower priority master is also making a request to the same slave port as the pending access of the higher priority master.

After the master has control of the slave port it is targeting, the master remains in control of the slave port until it relinquishes the slave port by running an IDLE cycle or by targeting a different slave port for its next access.

The master can also lose control of the slave port if another higher-priority master makes a request to the slave port. However, if the master is running a fixed- or undefined-length burst transfer it retains control of the slave port until that transfer completes. Based on

MGPCR[AULB], the master either retains control of the slave port when doing undefined-length, incrementing burst transfers or loses the bus to a higher-priority master.

The crossbar terminates all master IDLE transfers, as opposed to allowing the termination to come from one of the slave buses. Additionally, when no master is requesting access to a slave port, the crossbar drives IDLE transfers onto the slave bus, even though a default master may be granted access to the slave port.

When a slave bus is being idled by the crossbar, it can park the slave port on the master port indicated by CRS*n*[PARK] . This is done to save the initial clock of arbitration delay that otherwise would be seen if the same master had to arbitrate to gain control of the slave port. The slave port can also be put into Low Power Park mode to save power, by using CRS*n*[PCTL].

10.4.2 Register coherency

The operation of the crossbar is affected as soon as a register is written. The values of the registers do not track with slave-port-related master accesses, but instead track only with slave accesses.

The MGPCRx[AULB] bits are the exception to this rule. The update of these bits is only recognized when the master on that master port runs an IDLE cycle, even though the slave bus cycle to write them will have already terminated successfully. If the MGPCRx[AULB] bits are written between two burst accesses, the new AULB encodings do not take effect until an IDLE cycle is initiated by the master on that master port.

10.4.3 Arbitration

The crossbar switch supports two arbitration algorithms:

- Fixed priority
- Round-robin

The arbitration scheme is independently programmable for each slave port.

10.4.3.1 Arbitration during undefined length bursts

Arbitration points during an undefined length burst are defined by the current master's MGPCR[AULB] field setting. When a defined length is imposed on the burst via the AULB bits, the undefined length burst is treated as a single or series of single back-to-back fixed-length burst accesses.

The following figure illustrates an example:

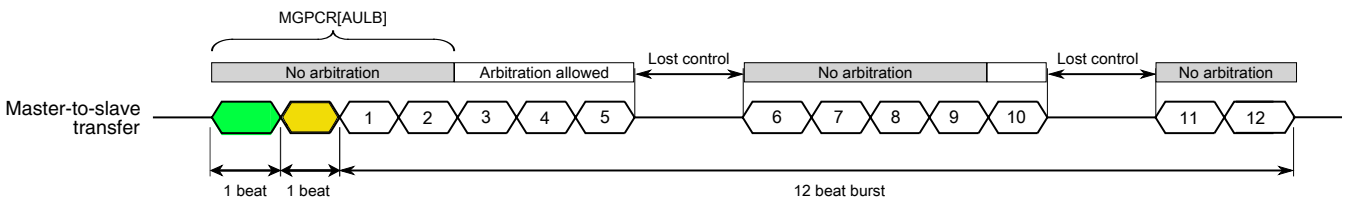


Figure 10-30. Undefined length burst example

In this example, a master runs an undefined length burst and the MGPCR[AULB] bits indicate arbitration occurs after the fourth beat of the burst. The master runs two sequential beats and then starts what will be a 12-beat undefined length burst access to a new address within the same slave port region as the previous access. The crossbar does not allow an arbitration point until the fourth overall access, or the second beat of the second burst. At that point, all remaining accesses are open for arbitration until the master loses control of the slave port.

Assume the master loses control of the slave port after the fifth beat of the second burst. After the master regains control of the slave port no arbitration point is available until after the master has run four more beats of its burst. After the fourth beat of the now continued burst, or the ninth beat of the second burst from the master's perspective, is taken, all beats of the burst are once again open for arbitration until the master loses control of the slave port.

Assume the master again loses control of the slave port on the fifth beat of the third now continued burst, or the 10th beat of the second burst from the master's perspective. After the master regains control of the slave port, it is allowed to complete its final two beats of its burst without facing arbitration.

Note

Fixed-length burst accesses are not affected by the AULB bits. All fixed-length burst accesses lock out arbitration until the last beat of the fixed-length burst.

10.4.3.2 Fixed-priority operation

When operating in fixed-priority mode, each master is assigned a unique priority level in the priority registers (PRS*n*). If two masters request access to the same slave port, the master with the highest priority in the selected priority register gains control over the slave port.

NOTE

In this arbitration mode, a higher-priority master can monopolize a slave port, preventing accesses from any lower-priority master to the port.

When a master makes a request to a slave port, the slave port checks whether the new requesting master's priority level is higher than that of the master that currently has control over the slave port, unless the slave port is in a parked state. The slave port performs an arbitration check at every clock edge to ensure that the proper master, if any, has control of the slave port.

The following table describes possible scenarios based on the requesting master port:

Table 10-35. How the Crossbar Switch grants control of a slave port to a master

When	Then the Crossbar Switch grants control to the requesting master
Both of the following are true: • The current master is not running a transfer. • The new requesting master's priority level is higher than that of the current master.	At the next clock edge
Both of the following are true: • The current master is running a fixed length burst transfer or a locked transfer. • The requesting master's priority level is higher than that of the current master.	At the end of the burst transfer or locked transfer
Both of the following are true: • The current master is running an undefined length burst transfer. • The requesting master's priority level is higher than that of the current master.	At the next arbitration point for the undefined length burst transfer **NOTE:** Arbitration points for an undefined length burst are defined in the MGPCR for each master.
The requesting master's priority level is lower than the current master.	At the conclusion of one of the following cycles: • An IDLE cycle • A non-IDLE cycle to a location other than the current slave port

10.4.3.3 Round-robin priority operation

When operating in round-robin mode, each master is assigned a relative priority based on the master port number. This relative priority is compared to the master port number (ID) of the last master to perform a transfer on the slave bus. The highest priority requesting master becomes owner of the slave bus at the next transfer boundary, accounting for locked and fixed-length burst transfers. Priority is based on how far ahead the ID of the requesting master is to the ID of the last master.

After granted access to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line is granted access to the slave port at the next transfer boundary, or possibly on the next clock cycle if the current master has no pending access request.

As an example of arbitration in round-robin mode, assume the crossbar is implemented with master ports 0, 1, 4, and 5. If the last master of the slave port was master 1, and master 0, 4, and 5 make simultaneous requests, they are serviced in the order: 4 then 5 then 0.

Parking may continue to be used in a round-robin mode, but does not affect the round-robin pointer unless the parked master actually performs a transfer. Handoff occurs to the next master in line after one cycle of arbitration. If the slave port is put into low-power park mode, the round-robin pointer is reset to point at master port 0, giving it the highest priority.

10.4.3.4 Priority assignment

Each master port must be assigned a unique 3-bit priority level. If an attempt is made to program multiple master ports with the same priority level within the priority registers (PRS*n*), the crossbar switch responds with a bus error and the registers are not updated.

10.5 Initialization/application information

No initialization is required for the crossbar switch.

Hardware reset ensures all the register bits used by the crossbar switch are properly initialized to a valid state. However, settings and priorities may be programmed to achieve maximum system performance.

Chapter 11
Crossbar Integrity Checker (XBIC)

11.1 Overview

The Crossbar Integrity Checker (XBIC) verifies the integrity of the crossbar transfers.

11.2 Features

The XBIC has the following features:

- Verification of attribute information for all crossbar transfers
  - EDC (72,64) code protects against single and double bit errors
- Verification of feedback information for each data phase during crossbar transfer
- Error injection for testing
  - Programmable master and slave port specifiers
  - Programmable 8-bit toggle vector to insert error in master EDC checkbit value
  - Address, EDC syndrome, master and slave port information captured on error
- Programmable integrity check enable on a per-slave-port basis
- Programmable integrity feedback check enable on a per-master-port basis

11.3 Block diagram

The crossbar transfer attribute information for all master and slave ports is routed to the XBIC, which calculates and checks the EDC parity over the attribute information as shown in the following diagram.

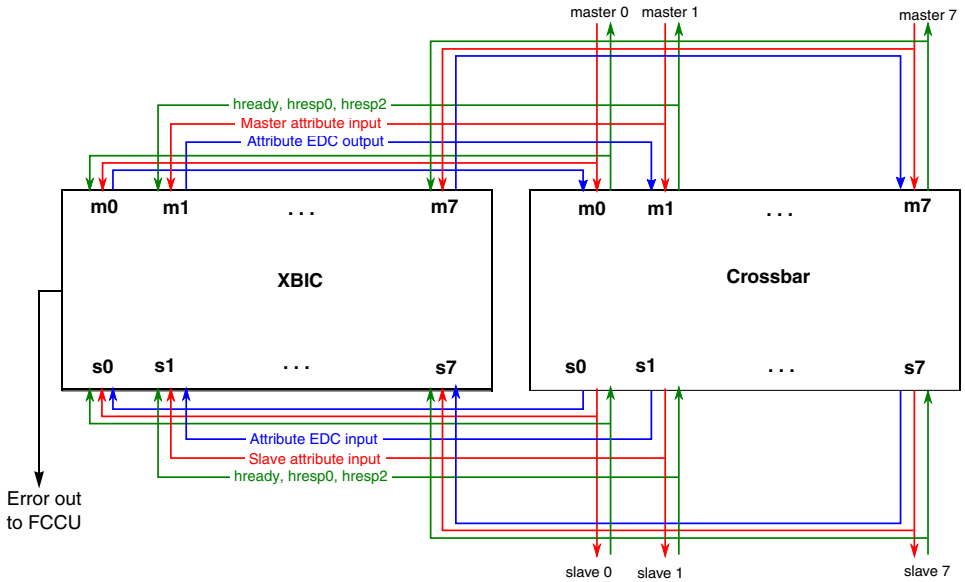


Figure 11-1. XBIC system block diagram

11.4 External signal description

The XBIC has no external interface signals.

11.5 Memory map and register definition

The XBIC programming model has four 32-bit registers. The programming model can only be accessed in supervisor mode using 32-bit (word) accesses. Attempted references with a different access size, to an undefined (reserved) address, with a non-supported access type (a write to a read-only register), or in user mode generate an error termination.

XBIC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	XBIC Module Control Register (XBIC_MCR)	32	R/W	FFFF_0000h	11.5.1/331
4	XBIC Error Injection Register (XBIC_EIR)	32	R/W	0000_0000h	11.5.2/333
8	XBIC Error Status Register (XBIC_ESR)	32	R	0000_0000h	11.5.3/333
C	XBIC Error Address Register (XBIC_EAR)	32	R	0000_0000h	11.5.4/336

11.5.1 XBIC Module Control Register (XBIC_MCR)

The XBIC_MCR allows attribute integrity checking to be enabled on a per-port basis.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SE0	SE1	SE2	SE3	SE4	SE5	SE6	SE7	ME0	ME1	ME2	ME3	ME4	ME5	ME6	ME7
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBIC_MCR field descriptions

Field	Description
0 SE0	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 0. 1 Attribute integrity checking enabled for slave port 0.
1 SE1	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 1. 1 Attribute integrity checking enabled for slave port 1.
2 SE2	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 2. 1 Attribute integrity checking enabled for slave port 2.
3 SE3	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 3. 1 Attribute integrity checking enabled for slave port 3.
4 SE4	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 4. 1 Attribute integrity checking enabled for slave port 4.
5 SE5	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 5. 1 Attribute integrity checking enabled for slave port 5.
6 SE6	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 6. 1 Attribute integrity checking enabled for slave port 6.
7 SE7	Slave Port Enable for EDC error detect. 0 Attribute integrity checking disabled for slave port 7. 1 Attribute integrity checking enabled for slave port 7.

Table continues on the next page...

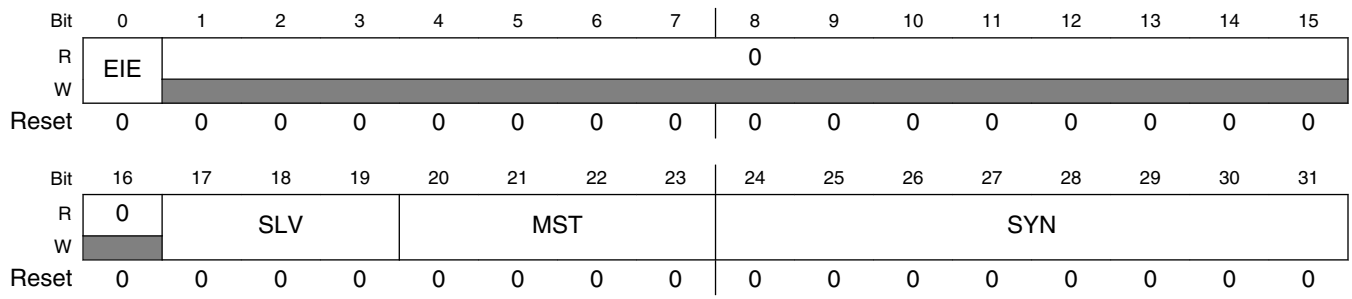
XBIC_MCR field descriptions (continued)

Field	Description
8 ME0	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 0. 1 Attribute integrity checking enabled for master port 0.
9 ME1	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 1. 1 Attribute integrity checking enabled for master port 1.
10 ME2	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 2. 1 Attribute integrity checking enabled for master port 2.
11 ME3	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 3. 1 Attribute integrity checking enabled for master port 3.
12 ME4	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 4. 1 Attribute integrity checking enabled for master port 4.
13 ME5	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 5. 1 Attribute integrity checking enabled for master port 5.
14 ME6	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 6. 1 Attribute integrity checking enabled for master port 6.
15 ME7	Master Port Enable for slave driven signal safety check. 0 Attribute integrity checking disabled for master port 7. 1 Attribute integrity checking enabled for master port 7.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

11.5.2 XBIC Error Injection Register (XBIC_EIR)

The XBIC_EIR contains fields for controlling the XBIC error injection function. When an error is injected, the EDC syndrome associated with the programmed master and slave ports is modified, causing the XBIC to assert the error indication to the FCCU and capturing the transfer information in the XBIC_ESR and XBIC_EAR registers. Otherwise, transfers are not affected by XBIC error injection.

Address: 0h base + 4h offset = 4h



XBIC_EIR field descriptions

Field	Description
0 EIE	Error Injection Enable. 0 Error injection disabled 1 Error injection enabled
1–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–19 SLV	Target Slave Port. Error injection is enabled for the designated slave port. Other slave ports are unaffected.
20–23 MST	Target Master ID. Error injection is enabled for the designated master ID. Transfers with other master IDs are not affected.
24–31 SYN	Syndrome. This value is exclusive-ored with the calculated syndrome (which should be zero) to generate an error with the specified syndrome. A value of zero does not generate an error.

11.5.3 XBIC Error Status Register (XBIC_ESR)

The Error Status Register (XBIC_ESR) contains two types of error information about the last transfer. If an attribute integrity check error was detected, the slave port, the master port, and the syndrome are captured in the SLV, MST, and SYN fields. If there is a mismatch among internal signals (hready, hresp0, and hresp2) during the data phase, the slave and master port are catured in the DPSE0-7 and DPME0-7 fields.

This register is cleared only on reset.

NOTE

When the XBIC detects back-to-back faults on a system bus path through the crossbar switch, the fault information captured in the XBIC_ESR does not correspond to the initial fault event but rather the subsequent fault event. While the fault event is properly detected, diagnostic status information in the XBIC_ESR register describing the initial fault event is lost. This can only occur in the event of a series of bus transactions targeting the same crossbar slave target where the series of bus transactions are not separated by idle or stall cycles.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	DPSE0	DPSE1	DPSE2	DPSE3	DPSE4	DPSE5	DPSE6	DPSE7	DPME0	DPME1	DPME2	DPME3	DPME4	DPME5	DPME6
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DPME7	SLV			MST				SYN							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBIC_ESR field descriptions

Field	Description
0 VLD	Error Status Valid. 0 No error detected, other fields of the ESR and EAR are not valid. 1 Error detected, all fields of the ESR and EAR are valid.
1 DPSE0	Data phase slave port error.

Table continues on the next page...

XBIC_ESR field descriptions (continued)

Field	Description
	0 No error on slave port 0 1 Data phase error on slave port 0
2 DPSE1	Data phase slave port error. 0 No error on slave port 1 1 Data phase error on slave port 1
3 DPSE2	Data phase slave port error. 0 No error on slave port 2 1 Data phase error on slave port 2
4 DPSE3	Data phase slave port error. 0 No error on slave port 3 1 Data phase error on slave port 3
5 DPSE4	Data phase slave port error. 0 No error on slave port 4 1 Data phase error on slave port 4
6 DPSE5	Data phase slave port error. 0 No error on slave port 5 1 Data phase error on slave port 5
7 DPSE6	Data phase slave port error. 0 No error on slave port 6 1 Data phase error on slave port 6
8 DPSE7	Data phase slave port error. 0 No error on slave port 7 1 Data phase error on slave port 7
9 DPME0	Data phase master port error. 0 No error on master port 0 1 Data phase error on master port 0
10 DPME1	Data phase master port error. 0 No error on master port 1 1 Data phase error on master port 1
11 DPME2	Data phase master port error. 0 No error on master port 2 1 Data phase error on master port 2
12 DPME3	Data phase master port error. 0 No error on master port 3 1 Data phase error on master port 3
13 DPME4	Data phase master port error.

Table continues on the next page...

XBIC_ESR field descriptions (continued)

Field	Description
	0 No error on master port 4 1 Data phase error on master port 4
14 DPME5	Data phase master port error. 0 No error on master port 5 1 Data phase error on master port 5
15 DPME6	Data phase master port error. 0 No error on master port 6 1 Data phase error on master port 6
16 DPME7	Data phase master port error. 0 No error on master port 7 1 Data phase error on master port 7
17–19 SLV	Slave Port. The slave port targeted by the last transfer with an error.
20–23 MST	Master ID. The master ID value of the last transfer with an error.
24–31 SYN	Syndrome. The syndrome calculated for the last transfer with an error. For single bit errors the signal in error can be determined, see Table 11-6 .

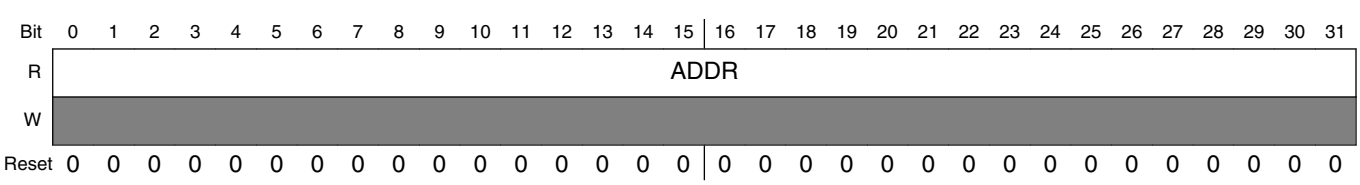
11.5.4 XBIC Error Address Register (XBIC_EAR)

The Error Address Register (XBIC_EAR) contains the address of the first transfer in which an attribute integrity check error was detected on an enabled slave port. This register is only cleared on reset.

NOTE

When the XBIC detects back-to-back faults on a system bus path through the crossbar switch, the fault information captured in the XBIC_EAR does not correspond to the initial fault event but rather the subsequent fault event. While the fault event is properly detected, diagnostic status information in the XBIC_ESR register describing the initial fault event is lost. This can only occur in the event of a series of bus transactions targeting the same crossbar slave target where the series of bus transactions are not separated by idle or stall cycles.

Address: 0h base + Ch offset = Ch



XBIC_EAR field descriptions

Field	Description
0–31 ADDR	The address of the last transfer with an error.

11.6 Functional description

The Crossbar Integrity Checker (XBIC) verifies the integrity of the crossbar interface. The Module Control Register (XBIC_MCR) allows integrity checking to be enabled on a individual port basis. When an error is detected, it is captured in XBIC_ESR register and XBIC also sends out an error signal to the Fault Collection and Control Unit (FCCU). Detection of an error does not generate a bus error. The XBIC integrity checking is independent from the end-to-end ECC that covers the transfer address and data.

During the AHB address phase, the crossbar attribute information is checked using an 8-bit Error Detection Code (EDC), which detects any single- or double-bit errors. When an error is detected, due either to hardware fault or error injection, information related to the error is captured and reported in XBIC_ESR and XBIC_EAR. The cause of a single-bit error can be determined by the syndrome value reported in the XBIC_ESR as shown in Table 11-6. Any other syndrome indicates a multi-bit error.

During the AHB data phase, the responding signals from the slave to master are checked when passing through the crossbar. When an error is detected, the information is captured in XBIC_ESR DSME0-7 field.

The XBIC can also be programmed to intentionally inject EDC errors for testing. Error injection is targeted toward a single slave port and a single master ID at a time as determined by the settings in the Error Injection Register (XBIC_EIR). When an error is injected, the EDC syndrome is modified which causes the XBIC to assert the error indication to the FCCU. Otherwise transfers are unaffected by XBIC error injection. This approach allows the check logic to be verified without compromising the integrity of the data transfer. Once the error injection function is enabled by setting XBIC_EIR[EIE], errors are induced on all subsequent targeted transactions until XBIC_EIR[EIE] is cleared

Table 11-6. Hexadecimal attribute single-bit error syndromes

Signal	SYN	Signal	SYN	Signal	SYN	Signal	SYN
hwrite	07	hbstrb[7]	70	hdecor[31]	25	hdecor[15]	23
htrans[0]	0b	hbstrb[6]	32	hdecor[30]	68	hdecor[14]	51
hsize[2]	0d	hbstrb[5]	52	hdecor[29]	c7	hdecor[13]	54

Table continues on the next page...

Table 11-6. Hexadecimal attribute single-bit error syndromes (continued)

Signal	SYN	Signal	SYN	Signal	SYN	Signal	SYN
hsize[1]	0e	hbstrb[4]	a8	hdecor[28]	83	hdecor[12]	61
hsize[0]	13	hbstrb[3]	43	hdecor[27]	85	hdecor[11]	e3
hprot[5]	15	hbstrb[2]	45	hdecor[26]	86	hdecor[10]	e6
hprot[4]	16	hbstrb[1]	4c	hdecor[25]	89	hdecor[9]	f8
hprot[3]	19	hbstrb[0]	a4	hdecor[24]	8a	hdecor[8]	38
hprot[2]	1a	hmaster[3]	a2	hdecor[23]	8c	hdecor[7]	58
hprot[1]	1c	hmaster[2]	b0	hdecor[22]	49	hdecor[6]	37
hprot[0]	91	hmaster[1]	c1	hdecor[21]	92	hdecor[5]	f1
hburst[2]	a1	hmaster[0]	c2	hdecor[20]	94	hdecor[4]	3b
hburst[1]	64	hslave[2]	c4	hdecor[19]	98	hdecor[3]	3d
hburst[0]	29	hslave[1]	c8	hdecor[18]	46	hdecor[2]	3e
hmastlock	2a	hslave[0]	d0	hdecor[17]	34	hdecor[1]	4f
hunalign	2c	hdecorated	e0	hdecor[16]	4a	hdecor[0]	6e
edc[7]	80	edc[6]	40	edc[5]	20	edc[4]	10
edc[3]	08	edc[2]	04	edc[1]	02	edc[0]	01

Chapter 12
Peripheral Bridge (AIPS-Lite)

12.1 Peripheral Bridge chip-specific module information

This section summarizes how the AIPS-Lite modules have been configured in the chip.

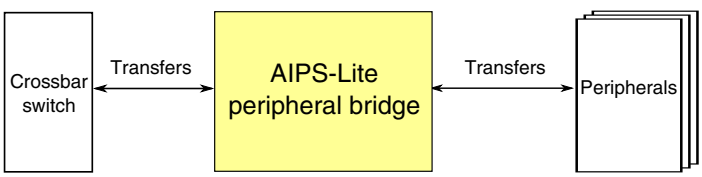


Figure 12-1. Peripheral bridge configuration

12.1.1 Number of peripheral bridges

This device contains two identical peripheral bridges, PBRIDGE_0 and PBRIDGE_1.

NOTE

Peripheral bridge instances PBRIDGE_0 and PBRIDGE_1 may also be referred to as AIPS0 and AIPS1, respectively, elsewhere in this document.

12.1.2 Memory maps

The peripheral bridges are used to access the registers of most of the modules on this chip. See the Peripheral memory map for the memory slot assignment for each module.

12.1.3 MPRA registers

The AIPS0_MPRA and AIPS1_MPRA registers contain the access-privilege fields for each master on this chip. The access-privilege field number corresponds to the logical master ID assigned to each master. Master 8 (Core 0 Debug) and master 9 (Core 1 Debug) share the same access privileges as master 0 (Core 0) and master 1 (Core 1), respectively. See System Memory Protection Unit (SMPU) chip-specific module information for the master port assignments for this chip.

The reset value for the AIPS0_MPRA and AIPS1_MPRA registers is 0x7777_7777.

12.1.4 PACR registers

Each peripheral on this chip is assigned to a 4-bit PACR field, PACR*n*, within the PACR*x* registers. See Peripheral memory map for the peripheral slot assignments for this chip. Unused PACR*n* fields are reserved.

The AIPS*n*_PACR*x* reset values depend on whether the peripheral is available on this chip. For each populated slot in Peripheral memory map, the corresponding peripheral's PACR field resets to 0x4.

12.2 Introduction

The peripheral bridge converts the crossbar switch interface to an interface that can access most of the slave peripherals on this chip.

The peripheral bridge occupies 64 MB of the address space, which is divided into peripheral slots of 16 KB. (It might be possible that all the peripheral slots are not used. See the memory map chapter for details on slot assignments.) The bridge includes separate clock enable inputs for each of the slots to accommodate slower peripherals.

12.2.1 Features

Key features of the peripheral bridge are:

- Supports peripheral slots with 8-, 16-, and 32-bit datapath width
- Programming model provides memory protection functionality

12.2.2 General operation

The slave devices connected to the peripheral bridge are modules which contain a programming model of control and status registers. The system masters read and write these registers through the peripheral bridge.

The register maps of the peripherals are located on 16 KB boundaries. Each peripheral is allocated one or more 16-KB block(s) of the memory map.

12.3 Memory map/register definition

The 32-bit peripheral bridge registers can be accessed only in supervisor mode by trusted bus masters. Additionally, these registers must be read from or written to only by a 32-bit aligned access. The peripheral bridge registers are mapped into the Peripheral Access Control Register A PACRA[PACR0] address space.

AIPS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Master Privilege Register A (AIPS_MPRA)	32	R/W	See section	12.3.1/343
100	Peripheral Access Control Register (AIPS_PACRA)	32	R/W	See section	12.3.2/346
104	Peripheral Access Control Register (AIPS_PACRB)	32	R/W	See section	12.3.2/346
108	Peripheral Access Control Register (AIPS_PACRC)	32	R/W	See section	12.3.2/346
10C	Peripheral Access Control Register (AIPS_PACRD)	32	R/W	See section	12.3.2/346
110	Peripheral Access Control Register (AIPS_PACR_Reserved)	32	R/W	See section	12.3.2/346
114	Peripheral Access Control Register (AIPS_PACRF)	32	R/W	See section	12.3.2/346
118	Peripheral Access Control Register (AIPS_PACRG)	32	R/W	See section	12.3.2/346
11C	Peripheral Access Control Register (AIPS_PACRH)	32	R/W	See section	12.3.2/346
140	Off-Platform Peripheral Access Control Register (AIPS_OPACRA)	32	R/W	See section	12.3.3/351
144	Off-Platform Peripheral Access Control Register (AIPS_OPACRB)	32	R/W	See section	12.3.3/351
148	Off-Platform Peripheral Access Control Register (AIPS_OPACRC)	32	R/W	See section	12.3.3/351
14C	Off-Platform Peripheral Access Control Register (AIPS_OPACRD)	32	R/W	See section	12.3.3/351
150	Off-Platform Peripheral Access Control Register (AIPS_OPACRE)	32	R/W	See section	12.3.3/351
154	Off-Platform Peripheral Access Control Register (AIPS_OPACRF)	32	R/W	See section	12.3.3/351
158	Off-Platform Peripheral Access Control Register (AIPS_OPACRG)	32	R/W	See section	12.3.3/351

Table continues on the next page...

AIPS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
15C	Off-Platform Peripheral Access Control Register (AIPS_OPACRH)	32	R/W	See section	12.3.3/351
160	Off-Platform Peripheral Access Control Register (AIPS_OPACRI)	32	R/W	See section	12.3.3/351
164	Off-Platform Peripheral Access Control Register (AIPS_OPACRJ)	32	R/W	See section	12.3.3/351
168	Off-Platform Peripheral Access Control Register (AIPS_OPACRK)	32	R/W	See section	12.3.3/351
16C	Off-Platform Peripheral Access Control Register (AIPS_OPACRL)	32	R/W	See section	12.3.3/351
170	Off-Platform Peripheral Access Control Register (AIPS_OPACRM)	32	R/W	See section	12.3.3/351
174	Off-Platform Peripheral Access Control Register (AIPS_OPACRN)	32	R/W	See section	12.3.3/351
178	Off-Platform Peripheral Access Control Register (AIPS_OPACRO)	32	R/W	See section	12.3.3/351
17C	Off-Platform Peripheral Access Control Register (AIPS_OPACRP)	32	R/W	See section	12.3.3/351
180	Off-Platform Peripheral Access Control Register (AIPS_OPACRQ)	32	R/W	See section	12.3.3/351
184	Off-Platform Peripheral Access Control Register (AIPS_OPACRR)	32	R/W	See section	12.3.3/351
188	Off-Platform Peripheral Access Control Register (AIPS_OPACRS)	32	R/W	See section	12.3.3/351
18C	Off-Platform Peripheral Access Control Register (AIPS_OPACRT)	32	R/W	See section	12.3.3/351
190	Off-Platform Peripheral Access Control Register (AIPS_OPACRU)	32	R/W	See section	12.3.3/351
194	Off-Platform Peripheral Access Control Register (AIPS_OPACRV)	32	R/W	See section	12.3.3/351
198	Off-Platform Peripheral Access Control Register (AIPS_OPACRW)	32	R/W	See section	12.3.3/351
19C	Off-Platform Peripheral Access Control Register (AIPS_OPACRX)	32	R/W	See section	12.3.3/351
1A0	Off-Platform Peripheral Access Control Register (AIPS_OPACRY)	32	R/W	See section	12.3.3/351
1A4	Off-Platform Peripheral Access Control Register (AIPS_OPACRZ)	32	R/W	See section	12.3.3/351
1A8	Off-Platform Peripheral Access Control Register (AIPS_OPACRAA)	32	R/W	See section	12.3.3/351
1AC	Off-Platform Peripheral Access Control Register (AIPS_OPACRAB)	32	R/W	See section	12.3.3/351
1B0	Off-Platform Peripheral Access Control Register (AIPS_OPACRAC)	32	R/W	See section	12.3.3/351

Table continues on the next page...

AIPS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1B4	Off-Platform Peripheral Access Control Register (AIPS_OPACRAD)	32	R/W	See section	12.3.3/351
1B8	Off-Platform Peripheral Access Control Register (AIPS_OPACRAE)	32	R/W	See section	12.3.3/351
1BC	Off-Platform Peripheral Access Control Register (AIPS_OPACRAF)	32	R/W	See section	12.3.3/351

12.3.1 Master Privilege Register A (AIPS_MPRA)

This register specifies eight identical 4-bit fields defining the access-privilege level associated with a bus master in the device to various peripherals. The register provides one field per bus master. Each master is assigned a logical master number. See the device configuration information for details.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MTR0	MTW0	MPL0	0	MTR1	MTW1	MPL1	Reserved				0	MTR3	MTW3	MPL3
W																
Reset	0*	1*	1*	1*	0*	0*	0*	0*	0	0	0	0	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MTR4	MTW4	MPL4	Reserved				Reserved				0	MTR7	MTW7	MPL7
W																
Reset	0*	0*	0*	0*	0	0	0	0	0	0	0	0	0*	0*	0*	0*

* Notes:
- Reset value is chip-dependent. See the AIPS chip-specific information.

AIPS_MPRA field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 MTR0	Master 0 Trusted For Read Determines whether the master is trusted for read accesses.

Table continues on the next page...

AIPS_MPRA field descriptions (continued)

Field	Description
	0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
2 MTW0	Master 0 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
3 MPL0	Master 0 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 MTR1	Master 1 trusted for read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
6 MTW1	Master 1 trusted for writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
7 MPL1	Master 1 privilege level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
8–11 Reserved	This field is reserved.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 MTR3	Master 3 Trusted For Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
14 MTW3	Master 3 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.

Table continues on the next page...

AIPS_MPRA field descriptions (continued)

Field	Description
15 MPL3	Master 3 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 MTR4	Master 4 Trusted For Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
18 MTW4	Master 4 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
19 MPL4	Master 4 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
20–23 Reserved	This field is reserved.
24–27 Reserved	This field is reserved.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 MTR7	Master 7 trusted for read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
30 MTW7	Master 7 trusted for writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
31 MPL7	Master 7 privilege level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.

12.3.2 Peripheral Access Control Register (AIPS_PACR*n*)

Each of the on-platform peripherals has a 4-bit PACR *n* field, which defines the access levels supported by the given module. Eight PACR fields are grouped together to form a 32-bit PACR *x* register.

The peripheral assignments to each PACR are defined by the memory map slot that the peripherals are assigned. See the device's memory map details for the assignments for a particular device. If a peripheral is absent, the corresponding PACR field is not implemented. Reads to the location return zeroes and writes are ignored.

The following table shows the top-level structure of PACRs.

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x100	PACRA	PACR0	PACR1	PACR2	PACR3	PACR4	PACR5	PACR6	PACR7
0x104	PACRB	PACR8	PACR9	PACR10	PACR11	PACR12	PACR13	PACR14	PACR15
0x108	PACRC	PACR16	PACR17	PACR18	PACR19	PACR20	PACR21	PACR22	PACR23
0x10C	PACRD	PACR24	PACR25	PACR26	PACR27	PACR28	PACR29	PACR30	PACR31
0x114	PACRF	PACR40	PACR41	PACR42	PACR43	PACR44	PACR45	PACR46	PACR47
0x118	PACRG	PACR48	PACR49	PACR50	PACR51	PACR52	PACR53	PACR54	PACR55
0x11C	PACRH	PACR56	PACR57	PACR58	PACR59	PACR60	PACR61	PACR62	PACR63

Address: 0h base + 100h offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	SP0	WP0	TP0	0	SP1	WP1	TP1	0	SP2	WP2	TP2	0	SP3	WP3	TP3
W																
Reset	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SP4	WP4	TP4	0	SP5	WP5	TP5	0	SP6	WP6	TP6	0	SP7	WP7	TP7
W																
Reset	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*

* Notes:
- The reset value of PACR0 is 0101b.

AIPS_PACR*n* field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 SP0	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
	field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
2 WP0	Write Protect Determines whether the peripheral allows write accesss. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
3 TP0	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SP1	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
6 WP1	Write Protect Determines whether the peripheral allows write accessses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
7 TP1	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 SP2	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
	field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
10 WP2	Write Protect Determines whether the peripheral allows write accesss. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
11 TP2	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SP3	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this bit is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control bit for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
14 WP3	Write Protect Determines whether the peripheral allows write accessses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
15 TP3	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 SP4	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
	field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
18 WP4	Write Protect Determines whether the peripheral allows write accesss. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
19 TP4	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 SP5	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
22 WP5	Write Protect Determines whether the peripheral allows write accessses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
23 TP5	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 SP6	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
	field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
26 WP6	Write Protect Determines whether the peripheral allows write accessses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
27 TP6	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 SP7	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
30 WP7	Write Protect Determines whether the peripheral allows write accessses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
31 TP7	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

12.3.3 Off-Platform Peripheral Access Control Register (AIPS_OPACR*n*)

Each of the peripherals has a 4-bit OPACR *n* field, which defines the access levels supported by this module. Eight OPACR fields are grouped together to form a 32-bit OPACR *x* register.

The peripheral assignments to each OPACR are defined by the memory map slot that the peripherals are assigned. See the device's memory map details for the assignments for a particular device. If a peripheral is absent, the corresponding OPACR field is not implemented. Reads to the location return zeroes and writes are ignored.

The following table shows the top-level structure of OPACRs.

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x140	OPACRA	OPACR0	OPACR1	OPACR2	OPACR3	OPACR4	OPACR5	OPACR6	OPACR7
0x144	OPACRB	OPACR8	OPACR9	OPACR10	OPACR11	OPACR12	OPACR13	OPACR14	OPACR15
0x148	OPACRC	OPACR16	OPACR17	OPACR18	OPACR19	OPACR20	OPACR21	OPACR22	OPACR23
0x14C	OPACRD	OPACR24	OPACR25	OPACR26	OPACR27	OPACR28	OPACR29	OPACR30	OPACR31
0x150	OPACRE	OPACR32	OPACR33	OPACR34	OPACR35	OPACR36	OPACR37	OPACR38	OPACR39
0x154	OPACRF	OPACR40	OPACR41	OPACR42	OPACR43	OPACR44	OPACR45	OPACR46	OPACR47
0x158	OPACRG	OPACR48	OPACR49	OPACR50	OPACR51	OPACR52	OPACR53	OPACR54	OPACR55
0x15C	OPACRH	OPACR56	OPACR57	OPACR58	OPACR59	OPACR60	OPACR61	OPACR62	OPACR63
0x160	OPACRI	OPACR64	OPACR65	OPACR66	OPACR67	OPACR68	OPACR69	OPACR70	OPACR71
0x164	OPACRJ	OPACR72	OPACR73	OPACR74	OPACR75	OPACR76	OPACR77	OPACR78	OPACR79
0x168	OPACRK	OPACR80	OPACR81	OPACR82	OPACR83	OPACR84	OPACR85	OPACR86	OPACR87
0x16C	OPACRL	OPACR88	OPACR89	OPACR90	OPACR91	OPACR92	OPACR93	OPACR94	OPACR95
0x170	OPACRM	OPACR96	OPACR97	OPACR98	OPACR99	OPACR100	OPACR101	OPACR102	OPACR103
0x174	OPACRN	OPACR104	OPACR105	OPACR106	OPACR107	OPACR108	OPACR109	OPACR110	OPACR111
0x178	OPACRO	OPACR112	OPACR113	OPACR114	OPACR115	OPACR116	OPACR117	OPACR118	OPACR119
0x17C	OPACRP	OPACR120	OPACR121	OPACR122	OPACR123	OPACR124	OPACR125	OPACR126	OPACR127
0x180	OPACRQ	OPACR128	OPACR129	OPACR130	OPACR131	OPACR132	OPACR133	OPACR134	OPACR135
0x184	OPACRR	OPACR136	OPACR137	OPACR138	OPACR139	OPACR140	OPACR141	OPACR142	OPACR143
0x188	OPACRS	OPACR144	OPACR145	OPACR146	OPACR147	OPACR148	OPACR149	OPACR150	OPACR151
0x18C	OPACRT	OPACR152	OPACR153	OPACR154	OPACR155	OPACR156	OPACR157	OPACR158	OPACR159

Table continues on the next page...

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x190	OPACRU	OPACR160	OPACR161	OPACR162	OPACR163	OPACR164	OPACR165	OPACR166	OPACR167
0x194	OPACRV	OPACR168	OPACR169	OPACR170	OPACR171	OPACR172	OPACR173	OPACR174	OPACR175
0x198	OPACRW	OPACR176	OPACR177	OPACR178	OPACR179	OPACR180	OPACR181	OPACR182	OPACR183
0x19C	OPACRX	OPACR184	OPACR185	OPACR186	OPACR187	OPACR188	OPACR189	OPACR190	OPACR191
0x1A0	OPACRY	OPACR192	OPACR193	OPACR194	OPACR195	OPACR196	OPACR197	OPACR198	OPACR199
0x1A4	OPACRZ	OPACR200	OPACR201	OPACR202	OPACR203	OPACR204	OPACR205	OPACR206	OPACR207
0x1A8	OPACRAA	OPACR208	OPACR209	OPACR210	OPACR211	OPACR212	OPACR213	OPACR214	OPACR215
0x1AC	OPACRAB	OPACR216	OPACR217	OPACR218	OPACR219	OPACR220	OPACR221	OPACR222	OPACR223
0x1B0	OPACRAC	OPACR224	OPACR225	OPACR226	OPACR227	OPACR228	OPACR229	OPACR230	OPACR231
0x1B4	OPACRAD	OPACR232	OPACR233	OPACR234	OPACR235	OPACR236	OPACR237	OPACR238	OPACR239
0x1B8	OPACRAE	OPACR240	OPACR241	OPACR242	OPACR243	OPACR244	OPACR245	OPACR246	OPACR247
0x1BC	OPACRAF	OPACR248	OPACR249	OPACR250	OPACR251	OPACR252	OPACR253	OPACR254	OPACR255

Address: 0h base + 140h offset + (4d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	SP0	WP0	TP0	0	SP1	WP1	TP1	0	SP2	WP2	TP2	0	SP3	WP3	TP3
W																
Reset	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SP4	WP4	TP4	0	SP5	WP5	TP5	0	SP6	WP6	TP6	0	SP7	WP7	TP7
W																
Reset	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*

* Notes:
- Reset value is chip-dependent. See the AIPS chip-specific information.

AIPS_OPACR*n* field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 SP0	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control

Table continues on the next page...

AIPS_OPACR*n* field descriptions (continued)

Field	Description
	field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
2 WP0	Write Protect Determines whether the peripheral allows write accessses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
3 TP0	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SP1	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
6 WP1	Write Protect Determines whether the peripheral allows write accessses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
7 TP1	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 SP2	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this bit is set, the master privilege level must indicate the supervisor access attributeMPR *x* [MPL *n*], and the MPR *x* [MPL

Table continues on the next page...

AIPS_OPACR*n* field descriptions (continued)

Field	Description
	n] control bit for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
10 WP2	Write Protect Determines whether the peripheral allows write accessses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
11 TP2	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SP3	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
14 WP3	Write Protect Determines whether the peripheral allows write accesss. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
15 TP3	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 SP4	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this bit is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control bit for

Table continues on the next page...

AIPS_OPACR*n* field descriptions (continued)

Field	Description
	the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
18 WP4	Write Protect Determines whether the peripheral allows write accessses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
19 TP4	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 SP5	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
22 WP5	Write Protect Determines whether the peripheral allows write accessses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
23 TP5	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 SP6	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control

Table continues on the next page...

AIPS_OPACR*n* field descriptions (continued)

Field	Description
	field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
26 WP6	Write Protect Determines whether the peripheral allows write accessses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
27 TP6	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 SP7	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
30 WP7	Write Protect Determines whether the peripheral allows write accessses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
31 TP7	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

12.4 Functional description

The peripheral bridge functions as a bus protocol translator between the crossbar switch and the slave peripheral bus.

The peripheral bridge manages all transactions destined for the attached slave devices and generates select signals for modules on the peripheral bus by decoding accesses within the attached address space.

12.4.1 Access support

All combinations of access size and peripheral data port width are supported. An access that is larger than the target peripheral's data width will be decomposed to multiple, smaller accesses. Bus decomposition is terminated by a transfer error caused by an access to an empty register area.

Chapter 13
System Memory Protection Unit (SMPU)

13.1 System Memory Protection Unit (SMPU) chip-specific module information

This chip contains two instances of the System Memory Protection Unit (SMPU):

- SMPU0 for the data XBAR (XBAR_0)
- SMPU1 for the instruction XBAR (XBAR_1)

The following sections explain how each SMPU is configured on this chip.

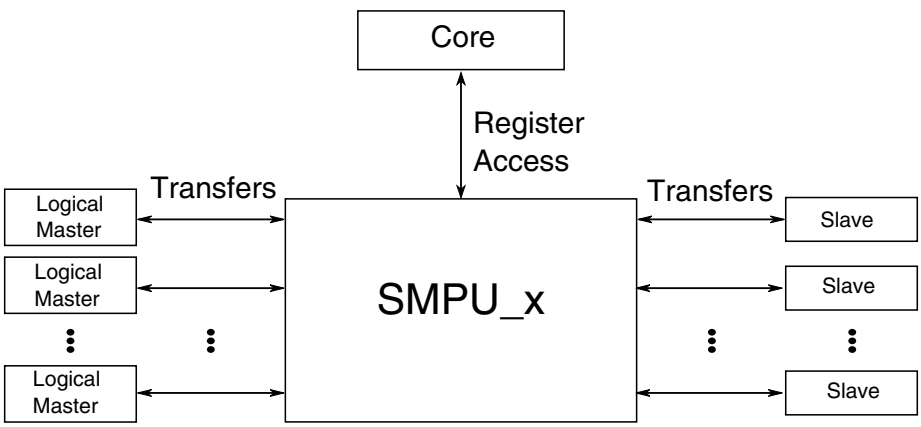


Figure 13-1. SMPUx configuration

13.1.1 SMPUx configuration

This section summarizes how the module instance has been configured in the chip.

13.1.1.1 SMPUx Logical bus Master Assignments

The logical bus master assignments for the SMPUx are shown in the following table.

Table 13-1. Master IDs

Master Name	Master ID
Core0	0
Core1	1
Reserved	2
DMA0	3
Ethernet	4
Reserved	5
Reserved	6
Zipwire	7
Core0 Debug	8
Core1 Debug	9
Reserved	10
Reserved	11
Reserved	12
Reserved	13
Reserved	14
Reserved	15

13.1.1.2 Number of region descriptors supported by SMPU0 and SMPU1

- SMPU0 supports a maximum of 12 region descriptors.
- SMPU1 supports a maximum of eight region descriptors.

13.2 Overview

The System Memory Protection Unit (SMPU) provides hardware access control for system bus memory references. The SMPU concurrently monitors system bus transactions and evaluates their appropriateness using preprogrammed region descriptors that define memory spaces and their access rights. Memory references that have sufficient access control rights are allowed to complete, while references that are not mapped to any region descriptor or have insufficient rights are terminated with an access error response.

13.2.1 Block diagram

A simplified block diagram of the SMPU module is shown in the following figure. The access signals coming from the crossbar switch slave ports are shown on the left. The access evaluation macros and region descriptors are in the middle, and the peripheral bus interface is on the right side.

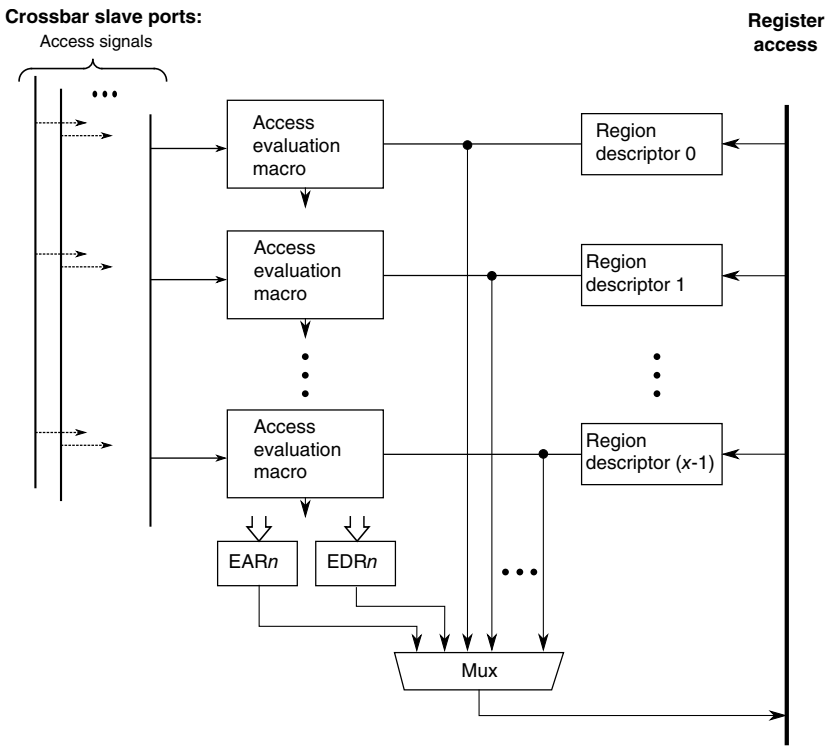


Figure 13-2. SMPU block diagram

The access evaluation macros determine hits to memory regions and detect violations to access protections. If a violation is detected, the macros update the appropriate master's error registers. For details of the access evaluation macro, see Access evaluation macro.

13.2.2 Features

The SMPU feature set includes:

- Arbitrarily sized protection regions alignable anywhere in memory.
  - Region sizes can vary from a minimum of 1 byte to a maximum of 4 GB.
- Read/write access control permissions are defined in the region descriptor.
- Global cache-inhibit attribute indicator.

- Hardware-assisted maintenance of the descriptor valid bit minimizes coherency issues.
- Priority given to granting permission over denying access for overlapping region descriptors.
- Detection of access errors if a memory reference does not hit in any memory region, or if the reference is illegal in all hit memory regions. If an access error occurs, the reference is terminated with an error response, and the SMPU inhibits the bus cycle being sent to the targeted slave device.
- Instruction storage (ISI) or data storage (DSI) interrupts generated by aborted core accesses. Buffered writes generate Machine check exception (refer to the documentation related to the core for details).
- Error registers (per bus master ID) capture the last faulting address, attributes, and other information.
- Global SMPU enable/disable control bit.

13.3 Memory map/register definition

The programming model is partitioned into three groups:

- Control registers
- Error capture registers
- Data structure containing the region descriptors

The programming model can be referenced only in supervisor mode using 32-bit accesses. Attempted references with a different access size, to an undefined (reserved) address, with a non-supported access type (a write to a read-only register, or a read of a write-only register), or in user mode generate an error termination.

NOTE

See the chip configuration details for any chip-specific register information for this module. For example, a specific instance of the module may support only eight or fewer region descriptors.

SMPU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control/Error Status Register 0 (SMPU_CESR0)	32	R/W	0000_8002h	13.3.1/365
4	Control/Error Status Register 1 (SMPU_CESR1)	32	R/W	See section	13.3.2/366
100	Error Address Register, Bus Master n (SMPU_EAR0)	32	R	0000_0000h	13.3.3/367

Table continues on the next page...

SMPU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
104	Error Detail Register, Bus Master n (SMPU_EDR0)	32	R	0000_0080h	13.3.4/367
108	Error Address Register, Bus Master n (SMPU_EAR1)	32	R	0000_0000h	13.3.3/367
10C	Error Detail Register, Bus Master n (SMPU_EDR1)	32	R	0000_0080h	13.3.4/367
110	Error Address Register, Bus Master n (SMPU_EAR2)	32	R	0000_0000h	13.3.3/367
114	Error Detail Register, Bus Master n (SMPU_EDR2)	32	R	0000_0080h	13.3.4/367
118	Error Address Register, Bus Master n (SMPU_EAR3)	32	R	0000_0000h	13.3.3/367
11C	Error Detail Register, Bus Master n (SMPU_EDR3)	32	R	0000_0080h	13.3.4/367
120	Error Address Register, Bus Master n (SMPU_EAR4)	32	R	0000_0000h	13.3.3/367
124	Error Detail Register, Bus Master n (SMPU_EDR4)	32	R	0000_0080h	13.3.4/367
128	Error Address Register, Bus Master n (SMPU_EAR5)	32	R	0000_0000h	13.3.3/367
12C	Error Detail Register, Bus Master n (SMPU_EDR5)	32	R	0000_0080h	13.3.4/367
130	Error Address Register, Bus Master n (SMPU_EAR6)	32	R	0000_0000h	13.3.3/367
134	Error Detail Register, Bus Master n (SMPU_EDR6)	32	R	0000_0080h	13.3.4/367
138	Error Address Register, Bus Master n (SMPU_EAR7)	32	R	0000_0000h	13.3.3/367
13C	Error Detail Register, Bus Master n (SMPU_EDR7)	32	R	0000_0080h	13.3.4/367
140	Error Address Register, Bus Master n (SMPU_EAR8)	32	R	0000_0000h	13.3.3/367
144	Error Detail Register, Bus Master n (SMPU_EDR8)	32	R	0000_0080h	13.3.4/367
148	Error Address Register, Bus Master n (SMPU_EAR9)	32	R	0000_0000h	13.3.3/367
14C	Error Detail Register, Bus Master n (SMPU_EDR9)	32	R	0000_0080h	13.3.4/367
150	Error Address Register, Bus Master n (SMPU_EAR10)	32	R	0000_0000h	13.3.3/367
154	Error Detail Register, Bus Master n (SMPU_EDR10)	32	R	0000_0080h	13.3.4/367
158	Error Address Register, Bus Master n (SMPU_EAR11)	32	R	0000_0000h	13.3.3/367
15C	Error Detail Register, Bus Master n (SMPU_EDR11)	32	R	0000_0080h	13.3.4/367
160	Error Address Register, Bus Master n (SMPU_EAR12)	32	R	0000_0000h	13.3.3/367
164	Error Detail Register, Bus Master n (SMPU_EDR12)	32	R	0000_0080h	13.3.4/367
168	Error Address Register, Bus Master n (SMPU_EAR13)	32	R	0000_0000h	13.3.3/367
16C	Error Detail Register, Bus Master n (SMPU_EDR13)	32	R	0000_0080h	13.3.4/367
170	Error Address Register, Bus Master n (SMPU_EAR14)	32	R	0000_0000h	13.3.3/367
174	Error Detail Register, Bus Master n (SMPU_EDR14)	32	R	0000_0080h	13.3.4/367
178	Error Address Register, Bus Master n (SMPU_EAR15)	32	R	0000_0000h	13.3.3/367
17C	Error Detail Register, Bus Master n (SMPU_EDR15)	32	R	0000_0080h	13.3.4/367
400	Region Descriptor n, Word 0 (SMPU_RGD0_WORD0)	32	R/W	0000_0000h	13.3.5/368
404	Region Descriptor n, Word 1 (SMPU_RGD0_WORD1)	32	R/W	0000_0000h	13.3.6/369
408	Region Descriptor n, Word 2 (SMPU_RGD0_WORD2)	32	R/W	0000_0000h	13.3.7/369
40C	Region Descriptor n, Word 3 (SMPU_RGD0_WORD3)	32	R/W	0000_0000h	13.3.8/371
410	Region Descriptor n, Word 0 (SMPU_RGD1_WORD0)	32	R/W	0000_0000h	13.3.5/368
414	Region Descriptor n, Word 1 (SMPU_RGD1_WORD1)	32	R/W	0000_0000h	13.3.6/369
418	Region Descriptor n, Word 2 (SMPU_RGD1_WORD2)	32	R/W	0000_0000h	13.3.7/369
41C	Region Descriptor n, Word 3 (SMPU_RGD1_WORD3)	32	R/W	0000_0000h	13.3.8/371

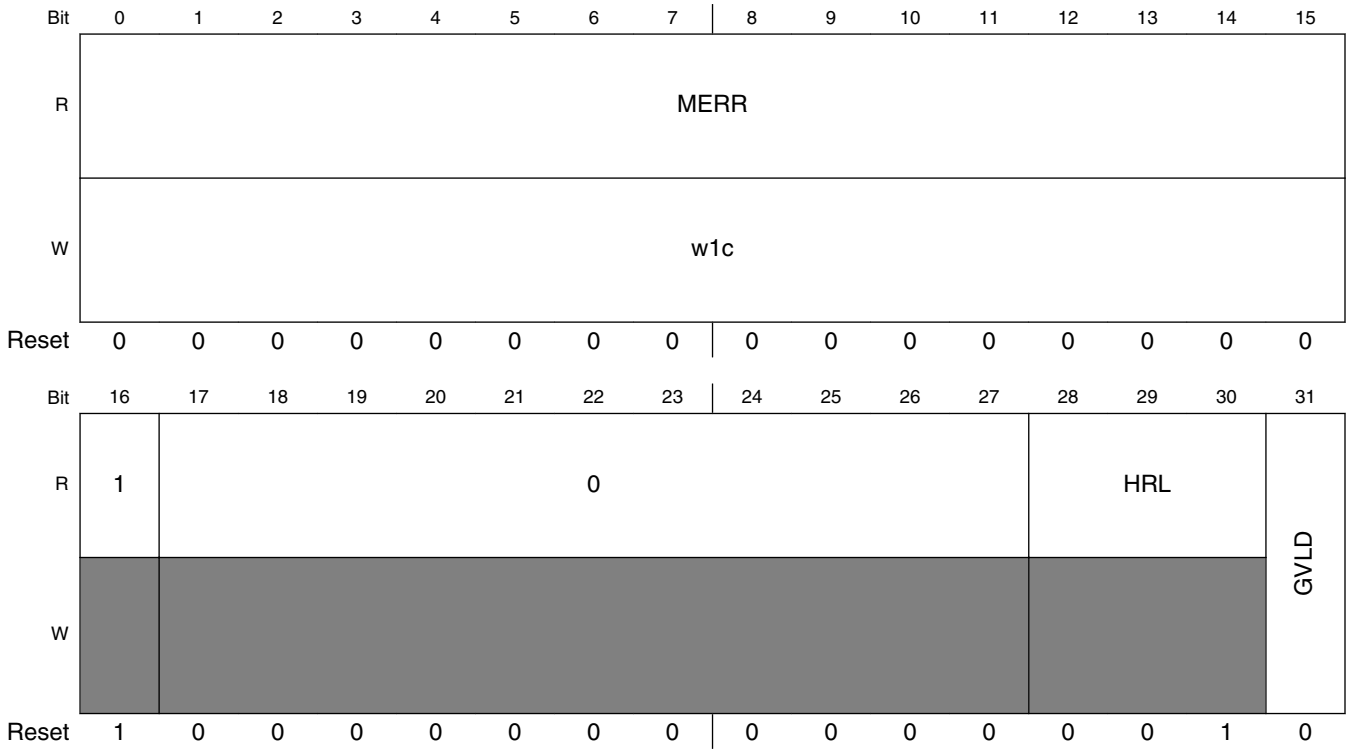
Table continues on the next page...

SMPU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
420	Region Descriptor n, Word 0 (SMPU_RGD2_WORD0)	32	R/W	0000_0000h	13.3.5/368
424	Region Descriptor n, Word 1 (SMPU_RGD2_WORD1)	32	R/W	0000_0000h	13.3.6/369
428	Region Descriptor n, Word 2 (SMPU_RGD2_WORD2)	32	R/W	0000_0000h	13.3.7/369
42C	Region Descriptor n, Word 3 (SMPU_RGD2_WORD3)	32	R/W	0000_0000h	13.3.8/371
430	Region Descriptor n, Word 0 (SMPU_RGD3_WORD0)	32	R/W	0000_0000h	13.3.5/368
434	Region Descriptor n, Word 1 (SMPU_RGD3_WORD1)	32	R/W	0000_0000h	13.3.6/369
438	Region Descriptor n, Word 2 (SMPU_RGD3_WORD2)	32	R/W	0000_0000h	13.3.7/369
43C	Region Descriptor n, Word 3 (SMPU_RGD3_WORD3)	32	R/W	0000_0000h	13.3.8/371
440	Region Descriptor n, Word 0 (SMPU_RGD4_WORD0)	32	R/W	0000_0000h	13.3.5/368
444	Region Descriptor n, Word 1 (SMPU_RGD4_WORD1)	32	R/W	0000_0000h	13.3.6/369
448	Region Descriptor n, Word 2 (SMPU_RGD4_WORD2)	32	R/W	0000_0000h	13.3.7/369
44C	Region Descriptor n, Word 3 (SMPU_RGD4_WORD3)	32	R/W	0000_0000h	13.3.8/371
450	Region Descriptor n, Word 0 (SMPU_RGD5_WORD0)	32	R/W	0000_0000h	13.3.5/368
454	Region Descriptor n, Word 1 (SMPU_RGD5_WORD1)	32	R/W	0000_0000h	13.3.6/369
458	Region Descriptor n, Word 2 (SMPU_RGD5_WORD2)	32	R/W	0000_0000h	13.3.7/369
45C	Region Descriptor n, Word 3 (SMPU_RGD5_WORD3)	32	R/W	0000_0000h	13.3.8/371
460	Region Descriptor n, Word 0 (SMPU_RGD6_WORD0)	32	R/W	0000_0000h	13.3.5/368
464	Region Descriptor n, Word 1 (SMPU_RGD6_WORD1)	32	R/W	0000_0000h	13.3.6/369
468	Region Descriptor n, Word 2 (SMPU_RGD6_WORD2)	32	R/W	0000_0000h	13.3.7/369
46C	Region Descriptor n, Word 3 (SMPU_RGD6_WORD3)	32	R/W	0000_0000h	13.3.8/371
470	Region Descriptor n, Word 0 (SMPU_RGD7_WORD0)	32	R/W	0000_0000h	13.3.5/368
474	Region Descriptor n, Word 1 (SMPU_RGD7_WORD1)	32	R/W	0000_0000h	13.3.6/369
478	Region Descriptor n, Word 2 (SMPU_RGD7_WORD2)	32	R/W	0000_0000h	13.3.7/369
47C	Region Descriptor n, Word 3 (SMPU_RGD7_WORD3)	32	R/W	0000_0000h	13.3.8/371
480	Region Descriptor n, Word 0 (SMPU_RGD8_WORD0)	32	R/W	0000_0000h	13.3.5/368
484	Region Descriptor n, Word 1 (SMPU_RGD8_WORD1)	32	R/W	0000_0000h	13.3.6/369
488	Region Descriptor n, Word 2 (SMPU_RGD8_WORD2)	32	R/W	0000_0000h	13.3.7/369
48C	Region Descriptor n, Word 3 (SMPU_RGD8_WORD3)	32	R/W	0000_0000h	13.3.8/371
490	Region Descriptor n, Word 0 (SMPU_RGD9_WORD0)	32	R/W	0000_0000h	13.3.5/368
494	Region Descriptor n, Word 1 (SMPU_RGD9_WORD1)	32	R/W	0000_0000h	13.3.6/369
498	Region Descriptor n, Word 2 (SMPU_RGD9_WORD2)	32	R/W	0000_0000h	13.3.7/369
49C	Region Descriptor n, Word 3 (SMPU_RGD9_WORD3)	32	R/W	0000_0000h	13.3.8/371
4A0	Region Descriptor n, Word 0 (SMPU_RGD10_WORD0)	32	R/W	0000_0000h	13.3.5/368
4A4	Region Descriptor n, Word 1 (SMPU_RGD10_WORD1)	32	R/W	0000_0000h	13.3.6/369
4A8	Region Descriptor n, Word 2 (SMPU_RGD10_WORD2)	32	R/W	0000_0000h	13.3.7/369
4AC	Region Descriptor n, Word 3 (SMPU_RGD10_WORD3)	32	R/W	0000_0000h	13.3.8/371
4B0	Region Descriptor n, Word 0 (SMPU_RGD11_WORD0)	32	R/W	0000_0000h	13.3.5/368
4B4	Region Descriptor n, Word 1 (SMPU_RGD11_WORD1)	32	R/W	0000_0000h	13.3.6/369
4B8	Region Descriptor n, Word 2 (SMPU_RGD11_WORD2)	32	R/W	0000_0000h	13.3.7/369
4BC	Region Descriptor n, Word 3 (SMPU_RGD11_WORD3)	32	R/W	0000_0000h	13.3.8/371

13.3.1 Control/Error Status Register 0 (SMPU_CESR0)

Address: 0h base + 0h offset = 0h

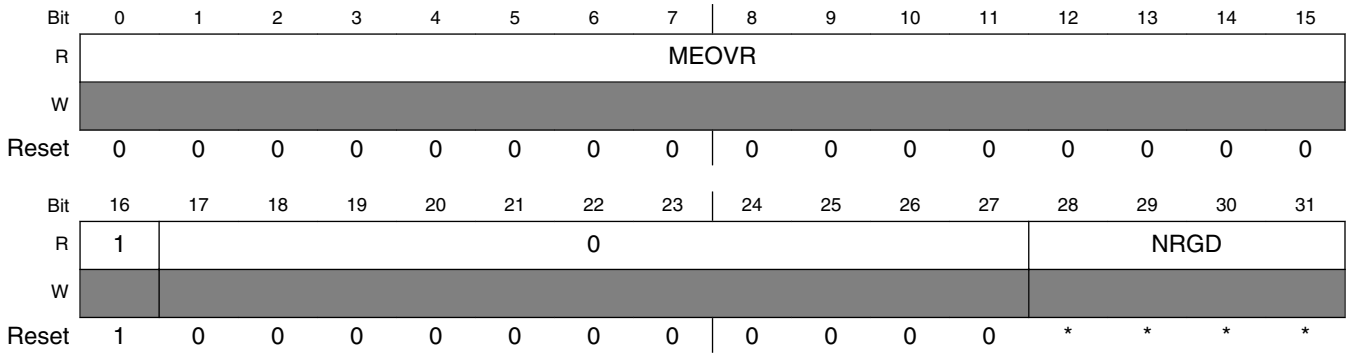


SMPU_CESR0 field descriptions

Field	Description
0–15 MERR	Master n error, where the bus master number matches the bit number Indicates a captured error in EARn and EDRn. A bit is set when the hardware detects an error and records the faulting address and attributes. It is cleared by writing one to it. If another error is captured at the exact same cycle as the write, the flag remains set. A find-first-one instruction (or equivalent) can detect the presence of a captured error. 0 No error has occurred for bus master n. 1 An error has occurred for bus master n.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
17–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–30 HRL	Hardware revision level Specifies the SMPU's hardware and definition revision level. It can be read by software to determine the functional definition of the module.
31 GVLD	Global Valid (global enable/disable for the SMPU) 0 SMPU is disabled. All accesses from all bus masters are allowed. 1 SMPU is enabled.

13.3.2 Control/Error Status Register 1 (SMPU_CESR1)

Address: 0h base + 4h offset = 4h



* Notes:
- NRGD field: See the chip-specific SMPU information for the reset value of this field.

SMPU_CESR1 field descriptions

Field	Description
0–15 MEOVR	Master n error overrun, where the bus master number matches the bit number Each bit in this field signals that another SMPU error for bus master n has occurred before the previous error was processed. The details of the first error are recorded in the EARn and EDRn registers and no information on subsequent errors is recorded until the associated CESR0[MERR] flag is cleared. 0 No error overrun condition has been detected for bus master n. 1 An error overrun condition has been detected for bus master n.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
17–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 NRGD	Number of region descriptors Indicates the number of region descriptors implemented in the SMPU. The number of region descriptors is (4 x NRGD). 0001 4 region descriptors 0010 8 region descriptors 0011 12 region descriptors 0100 16 region descriptors 0101 20 region descriptors 0110 24 region descriptors

13.3.3 Error Address Register, Bus Master n (SMPU_EAR*n*)

When the SMPU detects an access error on bus master n, the 32-bit reference address is captured in this read-only register and the corresponding bit in CESR0[MERR] set. Additional information about the faulting access is captured in the corresponding EDRn at the same time.[1]

Address: 0h base + 100h offset + (8d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SMPU_EAR*n* field descriptions

Field	Description
0–31 EADDR	Error address Indicates the reference address from bus master n that generated the access error.

1. The corresponding EARn and EDRn registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the CESR1[MEOVR] field until the CESR0[MERR] flag associated with the original error is cleared (by writing a 1).

13.3.4 Error Detail Register, Bus Master n (SMPU_EDR*n*)

When the SMPU detects an access error associated with bus master n, 32 bits of error detail are captured in this read-only register and the corresponding bit in CESR0[MERR] is set. Information on the faulting address is captured in the corresponding EARn register at the same time. [1]

Address: 0h base + 104h offset + (8d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EACD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EACD								1	EATTR		ERW	EMN			
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

SMPU_EDR*n* field descriptions

Field	Description
0–23 EACD	Error access control detail Indicates the region descriptor with the access error, where the region descriptor number matches the bit number. If EDRn contains a captured error and EACD is all zeroes, the access did not hit in any region descriptor. If only a single EACD bit is set, the access error was caused by a single non-overlapping region descriptor. If two or more EACD bits are set, the access error was caused by an overlapping set of region descriptors.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
25–26 EATTR	Error attributes Indicates attribute information about the faulting reference. 00 User mode, instruction access 01 User mode, data access 10 Supervisor mode, instruction access 11 Supervisor mode, data access
27 ERW	Error read/write Indicates the access type of the faulting reference. 0 Read 1 Write
28–31 EMN	Error master number Indicates the logical bus master number of the faulting reference.

1. The corresponding EARn and EDRn registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the CESR1[MEOVR] field until the CESR0[MERR] flag associated with the original error is cleared (by writing a 1).

13.3.5 Region Descriptor n, Word 0 (SMPU_RGD*n*_WORD0)

The first word of the region descriptor defines the byte start address of the memory region. Writes to this register clear the region descriptor's valid bit (RGDn_WORD3[VLD]).

Address: 0h base + 400h offset + (16d × i), where i=0d to 11d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SRTADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SMPU_RGD*n*_WORD0 field descriptions

Field	Description
0–31 SRTADDR	Start address Defines the byte start address of the memory region.

13.3.6 Region Descriptor n, Word 1 (SMPU_RGD*n*_WORD1)

The second word of the region descriptor defines the end address of the memory region. (RGDn_WORD3[VLD]).

Address: 0h base + 404h offset + (16d × i), where i=0d to 11d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ENDADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SMPU_RGD*n*_WORD1 field descriptions

Field	Description
0–31 ENDADDR	End address Defines the byte end address of the memory region. **NOTE:** The SMPU does not verify that ENDADDR ≥ SRTADDR.

13.3.7 Region Descriptor n, Word 2 (SMPU_RGD*n*_WORD2)

RGD_WORD2 defines the access control rights of the memory region. The access control rights are defined by separate read and write permissions. For these fields, the bus master number refers to the *logical* bus master number.

For the access control rights, there are two flags:

- Read (r) permission refers to the ability to access the referenced memory address using an operand (data) fetch or an instruction fetch.
- Write (w) permission refers to the ability to update the referenced memory address using a store (data) operation.

Each field consists of the two flags, with (r) being in the more significant position. For example, M0P has (r) as bit 0 and (w) as bit 1.

The bit settings are as follows:

- If set, the corresponding flag allows the specific access type (r = memory read, w = memory write).
- If cleared, the specific access type is not allowed.

Writes to this word clear the region descriptor's valid bit.

Address: 0h base + 408h offset + (16d × i), where i=0d to 11d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	M0P		M1P		M2P		M3P		M4P		M5P		M6P		M7P	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	M8P		M9P		M10P		M11P		M12P		M13P		M14P		M15P	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SMPU_RGD*n*_WORD2 field descriptions

Field	Description
0–1 M0P	Bus master 0 permissions
2–3 M1P	Bus master 1 permissions
4–5 M2P	Bus master 2 permissions
6–7 M3P	Bus master 3 permissions
8–9 M4P	Bus master 4 permissions
10–11 M5P	Bus master 5 permissions
12–13 M6P	Bus master 6 permissions
14–15 M7P	Bus master 7 permissions
16–17 M8P	Bus master 8 permissions
18–19 M9P	Bus master 9 permissions
20–21 M10P	Bus master 10 permissions
22–23 M11P	Bus master 11 permissions
24–25 M12P	Bus master 12 permissions
26–27 M13P	Bus master 13 permissions
28–29 M14P	Bus master 14 permissions

Table continues on the next page...

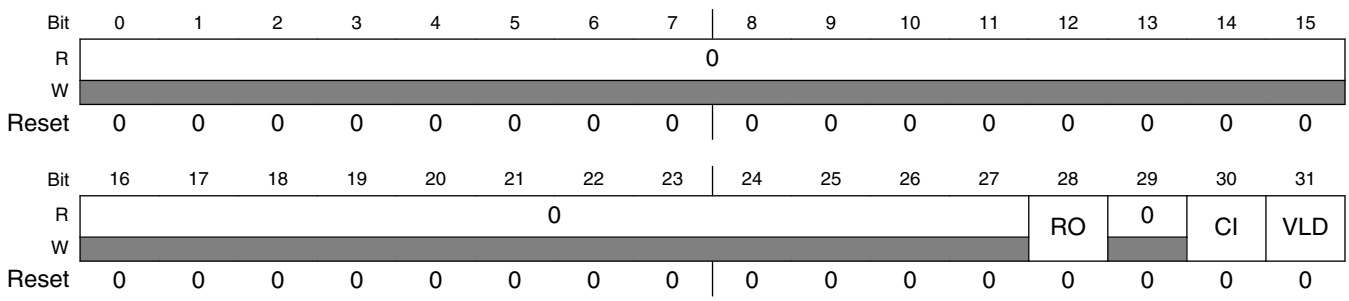
SMPU_RGD*n*_WORD2 field descriptions (continued)

Field	Description
30–31 M15P	Bus master 15 permissions

13.3.8 Region Descriptor n, Word 3 (SMPU_RGD*n*_WORD3)

The final word of the SMPU region descriptor contains the valid bit, plus other miscellaneous flags.

Address: 0h base + 40Ch offset + (16d × i), where i=0d to 11d



SMPU_RGD*n*_WORD3 field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 RO	Read-Only This bit is intended to prevent accidental writes of an SMPU region descriptor. **NOTE:** The valid bit of the RGD and the global valid bit have no effect on this bit functionality (once set, even if the RGD is not valid or SMPU is not global enabled, the RO functionality is present). 0 The region descriptor can be read or written. 1 Attempted writes to any location in the region descriptor are ignored with an error-free data transfer termination.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 CI	Cache Inhibit Defines the cacheability attribute of the memory region. This bit is returned to the bus master that initiated the memory reference. **NOTE:** An address range specified in an SMPU region descriptor for a cacheable space (that is, CI = 0) must be defined with a starting address aligned on a 0-modulo-32 byte address and with a multiple of the 32 byte cache line size factoring into the end address.

Table continues on the next page...

SMPU_RGD*n*_WORD3 field descriptions (continued)

Field	Description
	0 References to this region can be cached. 1 References to this region cannot be cached.
31 VLD	Valid Signals the region descriptor is valid. Any write to RGDn_WORD0–2 clears this bit. 0 Region descriptor is invalid. 1 Region descriptor is valid.

13.4 Functional description

In this section, the functional operation of the SMPU is detailed, including the operation of the access evaluation macro and the handling of error-terminated bus cycles.

13.4.1 Access evaluation macro

The basic operation of the SMPU is performed in the access evaluation macro, a hardware structure replicated in the connection matrix. The following block diagram represents an individual macro.

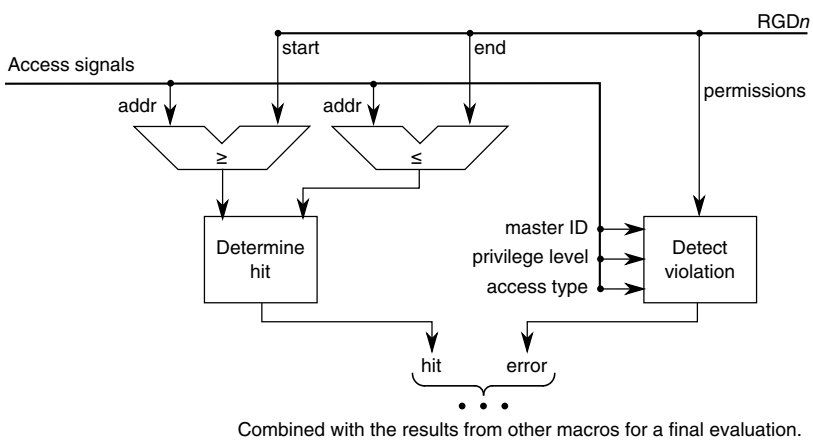


Figure 13-91. SMPU access evaluation macro

Each macro uses the information contained in the access signals (the address, the access type, and the master's ID and privilege level) and the contents of its corresponding region descriptor (RGD*n*) to perform two major functions: Hit determination and Violation determination. The individual hit and error results from each macro are then combined for a final evaluation; see Final evaluation and error terminations.

13.4.1.1 Hit determination

To determine if the current reference hits in the given region, two magnitude comparators are used with the region's start and end addresses. When RGD*n*_Word3[VLD] is 1, a region hit occurs when the requested address is between the start address (RGD*n*_Word0[SRTADDR]) and the end address (RGD*n*_Word1[ENDADDR]), inclusive.

NOTE

If ENDADDR < SRTADDR, no hit occurs.

NOTE

Region hit determination is based only on an address comparison of the first byte being accessed; that is, the SMPU does not check the size of the access to make sure it fits within an allowed region.

13.4.1.2 Violation determination

While the access evaluation macro is determining region hit, the logic is also evaluating if the current access is allowed by the permissions defined in the region descriptor. Using the master ID, a set of effective permissions is generated from the appropriate fields in the region descriptor. The protection violation logic then evaluates the access against the effective permissions using the specification shown below.

Table 13-91. Protection violation definition

Access type	Access permissions		Protection violation?
	r	w	
Instruction fetch read	0	—	Yes, no execute permission
	1	—	No, access is allowed
Data read	0	—	Yes, no read permission
	1	—	No, access is allowed
Data write	—	0	Yes, no write permission
	—	1	No, access is allowed

13.4.2 Final evaluation and error terminations

For each slave port monitored, the SMPU performs a reduction-AND of all the individual terms from each access evaluation macro. This expression then terminates the bus cycle with an error and reports an access error for three conditions:

1. The access does not hit in any region descriptor.
2. The access hits in a single region descriptor and that region has a protection violation.
3. The access hits in multiple (overlapping) regions and all regions have protection violations.

As shown in the third condition, granting permission is a higher priority than denying access for overlapping regions. This approach is more flexible to system software in region descriptor assignments. For an example of the use of overlapping region descriptors, see Application information.

NOTE

Similarly, the cache inhibit function gives a higher priority to the less restrictive result of overlapping regions. If two regions are hit and one has cache inhibit set, and the other does not, the result will not include a cache inhibit. It should also be noted that only the hit logic is taken into account for cache inhibit. For example, two region hits, one has protection violation and no cache inhibit, one has no violation and cache inhibit set. The result will be a granted access with no cache inhibit.

13.5 Initialization information

At system startup, load the appropriate number of region descriptors, including setting RGD*n*_Word3[VLD]. Setting CESR0[GVLD] enables the module.

If the system requires that all the loaded region descriptors be enabled simultaneously, first ensure that the entire SMPU is disabled (CESR0[GVLD]=0). Next, load the appropriate region descriptors (RGDn), including the setting of the RGDn_Word3[VLD] bits. Finally, set CESR0[GVLD] to enable the entire module.

Note

A region descriptor must be set to allow access to the SMPU registers if further changes are needed.

13.6 Application information

In an operational system, interfacing with the SMPU is generally classified into the following activities:

- Creating a new memory region

  Load the appropriate region descriptor into an available RGD*n*, using four sequential 32-bit writes. The hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes.

- Modifying a region descriptor

  Load the updates into the region descriptor using sequential 32-bit writes. Writing to Word 3 re-enables the region descriptor valid bit. The hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes.

- Removing a region descriptor

  Clearing RGD*n*_Word3[VLD] deletes an existing region descriptor.

- Accessing the SMPU

  Allocate a region descriptor to restrict SMPU access to supervisor mode from specific masters as appropriate.

- Detecting an access error

  The current bus cycle is terminated with an error response and EAR*n* and EDR*n* capture information on the faulting reference. The error-terminated bus cycle typically initiates an error response in the originating bus master. For example, a processor core may respond with a bus error exception, while a data movement bus master may respond with an error interrupt. The processor can retrieve the captured error address and detail information simply by reading E{A,D}R*n*. CESR0[MERR] signals which error registers contain captured fault data. Once the error information is retrieved, write a 1 to the appropriate CESR0[MERR] bit to clear it, thereby rearming the error capture logic associated with the EAR*n* and EDR*n* registers.

- Overlapping region descriptors

Applying overlapping regions often reduces the number of descriptors required for a given set of access controls. In the overlapping memory space, the protection rights of the corresponding region descriptors are logically summed together (the boolean OR operator).

Chapter 14
Intelligent AHB Gasket (IAHBG)

14.1 IAHBG chip-specific module information

14.1.1 IAHBG implemented operation modes

Table 14-1 shows the operating modes implemented in the IAHBG on this chip.

Table 14-1. IAHBG implemented operating modes

Operating mode (Master:Slave)	Comments
1:1	Master and slave at same operating frequency; gasket performance optimizations on by default
2:1	Master at twice frequency of slave; gasket performance optimizations on by default
1:2	Slave at twice frequency of master; gasket performance optimizations on by default

14.1.2 IAHBG clock selection ratios

The IAHBG is used as a bus gasket to connect busses running at different bus clock speeds. Internal signals generated by the MC_CGM module are connected to the IAHBG module that determine the clock ratio selection. User software is not used to determine the clock ratio setting for the IAHBG.

14.1.3 IAHBG features enabled by Platform Configuration Module (PCM)

The Platform Configuration Module (PCM) is used to enable/disable features of the IAHBG module. In general, the default setting in the PCM enables IAHBG features for the highest performance case. Unless otherwise noted, you generally do not have to change the PCM default settings that control IAHBG features.

14.2 Introduction

This section details the intelligent bridging gasket between the fast and slow clock domains. The intelligent operation of the gasket initiates pending reads early and optimizes bursts to recover some of the performance that is lost on a registered interface between different clock domains. Pending reads begin even if there is not an hready to the master. Non-bursted writes only begin when the slave is IDLE, pending writes wait until IDLE.

The IAHBG complies with the AHBV6lite protocol, including an extension to support masters that do address retraction during the error termination sequence or abort a burst read due to an error termination or another non-error termination like cache inhibit termination.

Table 14-2 lists the available operation modes.

Table 14-2. Operation modes available

Operating mode (Master:Slave)	Comments
0:0	Flowthru (no register wall)
1:1	Master and slave at same operating frequency; gasket performance optimizations on by default
2:1	Master at twice frequency of slave; gasket performance optimizations on by default
1:2	Slave at twice frequency of master; gasket performance optimizations on by default
4:1	Master at four times frequency of slave; gasket performance optimizations disabled at this time
1:4	Slave at four times frequency of master; gasket performance optimizations disabled at this time

14.3 Timing modes

14.3.1 1:1

In the 1:1 timing mode the master is running at the same clock frequency as the slave. The pending read can either be a single or burst transaction.

14.3.2 2:1

In the 2:1 timing mode the master is running at twice the clock frequency of the slave. The 4:1 mode is not shown because it is very similar to the 2:1 mode. The pending read can either be a single or burst transaction.

14.3.3 1:2

In the 1:2 timing mode, the slave is running at twice the clock frequency of the master. The 1:4 mode is not shown because it is very similar to the 1:2 mode. The pending read can either be a single or burst transaction.

Because the slave is operating at a faster clock frequency than the master, the transactions from the master must be stored into a queue of data and response attribute registers. At a certain point during the master burst transactions, the queue is either emptied into the master for a burst read or it is emptied into the slave for a burst write..

Chapter 15
Platform Configuration Module (PCM)

15.1 PCM memory map

The Platform Configuration Module contains three miscellaneous configuration registers for the device. Currently, the configuration registers are related to the operation of the FEC and intelligent bus bridging gasket. The module is mapped to PBRIDGE on-platform slot 10 with a base address of 0xFC02_8000.

PCM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	FEC Burst Optimization Master Control Register (PCM_FBOMCR)	32	R/W	0000_0000h	15.1.1/382
4	Bus Bridge Configuration Register 1 (PCM_IAHB_BE1)	32	R/W	0007_0707h	15.1.2/383
8	Bus Bridge Configuration Register 2 (PCM_IAHB_BE2)	32	R/W	0007_0707h	15.1.3/385
C	IPS Clock Gating Module Enable 0 Register (PCM_IPS_CGM_EN0)	32	R/W	0111_1111h	15.1.4/386

15.1.1 FEC Burst Optimization Master Control Register (PCM_FBOMCR)

The FEC burst optimization master control register (FBOMCR) controls FEC burst optimization behavior on the system bus.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					ACCERR	WBEN	RBEN	FXSBE							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PCM_FBOMCR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 ACCERR	Accumulate error This bit determines whether an error response for the first half of the write burst is accumulated to the second half of the write burst or discarded. In order to complete the burst, the FEC interface to the system bus responds by indicating that the first half of the burst completed without error before it actually writes the data so that it can fetch the second half of the write data from the FIFO. When actually written onto the system bus, the first half of the write burst can have an error. Because this half initially responded without an error to the FIFO, the error is discarded or accumulated with the error response for the second half of the burst. 0 Any error to the first half of the write burst is discarded. 1 Any actual error response to the first half of the write burst is accumulated in the second half's response. In other words, an error response to the first half will be seen in the response to the second half, even if the second half does not contain an error.
22 WBEN	Global write burst enable to XBAR slave port designated by FXSBE*n* 0 Write bursting to all XBAR slave ports is disabled. 1 Write bursting is enabled to any XBAR slave port whose FXSBE*n* bit is asserted.
23 RBEN	Global read burst enable from XBAR slave port designated by FXSBE*n* 0 Read bursting from all XBAR slave ports is disabled. 1 Read bursting is enabled from any XBAR slave port whose FXSBE*n* bit is asserted.

Table continues on the next page...

PCM_FBOMCR field descriptions (continued)

Field	Description
24–31 FXSBE	FEC XBAR slave burst enable. FXSBEn enables bursting by the FEC interface to the XBAR slave port controlled by that respective FXSBEn bit. If FXSBEn is asserted, then that XBAR slave port enabled by the bit can accept the bursts allowed by RBEN and WBEN. Otherwise, the FEC interface will not burst to the XBAR slave port controlled by that respective FXSBEn bit. Read bursts from that XBAR slave port are enabled by RBEN. Write bursts to that XBAR slave port are enabled by WBEN. FXSBEn assignments to XBAR slave ports: FXSBE0 = Flash FXSBE1 = reserved FXSBE2 = RAM, TCMA, TCMB FXSBE3 = reserved FXSBE4 = reserved FXSBE5 = reserved FXSBE6 = reserved FXSBE7 = PBRIDGE_0 and PBRIDGE_1

15.1.2 Bus Bridge Configuration Register 1 (PCM_IAHB_BE1)

NOTE

This register can only be accessed in supervisor mode.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0													1		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					PRE_S0	BRE_S0	BWE_S0	0					PRE_M3	BRE_M3	BWE_M3
W																
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1

PCM_IAHB_BE1 field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PCM_IAHB_BE1 field descriptions (continued)

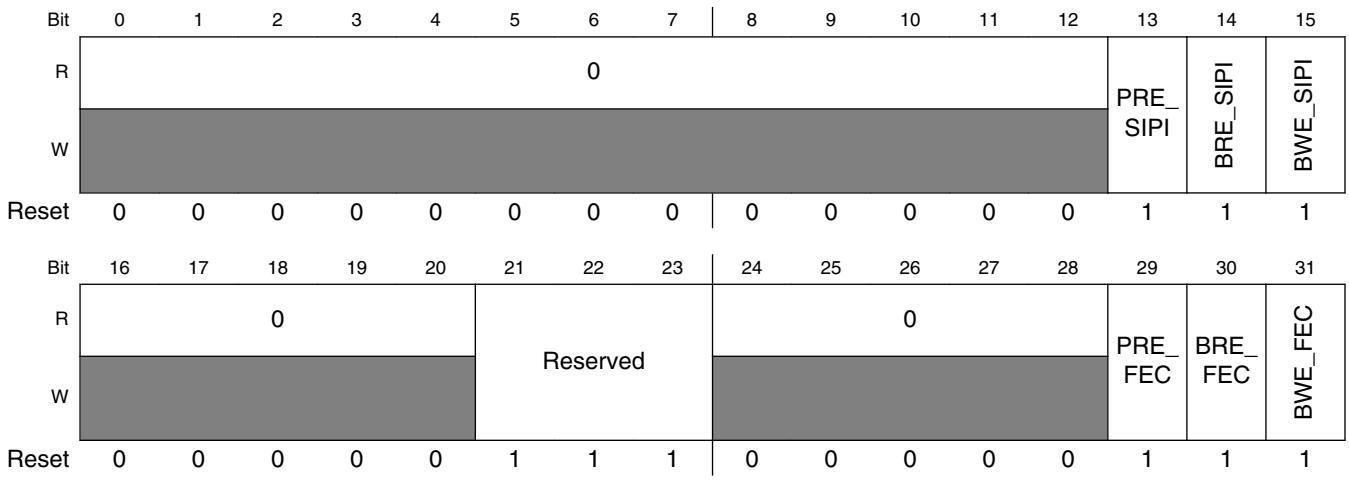
Field	Description
13–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 PRE_S0	Pending read enable (PRE) S0 This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled 1 Pending reads are enabled.
22 BRE_S0	Burst read enable (BRE) S0. This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance.
23 BWE_S0	Burst write enable (BWE) S0. This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 PRE_M3	Pending read enable (PRE) M3. This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
30 BRE_M3	Burst read enable (BRE) M3. This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
31 BWE_M3	Burst write enable (BWE) M3. This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.

15.1.3 Bus Bridge Configuration Register 2 (PCM_IAHB_BE2)

NOTE

This register can only be accessed in supervisor mode.

Address: 0h base + 8h offset = 8h



PCM_IAHB_BE2 field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 PRE_SIPI	Pending read enable (PRE) SIPI. This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
14 BRE_SIPI	Burst read enable (BRE) SIPI. This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
15 BWE_SIPI	Burst write enable (BWE) SIPI. This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

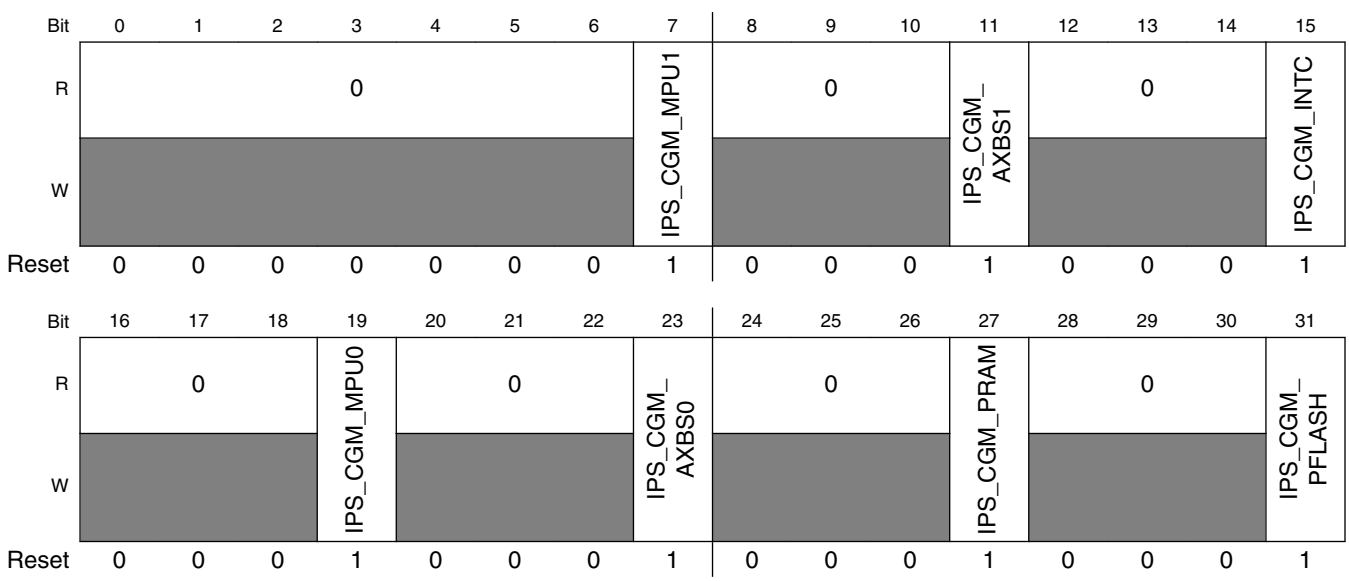
Table continues on the next page...

PCM_IAHB_BE2 field descriptions (continued)

Field	Description
21–23 Reserved	This field is reserved. Always write the reset value to this field.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 PRE_FEC	Pending read enable (PRE) FEC. This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending writes are enabled.
30 BRE_FEC	Burst read enable (BRE) FEC. This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
31 BWE_FEC	Burst write enable (BWE) FEC. This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat..

15.1.4 IPS Clock Gating Module Enable 0 Register (PCM_IPS_CGM_EN0)

Address: 0h base + Ch offset = Ch



PCM_IPS_CGM_EN0 field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 IPS_CGM_MPU1	IPS clock gating enable for MPU1 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 IPS_CGM_ AXBS1	IPS clock gating enable for AXBS1 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
12–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 IPS_CGM_INTC	IPS clock gating enable for INTC 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 IPS_CGM_MPU0	IPS clock gating enable for MPU0 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
20–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 IPS_CGM_ AXBS0	IPS clock gating enable for AXBS0 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 IPS_CGM_ PRAM	IPS clock gating enable for PRAM Controller 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.
28–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 IPS_CGM_ PFLASH	IPS clock gating enable for Flash Controller 0 Disable clock gating. 1 Enable clock gating, one IPS clock delay is introduced between master and slave.

Chapter 16
Semaphores2 (SEMA42)

16.1 Introduction

The SEMA42 is a memory-mapped module that provides robust hardware support needed in multi-core systems for implementing semaphores and provides a simple mechanism to achieve "lock and unlock" operations via a single write access. The hardware semaphore module provides hardware-enforced gates as well as other useful system functions related to the gating mechanisms.

16.1.1 Features

- Supports 16 hardware-enforced gates in a multi-processor configuration
  - Gates appear as an 16-entry byte-size array with read and write accesses.
    - Processors lock gates by writing "processor_number+1" to the appropriate gate and must read back the gate value to verify the lock operation was successful.
    - Once locked, the gate is unlocked by a write of zeroes from the locking processor.
    - The number of implemented gates is specified by a hardware configuration define.
  - Each hardware gate appears as a 16-state, 4-bit state machine.
    - 16-state implementation

      if gate = 0x0, then state = unlocked
      if gate = 0x1, then state = locked by processor (master) 0
      if gate = 0x2, then state = locked by processor (master) 1
      …
      if gate = 0xF, then state = locked by processor (master) 14

- Uses the logical bus master number as a reference attribute plus the specified data patterns to validate all write operations.
- Once locked, the gate can (and must) be unlocked by a write of zeroes from the locking processor.

- Secure reset mechanisms are supported to clear the contents of individual gates, as well as a clear_all capability.

A simplified block diagram of the Semaphores module is shown in the following figure. In the diagram, the register blocks named gate0, gate1,…, gate (n-2), gate (n-1) include the finite state machines implementing the semaphore gates.

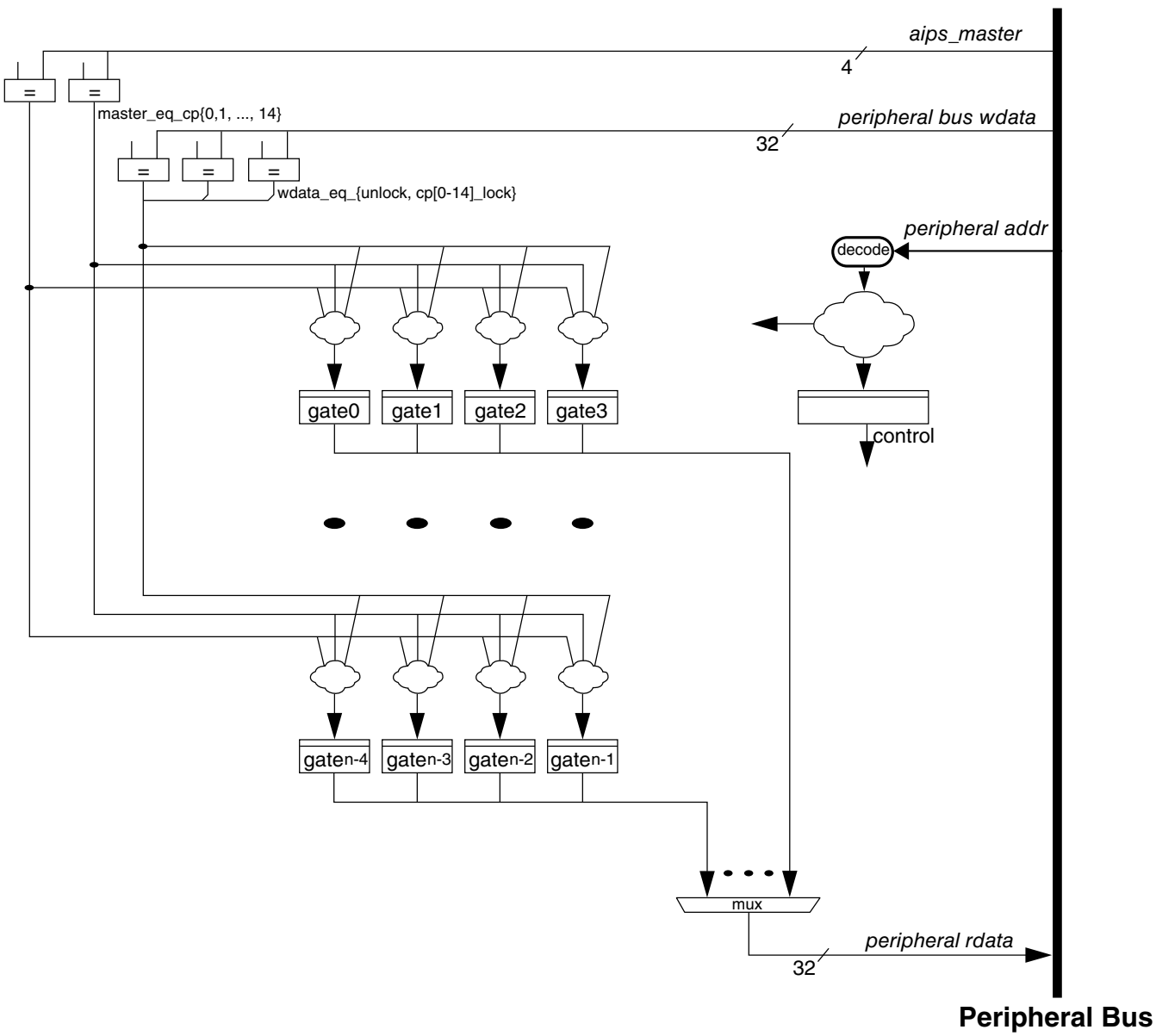


Figure 16-1. Semaphores2 block diagram

16.2 Memory map/register definition

Only Supervisor Mode accesses are allowed on these registers. User accesses will be terminated with an error.

SEMA42 memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Gate Register (SEMA42_GATE0)	8	R/W	00h	16.2.1/391
1	Gate Register (SEMA42_GATE1)	8	R/W	00h	16.2.1/391
2	Gate Register (SEMA42_GATE2)	8	R/W	00h	16.2.1/391
3	Gate Register (SEMA42_GATE3)	8	R/W	00h	16.2.1/391
4	Gate Register (SEMA42_GATE4)	8	R/W	00h	16.2.1/391
5	Gate Register (SEMA42_GATE5)	8	R/W	00h	16.2.1/391
6	Gate Register (SEMA42_GATE6)	8	R/W	00h	16.2.1/391
7	Gate Register (SEMA42_GATE7)	8	R/W	00h	16.2.1/391
8	Gate Register (SEMA42_GATE8)	8	R/W	00h	16.2.1/391
9	Gate Register (SEMA42_GATE9)	8	R/W	00h	16.2.1/391
A	Gate Register (SEMA42_GATE10)	8	R/W	00h	16.2.1/391
B	Gate Register (SEMA42_GATE11)	8	R/W	00h	16.2.1/391
C	Gate Register (SEMA42_GATE12)	8	R/W	00h	16.2.1/391
D	Gate Register (SEMA42_GATE13)	8	R/W	00h	16.2.1/391
E	Gate Register (SEMA42_GATE14)	8	R/W	00h	16.2.1/391
F	Gate Register (SEMA42_GATE15)	8	R/W	00h	16.2.1/391
40	Reset Gate Write (SEMA42_RSTGT_W)	16	W	See section	16.2.2/392
40	Reset Gate Read (SEMA42_RSTGT_R)	16	R	0000h	16.2.3/394

16.2.1 Gate Register (SEMA42_GATE*n*)

Each semaphore gate is implemented in a 4-bit finite state machine, right-justified in a byte data structure. The hardware uses the logical bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. Attempted writes with a data value that is neither the unlock value (0x00) nor the appropriate lock value (processor_number + 1) are simply

treated as "no operation" and do not affect any gate state. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes.

Address: 0h base + 0h offset + (1d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7
Read	0				GTFSM			
Write								
Reset	0	0	0	0	0	0	0	0

SEMA42_GATE*n* field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 GTFSM	Gate Finite State Machine. The state of the gate reflects the last processor that locked it, which can be useful during system debug. The hardware gate is maintained in a 16-state implementation, defined as: 0000 The gate is unlocked (free). 0001 The gate has been locked by processor 0. 0010 The gate has been locked by processor 1. 0011 The gate has been locked by processor 2. 0100 The gate has been locked by processor 3. 0101 The gate has been locked by processor 4. 0110 The gate has been locked by processor 5. 0111 The gate has been locked by processor 6. 1000 The gate has been locked by processor 7. 1001 The gate has been locked by processor 8. 1010 The gate has been locked by processor 9. 1011 The gate has been locked by processor 10. 1100 The gate has been locked by processor 11. 1101 The gate has been locked by processor 12. 1110 The gate has been locked by processor 13. 1111 The gate has been locked by processor 14.

16.2.2 Reset Gate Write (SEMA42_RSTGT_W)

NOTE

This section and the following section, Reset Gate Read (SEMA42_RSTGT_R), describe the same register. This section describes the general operation of the register and also shows how the register fields appear when the register is written. Reset Gate Read (SEMA42_RSTGT_R) shows how the register fields appear when the register is read.

Although the intent of the hardware gate implementation specifies a protocol where the locking domain must unlock the gate, it is recognized that system operation may require a reset function to re-initialize the state of any gate(s) without requiring a system-level reset.

To support this special gate reset requirement, the Semaphores module implements a "secure" reset mechanism that allows a hardware gate (or all the gates) to be initialized by following a specific dual-write access pattern. Using a technique similar to that required for the servicing of a software watchdog timer, the secure gate reset requires two consecutive writes with predefined data patterns from the same domain to force the clearing of the specified gate(s). The required access pattern is:

1. A domain performs a 16-bit write to the SEMA42_RSTGT memory location. The most significant byte (SEMA42_RSTGT[RSTGDP]) must be E2h; the least significant byte is a "don't_care" for this reference.
2. The same domain then performs a second 16-bit write to the SEMA42_RSTGT location. For this write, the upper byte (SEMA42_RSTGT[RSTGDP]) is the logical complement of the first data pattern (1Dh) and the lower byte (SEMA42_RSTGT[RSTGTN]) specifies the gate(s) to be reset. This gate field can specify a single gate be cleared, or that all gates are to be cleared. If the same domain writes incorrect data on the second access or another domain performs the second write access, the special gate reset sequence is aborted and no error signal is asserted.
3. Reads of the SEMA42_RSTGT location return information on the 2-bit state machine (SEMA42_RSTGT[RSTGSM]) that implements this function, the domain performing the reset (SEMA42_RSTGT[RSTGMS]), and the gate number(s) last cleared (SEMA42_RSTGT[RSTGTN]). Reads of the SEMA42_RSTGT register do not affect the secure reset finite state machine in any manner.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read																
Write	RSTGDP								RSTGTN							
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- Reset value is not applicable for writes.

SEMA42_RSTGT_W field descriptions

Field	Description
0–7 RSTGDP	Reset Gate Data Pattern. This write-only field is accessed with the specified data patterns on the two consecutive writes to enable the gate reset mechanism. For the first write, RSTGDP = E2h while the second write requires RSTGDP = 1Dh.

Table continues on the next page...

SEMA42_RSTGT_W field descriptions (continued)

Field	Description
8–15 RSTGTN	Reset Gate Number. This 8-bit field specifies the specific hardware gate to be reset. This field is updated by the second write. If RSTGTN < 64, then reset the single gate defined by RSTGTN, else reset all the gates.

16.2.3 Reset Gate Read (SEMA42_RSTGT_R)

This section describes how the Reset Gate register fields appear when the register is read. See Reset Gate Write (SEMA42_RSTGT_W) for the description of the register's overall operation and how the register fields appear when the register is written.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	ROZ		RSTGSM		RSTGMS				RSTGTN							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SEMA42_RSTGT_R field descriptions

Field	Description
0–1 ROZ	This field always returns the value 0 when read.
2–3 RSTGSM	Reset Gate Finite State Machine. Reads of the SEMA42_RSTGT register return the encoded state machine value. Note the RSTGSM = 10 state is valid for only a single machine cycle, so it is impossible for a read to return this value. The reset state machine is maintained in a 2-bit, 3-state implementation, defined as: 00 Idle, waiting for the first data pattern write. 01 Waiting for the second data pattern write. 10 The 2-write sequence has completed. Generate the specified gate reset(s). After the reset is performed, this machine returns to the idle (waiting for first data pattern write) state. The "01" state persists for only one clock cycle. Software cannot observe this state. 11 This state encoding is never used and therefore reserved.
4–7 RSTGMS	Reset Gate Bus Master. This 4-bit read-only field records the logical number of the bus master performing the gate reset function. The reset function requires that the two consecutive writes to this register must be initiated by the same bus master to succeed. This field is updated each time a write to this register occurs. The association between system bus master port numbers, the associated bus master device, and the logical processor number is SoC-specific. Consult the device reference manual for this information.
8–15 RSTGTN	Reset Gate Number. This 8-bit field specifies the specific hardware gate to be reset. This field is updated by the second write. If RSTGTN < 64, then reset the single gate defined by RSTGTN, else reset all the gates.

16.3 Functional description

Functional operation of the Semaphores module and specific details of the state machines of the SEMA42_GATEn registers are provided as follows.

As described previously, each of the SEMA42_GATEn registers implements a 4-bit, 16-state machine. A *simplified* diagram of the state transitions for each gate is shown in Figure 16-21.

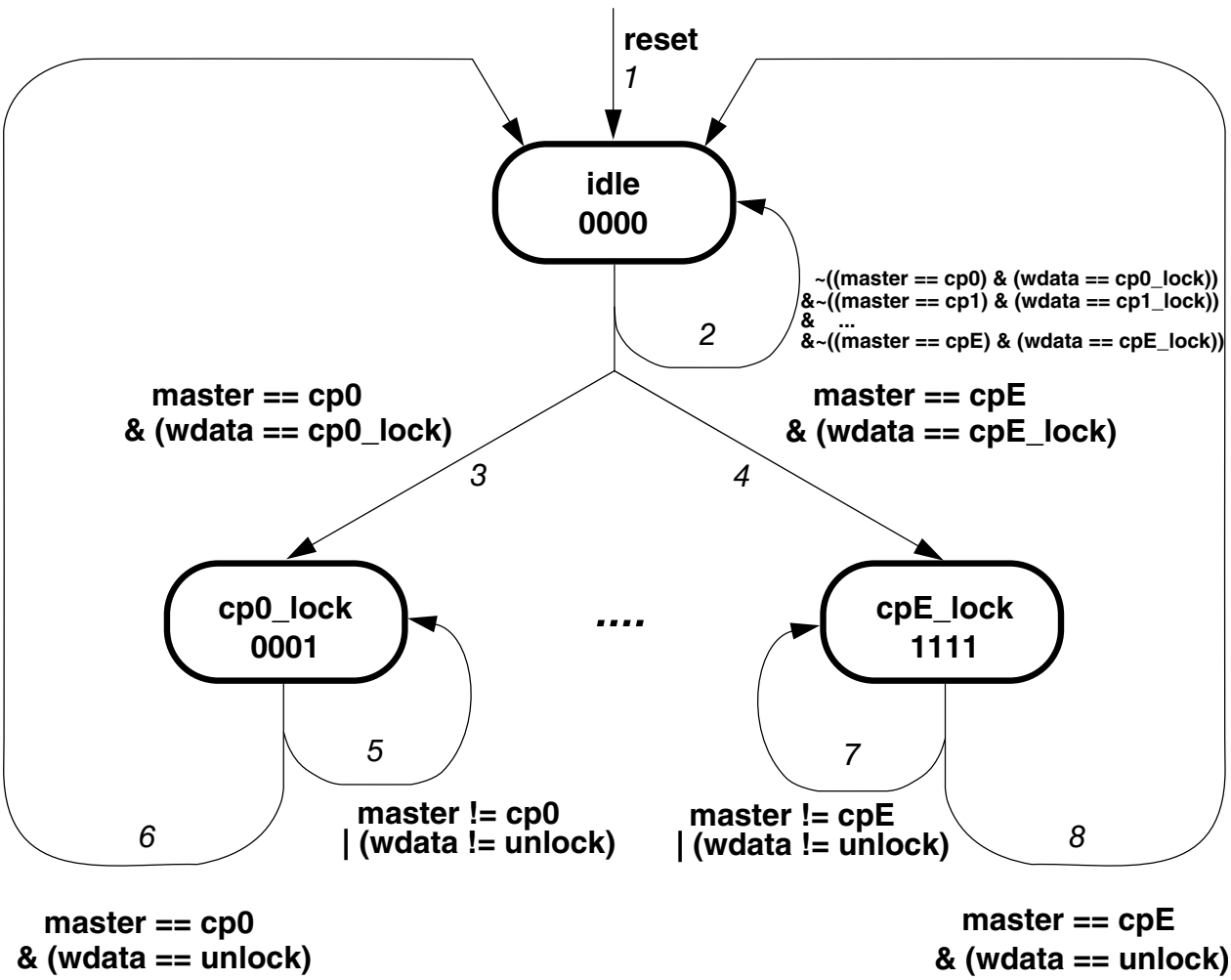


Figure 16-21. SEMA42_GATEn state machine

The bus master number is used to identify core processor *X* (cp*X*) where the notation cpE is used to represent core processor 14 (= 0xE). The platform passes the AHB bus master number through the peripheral bridge controller and drives a signal to the Semaphores module.

The state transitions for SEMA42_GATEn are defined in Table 16-21.

Table 16-21. SEMA42_GATEn state transitions

Current State	Next State	Transition	Description
–	idle	1	Any reset, whether a system reset or a software-initiated gate reset, unconditionally forces the gate into the idle state.
idle	idle	2	Unless a write of the appropriate lock value from the corresponding processor occurs, the gate remains in the idle state.
idle	cp0_lock	3	When a write of the "cp0_lock" data value is initiated by processor 0, the gate transitions into the cp0_lock state.
idle	cpE_lock	4	When a write of the "cpE_lock" value is initiated by processor 0xE, the gate transitions into the cpE_lock state.
cp0_lock	cp0_lock	5	Once in this state, the gate remains here if any attempted write is not from cp0 with the unlock data value.
cp0_lock	idle	6	The gate returns to the idle (unlocked) state once a write from cp0 with the unlock data value occurs.
cpE_lock	cpE_lock	7	Once in this state, the gate remains here if any attempted write is not from cpE with the unlock data value.
cpE_lock	idle	8	The gate returns to the idle (unlocked) state once a write from cpE with the unlock data value occurs.

The following gate data values are used:

- The lock data value is (processor number + 1) where the processor number and the platform bus master number are the same.
- The unlock data value is 0x00.

Chapter 17
Interrupt Controller (INTC)

17.1 Interrupt Controller (INTC) chip-specific module information

17.1.1 INTC configurable parameters

The INTC on this chip supports 2 processors.

17.1.2 Interrupt sources

The following table defines the interrupt sources for the INTC on this chip. All IRQ numbers not specifically listed are not used.

Table 17-1. Interrupt sources

IRQ #	Offset	Source description	Source name
0	1000	Software settable flag 0	INTC_SSCIR0[CLR0]
1	1004	Software settable flag 1	INTC_SSCIR0[CLR1]
2	1008	Software settable flag 2	INTC_SSCIR0[CLR2]
3	100C	Software settable flag 3	INTC_SSCIR0[CLR3]
4	1010	Software settable flag 4	INTC_SSCIR0[CLR4]
5	1014	Software settable flag 5	INTC_SSCIR0[CLR5]
6	1018	Software settable flag 6	INTC_SSCIR0[CLR6]
7	101C	Software settable flag 7	INTC_SSCIR0[CLR7]
8	1020	Software settable flag 8	INTC_SSCIR0[CLR8]
9	1024	Software settable flag 9	INTC_SSCIR0[CLR9]
10	1028	Software settable flag 10	INTC_SSCIR0[CLR10]
11	102C	Software settable flag 11	INTC_SSCIR0[CLR11]
12	1030	Software settable flag 12	INTC_SSCIR0[CLR12]
13	1034	Software settable flag 13	INTC_SSCIR0[CLR13]

Table continues on the next page...

Table 17-1. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
14	1038	Software settable flag 14	INTC_SSCIR0[CLR14]
15	103C	Software settable flag 15	INTC_SSCIR0[CLR15]
16	1040	Software settable flag 16	INTC_SSCIR0[CLR16]
17	1044	Software settable flag 17	INTC_SSCIR0[CLR17]
18	1048	Software settable flag 18	INTC_SSCIR0[CLR18]
19	104C	Software settable flag 19	INTC_SSCIR0[CLR19]
20	1050	Software settable flag 20	INTC_SSCIR0[CLR20]
21	1054	Software settable flag 21	INTC_SSCIR0[CLR21]
22	1058	Software settable flag 22	INTC_SSCIR0[CLR22]
23	105C	Software settable flag 23	INTC_SSCIR0[CLR23]
24	1060	Software settable flag 24	INTC_SSCIR0[CLR24]
25	1064	Software settable flag 25	INTC_SSCIR0[CLR25]
26	1068	Software settable flag 26	INTC_SSCIR0[CLR26]
27	106C	Software settable flag 27	INTC_SSCIR0[CLR27]
28	1070	Software settable flag 28	INTC_SSCIR0[CLR28]
29	1074	Software settable flag 29	INTC_SSCIR0[CLR29]
30	1078	Software settable flag 30	INTC_SSCIR0[CLR30]
31	107C	Software settable flag 31	INTC_SSCIR0[CLR31]
32	1080	Platform watchdog timer0	SWT_0_IR[TIF]
33	1084	Platform watchdog timer1	SWT_1_IR[TIF]
35	108C	Platform watchdog timer3	SWT_3_IR[TIF]
36	1090	Platform periodic timer 0_0 (STM)	STM_0_CIR0[CIF]
37	1094	Platform periodic timer 0_1 (STM)	STM_0_CIR1[CIF]
38	1098	Platform periodic timer 0_2 (STM)	STM_0_CIR2[CIF]
39	109A	Platform periodic timer 0_3 (STM)	STM_0_CIR3[CIF]
40	10A0	Platform periodic timer 1_0 (STM)	STM_1_CIR0[CIF]
41	10A4	Platform periodic timer 1_1 (STM)	STM_1_CIR1[CIF]
42	10A8	Platform periodic timer 1_2 (STM)	STM_1_CIR2[CIF]
43	10AC	Platform periodic timer 1_3 (STM)	STM_1_CIR3[CIF]
52	10D0	eDMA Combined Error 127 - 0	eDMA Channel Error Flags
53	10D4	eDMA Channel 0	DMA_INTL[INT0]
54	10D8	eDMA Channel 1	DMA_INTL[INT1]
55	10DC	eDMA Channel 2	DMA_INTL[INT2]
56	10E0	eDMA Channel 3	DMA_INTL[INT3]
57	10E4	eDMA Channel 4	DMA_INTL[INT4]
58	10E8	eDMA Channel 5	DMA_INTL[INT5]
59	10EC	eDMA Channel 6	DMA_INTL[INT6]
60	10F0	eDMA Channel 7	DMA_INTL[INT7]
61	10F4	eDMA Channel 8	DMA_INTL[INT8]

Table continues on the next page...

Table 17-1. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
62	10F8	eDMA Channel 9	DMA_INTL[INT9]
63	10FC	eDMA Channel 10	DMA_INTL[INT10]
64	1100	eDMA Channel 11	DMA_INTL[INT11]
65	1104	eDMA Channel 12	DMA_INTL[INT12]
66	1108	eDMA Channel 13	DMA_INTL[INT13]
67	110C	eDMA Channel 14	DMA_INTL[INT14]
68	1110	eDMA Channel 15	DMA_INTL[INT15]
69	1114	eDMA Channel 16	DMA_INTL[INT16]
70	1118	eDMA Channel 17	DMA_INTL[INT17]
71	111C	eDMA Channel 18	DMA_INTL[INT18]
72	1120	eDMA Channel 19	DMA_INTL[INT19]
73	1124	eDMA Channel 20	DMA_INTL[INT20]
74	1128	eDMA Channel 21	DMA_INTL[INT21]
75	112C	eDMA Channel 22	DMA_INTL[INT22]
76	1130	eDMA Channel 23	DMA_INTL[INT23]
77	1134	eDMA Channel 24	DMA_INTL[INT24]
78	1138	eDMA Channel 25	DMA_INTL[INT25]
79	113C	eDMA Channel 26	DMA_INTL[INT26]
80	1140	eDMA Channel 27	DMA_INTL[INT27]
81	1144	eDMA Channel 28	DMA_INTL[INT28]
82	1148	eDMA Channel 29	DMA_INTL[INT29]
83	114C	eDMA Channel 30	DMA_INTL[INT30]
84	1150	eDMA Channel 31	DMA_INTL[INT31]
85	1154	eDMA Channel 32	DMA_INTH[INT32]
86	1158	eDMA Channel 33	DMA_INTH[INT33]
87	115C	eDMA Channel 34	DMA_INTH[INT34]
88	1160	eDMA Channel 35	DMA_INTH[INT35]
89	1164	eDMA Channel 36	DMA_INTH[INT36]
90	1168	eDMA Channel 37	DMA_INTH[INT37]
91	116C	eDMA Channel 38	DMA_INTH[INT38]
92	1170	eDMA Channel 39	DMA_INTH[INT39]
93	1174	eDMA Channel 40	DMA_INTH[INT40]
94	1178	eDMA Channel 41	DMA_INTH[INT41]
95	117C	eDMA Channel 42	DMA_INTH[INT42]
96	1180	eDMA Channel 43	DMA_INTH[INT43]
97	1184	eDMA Channel 44	DMA_INTH[INT44]
98	1188	eDMA Channel 45	DMA_INTH[INT45]
99	118C	eDMA Channel 46	DMA_INTH[INT46]
100	1190	eDMA Channel 47	DMA_INTH[INT47]

Table continues on the next page...

Table 17-1. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
101	1194	eDMA Channel 48	DMA_INTH[INT48]
102	1198	eDMA Channel 49	DMA_INTH[INT49]
103	119C	eDMA Channel 50	DMA_INTH[INT50]
104	11A0	eDMA Channel 51	DMA_INTH[INT51]
105	11A4	eDMA Channel 52	DMA_INTH[INT52]
106	11A8	eDMA Channel 53	DMA_INTH[INT53]
107	11AC	eDMA Channel 54	DMA_INTH[INT54]
108	11B0	eDMA Channel 55	DMA_INTH[INT55]
109	11B4	eDMA Channel 56	DMA_INTH[INT56]
110	11B8	eDMA Channel 57	DMA_INTH[INT57]
111	11BC	eDMA Channel 58	DMA_INTH[INT58]
112	11C0	eDMA Channel 59	DMA_INTH[INT59]
113	11C4	eDMA Channel 60	DMA_INTH[INT60]
114	11C8	eDMA Channel 61	DMA_INTH[INT61]
115	11CC	eDMA Channel 62	DMA_INTH[INT62]
116	11D0	eDMA Channel 63	DMA_INTH[INT63]
185	12E4	Flash controller Prog/Erase/Suspend IRQ_0	MCR[DONE]
218	1368	Ethernet_0_0	EIR[TXF]
219	136C	Ethernet_0_1	EIR[RXF]
220	1370	Ethernet_0_2	EIR[HBERR] EIR[BABR] EIR[BABT] EIR[GRA] EIR[TXB] EIR[RXB] EIR[MII] EIR[EBERR] EIR[LC] EIR[RL] EIR[UN]
226	1388	Periodic Interrupt Timer (PIT0)	PIT_0_TFLG0[TIF]
227	138C	Periodic Interrupt Timer (PIT1)	PIT_0_TFLG1[TIF]
228	1390	Periodic Interrupt Timer (PIT2)	PIT_0_TFLG2[TIF]
229	1394	Periodic Interrupt Timer (PIT3)	PIT_0_TFLG3[TIF]
230	1398	Periodic Interrupt Timer (PIT4)	PIT_0_TFLG4[TIF]
231	139C	Periodic Interrupt Timer (PIT5)	PIT_0_TFLG5[TIF]
232	13A0	Periodic Interrupt Timer (PIT6)	PIT_0_TFLG6[TIF]
233	13A4	Periodic Interrupt Timer (PIT7)	PIT_0_TFLG7[TIF]
239	13BC	PIT_RTI	PIT_0_RTI_TFLG[TIF]
240	13C0	PIT_64_Lower	PIT_1_TFLG0[TIF]
241	13C4	PIT_64_Upper	PIT_1_TFLG1[TIF]
242	13C8	XOSC counter	XOSC
243	13CC	SIU External Interrupt_0	SIU External Interrupt_0
244	13D0	SIU External Interrupt_1	SIU External Interrupt_1
251	13EC	MC_ME 0	ME_IS[I_SAFE]
252	13F0	MC_ME 1	ME_IS[I_MTC]
253	13F4	MC_ME 2	ME_IS[I_IMODE]
254	13F8	MC_ME 3	ME_IS[I_ICONF]

Table continues on the next page...

Table 17-1. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
255	13FC	MC_RGM 0	MC_RGM Functional and destructive reset alternate event interrupt
259	140C	DSPI0_0	DSPI_0_SR[TFUF] | DSPI_0_SR[RFOF]| DSPI_0_SR[TFIWF]
260	1410	DSPI0_1	DSPI_0_SR[EOQF]
261	1414	DSPI0_2	DSPI_0_SR[TFFF]
262	1418	DSPI0_3	DSPI_0_SR[TCF]
263	141C	DSPI0_4	DSPI_0_SR[RFDF]
264	1420	DSPI0_5	DSPI_0_SR[CMD_TCF]
265	1424	DSPI0_6	DSPI_0_SR[CMDFFF]
266	1428	DSPI0_7	DSPI_0_SR[SPEF]
268	1430	DSPI1_0	DSPI_1_SR[TFUF] | DSPI_1_SR[RFOF]| DSPI_1_SR[TFIWF]
269	1434	DSPI1_1	DSPI_1_SR[EOQF]
270	1438	DSPI1_2	DSPI_1_SR[TFFF]
271	143C	DSPI1_3	DSPI_1_SR[TCF]
272	1440	DSPI1_4	DSPI_1_SR[RFDF]
273	1444	DSPI1_5	DSPI_1_SR[CMD_TCF]
274	1448	DSPI1_6	DSPI_1_SR[CMDFFF]
275	144C	DSPI1_7	DSPI_1_SR[SPEF]
277	1454	DSPI2_0	DSPI_2_SR[TFUF] | DSPI_2_SR[RFOF]| DSPI_2_SR[TFIWF]
278	1458	DSPI2_1	DSPI_2_SR[EOQF]
279	145C	DSPI2_2	DSPI_2_SR[TFFF]
280	1460	DSPI2_3	DSPI_2_SR[TCF]
281	1464	DSPI2_4	DSPI_2_SR[RFDF]
282	1468	DSPI2_5	DSPI_2_SR[CMD_TCF]
283	146C	DSPI2_6	DSPI_2_SR[CMDFFF]
284	1470	DSPI2_7	DSPI_2_SR[SPEF]
286	1478	DSPI3_0	DSPI_3_SR[TFUF] | DSPI_3_SR[RFOF]| DSPI_3_SR[TFIWF]
287	147C	DSPI3_1	DSPI_3_SR[EOQF]
288	1480	DSPI3_2	DSPI_3_SR[TFFF]
289	1484	DSPI3_3	DSPI_3_SR[TCF]
290	1488	DSPI3_4	DSPI_3_SR[RFDF]
291	148C	DSPI3_5	DSPI_3_SR[CMD_TCF]
292	1490	DSPI3_6	DSPI_3_SR[CMDFFF]
293	1494	DSPI3_7	DSPI_3_SR[SPEF]
295	149C	DSPI4_0	DSPI_4_SR[TFUF] | DSPI_4_SR[RFOF]| DSPI_4_SR[TFIWF]
296	14A0	DSPI4_1	DSPI_4_SR[EOQF]

Table continues on the next page...

Table 17-1. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
297	14A4	DSPI4_2	DSPI_4_SR[TFFF]
298	14A8	DSPI4_3	DSPI_4_SR[TCF]
299	14AC	DSPI4_4	DSPI_4_SR[RFDF]
300	14B0	DSPI4_5	DSPI_4_SR[CMD_TCF]
301	14B4	DSPI4_6	DSPI_4_SR[CMDFFF]
302	14B8	DSPI4_7	DSPI_4_SR[SPEF]
304	14C0	DSPI_M0_0	DSPI_M0_SR[TFUF] | DSPI_M0_SR[RFOF]| DSPI_M0_SR[TFIWF]
305	14C4	DSPI_M0_1	DSPI_M0_SR[EOQF]
306	14C8	DSPI_M0_2	DSPI_M0_SR[TFFF]
307	14CC	DSPI_M0_3	DSPI_M0_SR[TCF]
308	14D0	DSPI_M0_4	DSPI_M0_SR[RFDF]
309	14D4	DSPI_M0_5	DSPI_M0_SR[SPITCF] | DSPI_M0_SR[CMD_TCF]
310	14D8	DSPI_M0_6	DSPI_M0_SR[DSITCF] | DSPI_M0_SR[CMDFFF]
311	14DC	DSPI_M0_7	DSPI_M0_SR[SPEF] | DSPI_M0_SR[DPEF]
312	14E0	DSPI_M0_8	DSPI_M0_SR[DDIF]
313	14E4	DSPI_M1_0	DSPI_M1_SR[TFUF] | DSPI_M1_SR[RFOF]| DSPI_M1_SR[TFIWF]
314	14E8	DSPI_M1_1	DSPI_M1_SR[EOQF]
315	14EC	DSPI_M1_2	DSPI_M1_SR[TFFF]
316	14F0	DSPI_M1_3	DSPI_M1_SR[TCF]
317	14F4	DSPI_M1_4	DSPI_M1_SR[RFDF]
318	14F8	DSPI_M1_5	DSPI_M1_SR[SPITCF] | DSPI_M1_SR[CMD_TCF]
319	14FC	DSPI_M1_6	DSPI_M1_SR[DSITCF] | DSPI_M1_SR[CMDFFF]
320	1500	DSPI_M1_7	DSPI_M1_SR[SPEF] | DSPI_M1_SR[DPEF]
321	1504	DSPI_M1_8	DSPI_M1_SR[DDIF]
376	15E0	Linflex/eSCI0_0	LINFlex_0_RXI
377	15E4	Linflex/eSCI0_1	LINFlex_0_TXI
378	15E8	Linflex/eSCI0_2	LINFlex_0_ERR
380	15F0	Linflex/eSCI1_0	LINFlex_1_RXI
381	15F4	Linflex/eSCI1_1	LINFlex_1_TXI
382	15F8	Linflex/eSCI1_2	LINFlex_1_ERR
384	1600	Linflex/eSCI2_0	LINFlex_2_RXI
385	1604	Linflex/eSCI2_1	LINFlex_2_TXI
386	1608	Linflex/eSCI2_2	LINFlex_2_ERR
388	1610	Linflex/eSCI3_0	LINFlex_3_RXI
389	1614	Linflex/eSCI3_1	LINFlex_3_TXI

Table continues on the next page...

Table 17-1. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
390	1618	Linflex/eSCI3_2	LINFlex_3_ERR
392	1620	Linflex/eSCI_M0_0	LINFlex_M0_RXI
393	1624	Linflex/eSCI_M0_1	LINFlex_M0_TXI
394	1628	Linflex/eSCI_M0_2	LINFlex_M0_ERR
396	1630	Linflex/eSCI_M1_0	LINFlex_M1_RXI
397	1634	Linflex/eSCI_M1_1	LINFlex_M1_TXI
398	1638	Linflex/eSCI_M1_2	LINFlex_M1_ERR
477	1774	Power Monitor Unit	GR_S[VD15] | GR_S[VD14] | GR_S[VD13] | GR_S[VD12] | GR_S[VD10] | GR_S[VD9] | GR_S[VD7] | GR_S[VD4] | GR_S[VD3]
478	1778	Power management Unit (temp sensor)	EPR_TD[TEMP_3] | EPR_TD[TEMP_2] | EPR_TD[TEMP_0]
480	1780	PLL_0_0	PLL0SR[LOLF]
482	1788	PLL_1_0	PLL1SR[LOLF]
484	1790	PLL_0_1	PLL0SR[EXTPDF]
485	1794	PLL_1_1	PLL1SR[EXTPDF]
488	17A0	FCCU_0	FCCU_IRQ_STAT[ALRM_STAT]
489	17A4	FCCU_1	FCU_IRQ_STAT[CFG_TO_STAT]
490	17A8	FCCU_2	FCCU_SCFS[RCCS0]
491	17AC	FCCU_3	FCCU_SCFS[RCCS1]
494	17B8	STCU2_0	STCU_RUNSW[LBIE]
495	17BC	STCU2_1	STCU_RUNSW[MBIE]
528	1840	SAR_ADC0 Combined interrupt	SAR_ADC_0_EOCTU (end of CTU injected conversion) SAR_ADC_0_JEOC (end of injected conversion of each channel) SAR_ADC_0_JECH (end of injected chain) SAR_ADC_0_EOC (end of normal conversion of each channel) SAR_ADC_0_ECH (end of normal chain)
529	1844	SAR_ADC0_ER	SAR_ADC0_ER
530	1848	SAR_ADC0_WD	SAR_ADC0_WD
532	1850	SAR_ADC1 Combined interrupt	SAR_ADC_1_EOCTU (end of CTU injected conversion) SAR_ADC_1_JEOC (end of injected conversion of each channel) SAR_ADC_1_JECH (end of injected chain) SAR_ADC_1_EOC (end of normal conversion of each channel) SAR_ADC_1_ECH (end of normal chain)
533	1854	SAR_ADC1_ER	SAR_ADC1_ER

Table continues on the next page...

Table 17-1. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
534	1858	SAR_ADC1_WD	SAR_ADC1_WD
536	1860	SAR_ADC2 Combined interrupt	SAR_ADC_2_EOCTU (end of CTU injected conversion) SAR_ADC_2_JEOC (end of injected conversion of each channel) SAR_ADC_2_JECH (end of injected chain) SAR_ADC_2_EOC (end of normal conversion of each channel) SAR_ADC_2_ECH (end of normal chain)
537	1864	SAR_ADC2_ER	SAR_ADC2_ER
538	1868	SAR_ADC2_WD	SAR_ADC2_WD
540	1870	SAR_ADC3 Combined interrupt	SAR_ADC_3_EOCTU (end of CTU injected conversion) SAR_ADC_3_JEOC (end of injected conversion of each channel) SAR_ADC_3_JECH (end of injected chain) SAR_ADC_3_EOC (end of normal conversion of each channel) SAR_ADC_3_ECH (end of normal chain)
541	1874	SAR_ADC3_ER	SAR_ADC3_ER
542	1878	SAR_ADC3_WD	SAR_ADC3_WD
544	1880	SD_ADC0	SD_ADC0
545	1884	SD_ADC1	SD_ADC1
546	1888	SD_ADC2	SD_ADC2
558	18B8	SENT_COMBINED_FAST_0	SENT_0_FMSG_RDY[0] | SENT_0_FMSG_RDY[1] | SENT_0_FMSG_RDY[2] | SENT_0_FMSG_RDY[3] | SENT_0_FMSG_RDY[4]
559	18BC	SENT_COMBINED_SLOW_0	SENT_0_SMSG_RDY[0] | SENT_0_SMSG_RDY[1] | SENT_0_SMSG_RDY[2] | SENT_0_SMSG_RDY[3] | SENT_0_SMSG_RDY[4]
560	18C0	SENT_COMBINED_ERR_0	SENT_0_GBL_STATUS[FMDU] | SENT_0_GBL_STATUS[SMDU] | SENT_0_CHn_STATUS[4:7] | SENT_0_CHn_STATUS[9:15]
561	18C4	SENT_COMBINED_FAST_1	SENT_1_FMSG_RDY[0] | SENT_1_FMSG_RDY[1] | SENT_1_FMSG_RDY[2] | SENT_1_FMSG_RDY[3] | SENT_1_FMSG_RDY[4]

Table continues on the next page...

Table 17-1. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
562	18C8	SENT_COMBINED_SLOW_1	SENT_1_SMSG_RDY[0] | SENT_1_SMSG_RDY[1] | SENT_1_SMSG_RDY[2] | SENT_1_SMSG_RDY[3] | SENT_1_SMSG_RDY[4]
563	18CC	SENT_COMBINED_ERR_1	SENT_1_GBL_STATUS[FMDU] | SENT_1_GBL_STATUS[SMDU] | SENT_1_CHn_STATUS[4:7] | SENT_1_CHn_STATUS[9:15]
564	18D0	SENT_0_CH0_FAST	SENT_0_FMSG_RDY[0]
565	18D4	SENT_0_CH0_SLOW	SENT_0_SMSG_RDY[0]
566	18D8	SENT_0_CH0_ERR	SENT_0_CH0_STATUS[4:7] | SENT_0_CH0_STATUS[9:15]
567	18DC	SENT_0_CH1_FAST	SENT_0_FMSG_RDY[1]
568	18E0	SENT_0_CH1_SLOW	SENT_0_SMSG_RDY[1]
569	18E4	SENT_0_CH1_ERR	SENT_0_CH1_STATUS[4:7] | SENT_0_CH1_STATUS[9:15]
570	18E8	SENT_0_CH2_FAST	SENT_0_FMSG_RDY[2]
571	18EC	SENT_0_CH2_SLOW	SENT_0_SMSG_RDY[2]
572	18F0	SENT_0_CH2_ERR	SENT_0_CH2_STATUS[4:7] | SENT_0_CH3_STATUS[9:15]
579	190C	SENT_1_CH0_FAST	SENT_1_FMSG_RDY[0]
580	1910	SENT_1_CH0_SLOW	SENT_1_SMSG_RDY[0]
581	1914	SENT_1_CH0_ERR	SENT_1_CH2_STATUS[4:7] | SENT_1_CH2_STATUS[9:15]
582	1918	SENT_1_CH1_FAST	SENT_1_FMSG_RDY[1]
583	191C	SENT_1_CH1_SLOW	SENT_1_SMSG_RDY[1]
584	1920	SENT_1_CH1_ERR	SENT_1_CH1_STATUS[4:7] | SENT_1_CH1_STATUS[9:15]
585	1924	SENT_1_CH2_FAST	SENT_1_FMSG_RDY[2]
586	1928	SENT_1_CH2_SLOW	SENT_1_SMSG_RDY[2]
587	192C	SENT_1_CH2_ERR	SENT_1_CH3_STATUS[4:7] | SENT_1_CH3_STATUS[9:15]
654	1A38	SIPI_0	SIPI_ERR[TOEn] | SIPI_ERR[TIDEn] | SIPI_ERR[ACKEn]
655	1A3C	SIPI_1	SIPI_SR[GCRCE)
656	1A40	SIPI_2	SIPI_CSR0[RAR] | SIPI_CSR0[ACKR]
657	1A44	SIPI_3	SIPI_CSR1[RAR] | SIPI_CSR1[ACKR]
658	1A48	SIPI_4	SIPI_CSR2[RAR] | SIPI_CSR2[ACKR]
659	1A4C	SIPI_5	SIPI_CSR3[RAR] | SIPI_CSR3[ACKR]
660	1A50	SIPI_6	SIPI_SR[TE0] | SIPI_SR[TE1] |SIPI_SR[TE2] | SIPI_SR[TE3] |SIPI_SR[MCR]

Table continues on the next page...

Table 17-1. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
661	1A54	LFAST0_0	LFAST_0_TISR[TXPNGF] | LFAST_0_TISR[TXUNSF] | LFAST_0_TISR[TXICLCF] | LFAST_0_TISR[TXDTF]
662	1A58	LFAST0_1	LFAST_0_TISR[TXIEF] | LFAST_0_TISR[TXOVF]
663	1A5C	LFAST0_2	LFAST_0_RISR[RXCTSF] | LFAST_0_RISR[RXDF] | LFAST_0_RISR[RXUNSF]
664	1A60	LFAST0_3	LFAST_0_RISR[RXUOF] | LFAST_0_RISR[RXMNF] | LFAST_0_RISR[RXMXF] | LFAST_0_RISR[RXUFF] | LFAST_0_RISR[RXOFF] | LFAST_0_RISR[RXSZF] | LFAST_0_RISR[RXICF] | LFAST_0_RISR[RXLCEF]
665	1A64	LFAST0_4	LFAST_0_RIISR[ICPFF] | LFAST_0_RIISR[ICPSF] | LFAST_0_RIISR[ICPRF] | LFAST_0_RIISR[ICTOF] | LFAST_0_RIISR[ICLPF] | LFAST_0_RIISR[ICCTF] | LFAST_0_RIISR[ICTDF] | LFAST_0_RIISR[ICTEF] | LFAST_0_RIISR[ICRFF] | LFAST_0_RIISR[ICRSF] | LFAST_0_RIISR[ICTFF] | LFAST_0_RIISR[ICTSF] | LFAST_0_RIISR[ICPOFF] | LFAST_0_RIISR[ICPONF]
674	1A88	JTAGM	JTAGM_SR[SPU_INT] | JTAGM_SR[Idle]
675	1A8C	JDC	JDC_MSR[JIN_INT] | JDC_MSR[JOUT_INT]
677	1A94	FlexCAN_0_0	CAN0_ESR1[BOFFINT] | CAN0_ESR1[BOFFDONEINT]
678	1A98	FlexCAN_0_1	CAN0_ESR1[ERRINT] | CAN0_ESR1[ERRINT_FAST]
679	1A9C	FlexCAN_0_5	CAN0_IFLAG1[BUF15:BUF0]
680	1AA0	FlexCAN_0_2	CAN0_IFLAG1[BUF31:BUF16]
681	1AA4	FlexCAN_0_3	CAN0_IFLAG2[BUF63:BUF32]
682	1AA8	FlexCAN_0_4	CAN0_IFLAG3[BUF95:BUF64]
683	1AAC	FlexCAN_1_0	CAN1_ESR1[BOFFINT] | CAN1_ESR1[BOFFDONEINT]
684	1AB0	FlexCAN_1_1	CAN1_ESR1[ERRINT] | CAN1_ESR1[ERRINT_FAST]
685	1AB4	FlexCAN_1_5	CAN1_IFLAG1[BUF15:BUF0]
686	1AB8	FlexCAN_1_2	CAN1_IFLAG1[BUF31:BUF16]
687	1ABC	FlexCAN_1_3	CAN1_IFLAG2[BUF63:BUF32]

Table continues on the next page...

Table 17-1. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
688	1AC0	FlexCAN_1_4	CAN1_IFLAG3[BUF95:BUF64]
689	1AC4	FlexCAN_2_0	CAN2_ESR1[BOFFINT]
690	1AC8	FlexCAN_2_1	CAN2_ESR1[ERRINT]
691	1ACC	FlexCAN_2_5	CAN2_IFLAG1[BUF7:BUF0]
692	1AD0	FlexCAN_2_2	CAN2_IFLAG1[BUF15:BUF8]
693	1AD4	FlexCAN_2_3	CAN2_IFLAG1[BUF31:BUF16]
694	1AD8	FlexCAN_2_4	CAN2_IFLAG2[BUF63:BUF32]
695	1ADC	FlexCAN_3_0	CAN3_ESR1[BOFFINT]
696	1AE0	FlexCAN_3_1	CAN3_ESR1[ERRINT]
697	1AE4	FlexCAN_3_5	CAN3_IFLAG1[BUF7:BUF0]
698	1AE8	FlexCAN_3_2	CAN3_IFLAG1[BUF15:BUF8]
699	1AEC	FlexCAN_3_3	CAN3_IFLAG1[BUF31:BUF16]
700	1AF0	FlexCAN_3_4	CAN3_IFLAG2[BUF63:BUF32]
706	1B08	eMIOS_0 channel 8	EMIOS_0_GFR[F8]
707	1B0C	eMIOS_0 channel 9	EMIOS_0_GFR[F9]
708	1B10	eMIOS_0 channel 10	EMIOS_0_GFR[F10]
709	1B14	eMIOS_0 channel 11	EMIOS_0_GFR[F11]
710	1B18	eMIOS_0 channel 12	EMIOS_0_GFR[F12]
711	1B1C	eMIOS_0 channel 13	EMIOS_0_GFR[F13]
712	1B20	eMIOS_0 channel 14	EMIOS_0_GFR[F14]
713	1B24	eMIOS_0 channel 15	EMIOS_0_GFR[F15]
714	1B28	eMIOS_0 channel 16	EMIOS_0_GFR[F16]
715	1B2C	eMIOS_0 channel 17	EMIOS_0_GFR[F17]
716	1B30	eMIOS_0 channel 18	EMIOS_0_GFR[F18]
717	1B34	eMIOS_0 channel 19	EMIOS_0_GFR[F19]
718	1B38	eMIOS_0 channel 20	EMIOS_0_GFR[F20]
719	1B3C	eMIOS_0 channel 21	EMIOS_0_GFR[F21]
720	1B40	eMIOS_0 channel 22	EMIOS_0_GFR[F22]
721	1B44	eMIOS_0 channel 23	EMIOS_0_GFR[F23]
722	1B48	eMIOS_1 channel 8	EMIOS_1_GFR[F8]
723	1B4C	eMIOS_1 channel 9	EMIOS_1_GFR[F9]
724	1B50	eMIOS_1 channel 10	EMIOS_1_GFR[F10]
725	1B54	eMIOS_1 channel 11	EMIOS_1_GFR[F11]
726	1B58	eMIOS_1 channel 12	EMIOS_1_GFR[F12]
727	1B5C	eMIOS_1 channel 13	EMIOS_1_GFR[F13]
728	1B60	eMIOS_1 channel 14	EMIOS_1_GFR[F14]
729	1B64	eMIOS_1 channel 15	EMIOS_1_GFR[F15]
730	1B68	eMIOS_1 channel 16	EMIOS_1_GFR[F16]
731	1B6C	eMIOS_1 channel 17	EMIOS_1_GFR[F17]

Table continues on the next page...

Table 17-1. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
732	1B70	eMIOS_1 channel 18	EMIOS_1_GFR[F18]
733	1B74	eMIOS_1 channel 19	EMIOS_1_GFR[F19]
734	1B78	eMIOS_1 channel 20	EMIOS_1_GFR[F20]
735	1B7C	eMIOS_1 channel 21	EMIOS_1_GFR[F21]
736	1B80	eMIOS_1 channel 22	EMIOS_1_GFR[F22]
737	1B84	eMIOS_1 channel 23	EMIOS_1_GFR[F23]
738	1B88	eTPU Engine 0 and 1 Global Exception	ETPU_MCRAB[MGEA,MGEB,ILFA,ILFB,SCMMISF]
739	1B8C	eTPU Engine 0 Channel 0	ETPU_CISR_0[CIS0]
740	1B90	eTPU Engine 0 Channel 1	ETPU_CISR_0[CIS1]
741	1B94	eTPU Engine 0 Channel 2	ETPU_CISR_0[CIS2]
742	1B98	eTPU Engine 0 Channel 3	ETPU_CISR_0[CIS3]
743	1B9C	eTPU Engine 0 Channel 4	ETPU_CISR_0[CIS4]
744	1BA0	eTPU Engine 0 Channel 5	ETPU_CISR_0[CIS5]
745	1BA4	eTPU Engine 0 Channel 6	ETPU_CISR_0[CIS6]
746	1BA8	eTPU Engine 0 Channel 7	ETPU_CISR_0[CIS7]
747	1BAC	eTPU Engine 0 Channel 8	ETPU_CISR_0[CIS8]
748	1BB0	eTPU Engine 0 Channel 9	ETPU_CISR_0[CIS9]
749	1BB4	eTPU Engine 0 Channel 10	ETPU_CISR_0[CIS10]
750	1BB8	eTPU Engine 0 Channel 11	ETPU_CISR_0[CIS11]
751	1BBC	eTPU Engine 0 Channel 12	ETPU_CISR_0[CIS12]
752	1BC0	eTPU Engine 0 Channel 13	ETPU_CISR_0[CIS13]
753	1BC4	eTPU Engine 0 Channel 14	ETPU_CISR_0[CIS14]
754	1BC8	eTPU Engine 0 Channel 15	ETPU_CISR_0[CIS15]
755	1BCC	eTPU Engine 0 Channel 16	ETPU_CISR_0[CIS16]
756	1BD0	eTPU Engine 0 Channel 17	ETPU_CISR_0[CIS17]
757	1BD4	eTPU Engine 0 Channel 18	ETPU_CISR_0[CIS18]
758	1BD8	eTPU Engine 0 Channel 19	ETPU_CISR_0[CIS19]
759	1BDC	eTPU Engine 0 Channel 20	ETPU_CISR_0[CIS20]
760	1BE0	eTPU Engine 0 Channel 21	ETPU_CISR_0[CIS21]
761	1BE4	eTPU Engine 0 Channel 22	ETPU_CISR_0[CIS22]
762	1BE8	eTPU Engine 0 Channel 23	ETPU_CISR_0[CIS23]
763	1BEC	eTPU Engine 0 Channel 24	ETPU_CISR_0[CIS24]
764	1BF0	eTPU Engine 0 Channel 25	ETPU_CISR_0[CIS25]
765	1BF4	eTPU Engine 0 Channel 26	ETPU_CISR_0[CIS26]
766	1BF8	eTPU Engine 0 Channel 27	ETPU_CISR_0[CIS27]
767	1BFC	eTPU Engine 0 Channel 28	ETPU_CISR_0[CIS28]
768	1C00	eTPU Engine 0 Channel 29	ETPU_CISR_0[CIS29]
769	1C04	eTPU Engine 0 Channel 30	ETPU_CISR_0[CIS30]
770	1C08	eTPU Engine 0 Channel 31	ETPU_CISR_0[CIS31]

Table continues on the next page...

Table 17-1. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
771	1C0C	eTPU Engine 1 Channel 0	ETPU_CISR_1[CIS0]
772	1C10	eTPU Engine 1 Channel 1	ETPU_CISR_1[CIS1]
773	1C14	eTPU Engine 1 Channel 2	ETPU_CISR_1[CIS2]
774	1C18	eTPU Engine 1 Channel 3	ETPU_CISR_1[CIS3]
775	1C1C	eTPU Engine 1 Channel 4	ETPU_CISR_1[CIS4]
776	1C20	eTPU Engine 1 Channel 5	ETPU_CISR_1[CIS5]
777	1C24	eTPU Engine 1 Channel 6	ETPU_CISR_1[CIS6]
778	1C28	eTPU Engine 1 Channel 7	ETPU_CISR_1[CIS7]
779	1C2C	eTPU Engine 1 Channel 8	ETPU_CISR_1[CIS8]
780	1C30	eTPU Engine 1 Channel 9	ETPU_CISR_1[CIS9]
781	1C34	eTPU Engine 1 Channel 10	ETPU_CISR_1[CIS10]
782	1C38	eTPU Engine 1 Channel 11	ETPU_CISR_1[CIS11]
783	1C3C	eTPU Engine 1 Channel 12	ETPU_CISR_1[CIS12]
784	1C40	eTPU Engine 1 Channel 13	ETPU_CISR_1[CIS13]
785	1C44	eTPU Engine 1 Channel 14	ETPU_CISR_1[CIS14]
786	1C48	eTPU Engine 1 Channel 15	ETPU_CISR_1[CIS15]
787	1C4C	eTPU Engine 1 Channel 16	ETPU_CISR_1[CIS16]
788	1C50	eTPU Engine 1 Channel 17	ETPU_CISR_1[CIS17]
789	1C54	eTPU Engine 1 Channel 18	ETPU_CISR_1[CIS18]
790	1C58	eTPU Engine 1 Channel 19	ETPU_CISR_1[CIS19]
791	1C5C	eTPU Engine 1 Channel 20	ETPU_CISR_1[CIS20]
792	1C60	eTPU Engine 1 Channel 21	ETPU_CISR_1[CIS21]
793	1C64	eTPU Engine 1 Channel 22	ETPU_CISR_1[CIS22]
794	1C68	eTPU Engine 1 Channel 23	ETPU_CISR_1[CIS23]
795	1C6C	eTPU Engine 1 Channel 24	ETPU_CISR_1[CIS24]
796	1C70	eTPU Engine 1 Channel 25	ETPU_CISR_1[CIS25]
797	1C74	eTPU Engine 1 Channel 26	ETPU_CISR_1[CIS26]
798	1C78	eTPU Engine 1 Channel 27	ETPU_CISR_1[CIS27]
799	1C7C	eTPU Engine 1 Channel 28	ETPU_CISR_1[CIS28]
800	1C80	eTPU Engine 1 Channel 29	ETPU_CISR_1[CIS29]
801	1C84	eTPU Engine 1 Channel 30	ETPU_CISR_1[CIS30]
802	1C88	eTPU Engine 1 Channel 31	ETPU_CISR_1[CIS31]
900	1E10	BCTU_0_TRIG	BCTU_0_TRIG
901	1E14	BCTU_0_ADC0	BCTU_0_ADC0
902	1E18	BCTU_0_ADC1	BCTU_0_ADC1
903	1E1C	BCTU_0_ADC2	BCTU_0_ADC2
904	1E20	BCTU_0_ADC3	BCTU_0_ADC3
905	1E24	BCTU_0_LIST	BCTU_0_LIST
915	1E4C	REACM_G	ipi_int_reacm

Table continues on the next page...

Table 17-1. Interrupt sources (continued)

IRQ #	Offset	Source description	Source name
916	1E50	REACM_0	ipi_int_reacm_ch0
917	1E54	REACM_1	ipi_int_reacm_ch1
918	1E58	REACM_2	ipi_int_reacm_ch2
919	1E5C	REACM_3	ipi_int_reacm_ch3
920	1E60	REACM_4	ipi_int_reacm_ch4
921	1E64	REACM_5	ipi_int_reacm_ch5
922	1E68	REACM_6	ipi_int_reacm_ch6
923	1E6C	REACM_7	ipi_int_reacm_ch7
924	1E70	REACM_8	ipi_int_reacm_ch8
925	1E74	REACM_9	ipi_int_reacm_ch9
932	1E90	DECFIL_0_In	ipi_int_dec_filter_0_in
933	1E94	DECFIL_0_Out	ipi_int_dec_filter_0_out
934	1E98	DECFIL_0_Err	ipi_int_dec_filter_0_err
935	1E9C	DECFIL_1_In	ipi_int_dec_filter_1_in
936	1EA0	DECFIL_1_Out	ipi_int_dec_filter_1_out
937	1EA4	DECFIL_1_Err	ipi_int_dec_filter_1_err

17.2 Introduction

The INTC:

- Provides priority-based preemptive scheduling of interrupt requests
- Schedules interrupt requests (IRQs) from software and internal peripherals to one or more processors (PRCs)
- Provides interrupt prioritization and preemption, interrupt masking, interrupt priority elevation, and protocol support

This scheduling scheme is suitable for statically scheduled hard real-time systems.

The INTC is targeted to work with a Power Architecture processor and is capable of processing high-demand interrupt sources where the Interrupt Service Routines (ISRs) nest to multiple levels, but it also can be used with other processors and applications.

For high-priority interrupt requests in these target applications, it is necessary to minimize the interval between the assertion of the interrupt request from the peripheral and the point at which the processor is performing useful work to service the interrupt request. The INTC supports this goal by providing a unique vector for each interrupt

request source. It also provides 64 priorities so that lower priority ISRs do not delay the execution of higher priority ISRs. Since each individual application will have different priorities for each source of interrupt request, the priority of each interrupt request is configurable.

When multiple tasks share a resource, coherent accesses to that resource need to be supported. The INTC supports the Priority Ceiling Protocol for coherent accesses. By providing a modifiable priority mask, the priority can be raised temporarily so that all tasks which share the resource are unable to preempt each other.

Multiple processors can assert interrupt requests to each other through software-settable interrupt requests. These same software-settable interrupt requests can also be used to separate the work involved in servicing an interrupt request into two parts, a high-priority portion and a low-priority portion. The high-priority portion is initiated by a peripheral interrupt request, but then the ISR can assert a software-settable interrupt request to finish the servicing in a lower priority ISR. Therefore these software-settable interrupt requests can be used instead of having the peripheral ISR schedule a task through the RTOS.

17.3 Block diagram

The following figure shows the block diagram of an INTC with four processors (PRC0...PRC3). The actual number of processors is described in the chip-specific INTC information. The structure of the INTC remains the same for each implemented processor.

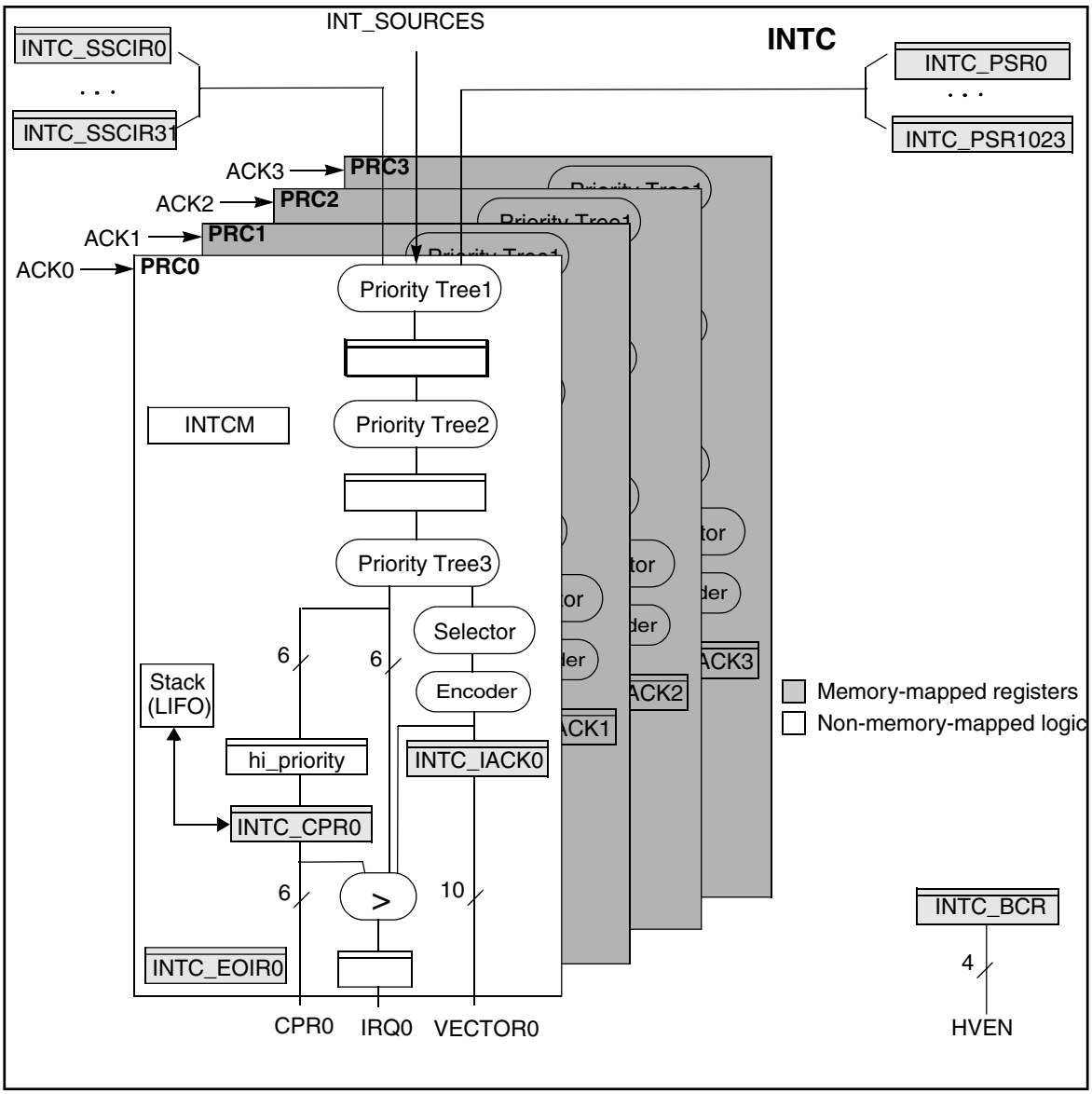


Figure 17-1. Block diagram for an INTC with four processors

17.4 Features

- Each peripheral interrupt source is software-steerable to any combination of interrupt request outputs to the following:
  - Processor 0
  - Processor 1
- 32 software-settable interrupt request sources
- 10-bit vector

    - Unique vector for each interrupt request source
    - Hardware connection to processor or read from register
- Each interrupt source can be programmed to one of 64 priorities
- Each interrupt source can be triggered by software
- Preemption
    - Preemptive prioritized interrupt requests to processor
    - ISR with higher priority preempts ISRs or tasks with lower priorities
    - Automatic pushing or popping of preempted priority to or from a LIFO
    - Ability to modify the ISR or task priority; modifying the priority can be used to implement the Priority Ceiling Protocol for accessing shared resources
    - 3 INTC clock cycles from interrupt request into interrupt request to CPU
- Low latency
    - 3 INTC clock cycles from receipt of interrupt request from peripheral to interrupt request to processor; four clock cycles from receipt of software request
- Monitor
    - The Interrupt Controller Monitor (INTCM) provides a way to set a maximum timer count for the interrupt latency from interrupt request (IRQ) to interrupt acknowledge (IACK) or from IRQ to return from interrupt (RFI). The purpose of this function is to provide a mechanism to monitor the latency of interrupt sources for a corresponding Processor to ensure that these critical interrupts execute within the expected window of time, increasing the reliability of the device. The hardware for this function is isolated in its own hierarchy to decrease the risk of fault interference.

17.5 Modes of operation

The interrupt controller has two handshaking modes with the processor: software vector mode and hardware vector mode. The state of the hardware vector enable bit, INTC_BCR[HVEN], determines which mode is used. In Power Architecture devices, software vector mode uses the autovector mode of the Processor for a single entry point for the ISR (16 bytes of vector space available), whereas the hardware vector mode uses the non-autovector mode where the vector offset from the INTC is provided to the Processor (4 bytes for each IRQ).

In debug mode the interrupt controller operation is identical to its normal operation of software vector mode or hardware vector mode.

17.5.1 Software vector mode

In software vector mode, software (that is, the interrupt exception handler) must read a register in the INTC to obtain the vector associated with the interrupt request to the processor. The INTC will use software vector mode for a given processor when its associated HVEN_PRCn bit in the INTC_BCR is negated. The hardware vector enable signal to any processor is driven as negated when its associated HVEN_PRC*n* bit is negated. The vector is read from the INTC_IACKRn registers. Reading the INTC_IACKR*n* negates the interrupt request to the associated processor. Even if a higher priority interrupt request has arrived while waiting for this interrupt acknowledge, the interrupt request to the processor will negate for at least one clock. The reading also pushes the PRI value in the INTC_CPRn onto the associated LIFO, and updates PRI in the associated INTC_CPR_PRC*n* with the new priority.

17.5.2 Hardware vector mode

In hardware vector mode, the hardware causes the first instruction that will be executed (when handling the interrupt request to the processor) to be an instruction that is specific to that vector. Therefore the interrupt exception handler is specific to a peripheral or software-settable interrupt request, rather than being common to all of them. The INTC will use hardware vector mode for a given processor when its associated HVENn bit in the INTC_BCR is asserted. The hardware vector enable signal to the associated processor is driven as asserted.

When the interrupt request to the associated processor asserts, the interrupt vector signal is updated. The value of that interrupt vector is the unique vector associated with the preempting peripheral or software-settable interrupt request. The vector value matches the value of the INTVEC field in the INTC_IACKRn, depending on which processor was assigned to handle a given interrupt source.

The assertion of the interrupt acknowledge signal for a given processor pushes the associated PRI value in the associated INTC_CPR*n* register onto the associated LIFO and updates the associated PRI in the associated INTC_CPR*n* register with the new priority. This pushing of the PRI value onto the associated LIFO and updating PRI in the associated INTC_CPR*n* does not occur when the associated interrupt acknowledge signal asserts and the INTC_SSCIRs is written at a time such that the PRI value in the associated INTC_CPR*n* register would need to be pushed and the previously last pushed

PRI value would need to be popped simultaneously. In this case, PRI in the associated INTC_CPR*n* is updated with the new priority, and the associated LIFO is neither pushed or popped.

17.6 Memory map and register definition

A transfer error will be asserted if an access is attempted outside of the memory map or for any unimplemented registers. For example, if the design has only 512 interrupt sources and a source > 512 is accessed, a transfer error is asserted.

With the exception of the INTC_SSCIR *n* and INTC_PSR *n* registers, all of the registers are 32-bit. Any combination of accessing the four bytes of a register with a single access is supported, provided that the access does not cross a register boundary. These supported accesses include types and sizes of 8 bits, aligned 16 bits, misaligned 16 bits to the middle two bytes, and aligned 32 bits.

Although INTC_SSCIR*n* are 8 bits wide and INTC_PSR*n* are 16 bits wide, they can be accessed with a single 16-bit or 32-bit access, provided that the access does not cross a 32-bit boundary.

Some registers have specific exceptions to these rules, as outlined in their definitions.

When writing to INTC_PSRs or INTC_SSCIRs, the following restrictions (illustrated in Figure 17-2) apply:

- For PSRs, write accesses must not span 16-bit boundaries where one register is implemented and one is unimplemented (unimplemented interrupt source).
- For SSCIRs, write accesses must not span 8-bit boundaries where one register is implemented and one is unimplemented (unimplemented interrupt source).

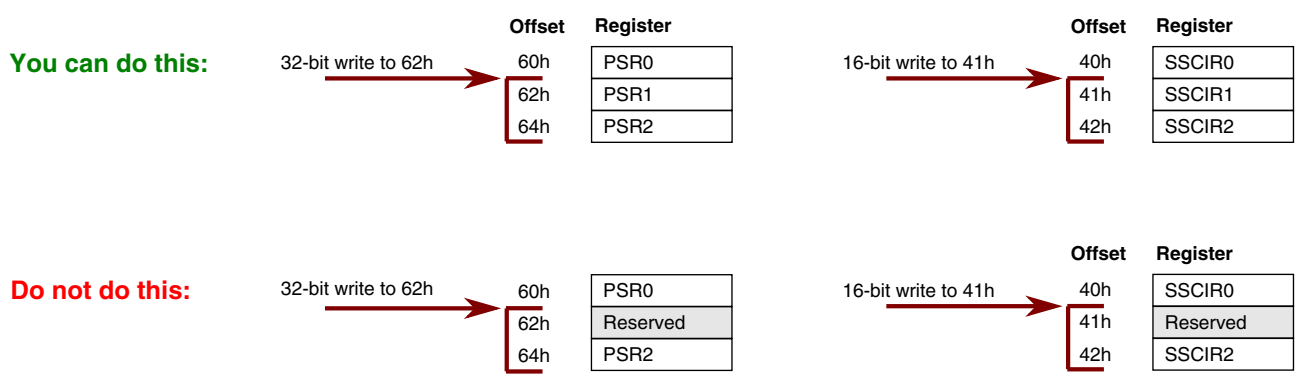


Figure 17-2. INTC_PSR*n* and INTC_SSCIR*n* write restrictions

In software vector mode, the effects of a read of the INTC Interrupt Acknowledge register (INTC_IACKR *n*) are the same regardless of the size of the read. In either software or hardware vector mode, the size of a write to the INTC end-of-interrupt register (INTC_EOIR *n*) does not affect the operation of the write.

INTC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	INTC Block Configuration Register (INTC_BCR)	32	R/W	0000_0000h	17.6.1/444
4	INTC Master Protection Register (INTC_MPROT)	32	R/W	0000_0000h	17.6.2/446
10	INTC Current Priority Register for Processor 0 (INTC_CPR0)	32	R/W	0000_003Fh	17.6.3/447
14	INTC Current Priority Register for Processor 1 (INTC_CPR1)	32	R/W	0000_003Fh	17.6.4/448
20	INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)	32	R/W	0000_0000h	17.6.5/449
24	INTC Interrupt Acknowledge Register for Processor 1 (INTC_IACKR1)	32	R/W	0000_0000h	17.6.6/450
30	INTC End Of Interrupt Register for Processor 0 (INTC_EOIR0)	32	W	0000_0000h	17.6.7/451
34	INTC End Of Interrupt Register for Processor 1 (INTC_EOIR1)	32	W	0000_0000h	17.6.8/451
40	INTC Software Set/Clear Interrupt Register (INTC_SSCIR0)	8	R/W	00h	17.6.9/452
41	INTC Software Set/Clear Interrupt Register (INTC_SSCIR1)	8	R/W	00h	17.6.9/452
42	INTC Software Set/Clear Interrupt Register (INTC_SSCIR2)	8	R/W	00h	17.6.9/452
43	INTC Software Set/Clear Interrupt Register (INTC_SSCIR3)	8	R/W	00h	17.6.9/452
44	INTC Software Set/Clear Interrupt Register (INTC_SSCIR4)	8	R/W	00h	17.6.9/452
45	INTC Software Set/Clear Interrupt Register (INTC_SSCIR5)	8	R/W	00h	17.6.9/452
46	INTC Software Set/Clear Interrupt Register (INTC_SSCIR6)	8	R/W	00h	17.6.9/452
47	INTC Software Set/Clear Interrupt Register (INTC_SSCIR7)	8	R/W	00h	17.6.9/452
48	INTC Software Set/Clear Interrupt Register (INTC_SSCIR8)	8	R/W	00h	17.6.9/452
49	INTC Software Set/Clear Interrupt Register (INTC_SSCIR9)	8	R/W	00h	17.6.9/452
4A	INTC Software Set/Clear Interrupt Register (INTC_SSCIR10)	8	R/W	00h	17.6.9/452
4B	INTC Software Set/Clear Interrupt Register (INTC_SSCIR11)	8	R/W	00h	17.6.9/452
4C	INTC Software Set/Clear Interrupt Register (INTC_SSCIR12)	8	R/W	00h	17.6.9/452
4D	INTC Software Set/Clear Interrupt Register (INTC_SSCIR13)	8	R/W	00h	17.6.9/452
4E	INTC Software Set/Clear Interrupt Register (INTC_SSCIR14)	8	R/W	00h	17.6.9/452
4F	INTC Software Set/Clear Interrupt Register (INTC_SSCIR15)	8	R/W	00h	17.6.9/452
50	INTC Software Set/Clear Interrupt Register (INTC_SSCIR16)	8	R/W	00h	17.6.9/452

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
51	INTC Software Set/Clear Interrupt Register (INTC_SSCIR17)	8	R/W	00h	17.6.9/452
52	INTC Software Set/Clear Interrupt Register (INTC_SSCIR18)	8	R/W	00h	17.6.9/452
53	INTC Software Set/Clear Interrupt Register (INTC_SSCIR19)	8	R/W	00h	17.6.9/452
54	INTC Software Set/Clear Interrupt Register (INTC_SSCIR20)	8	R/W	00h	17.6.9/452
55	INTC Software Set/Clear Interrupt Register (INTC_SSCIR21)	8	R/W	00h	17.6.9/452
56	INTC Software Set/Clear Interrupt Register (INTC_SSCIR22)	8	R/W	00h	17.6.9/452
57	INTC Software Set/Clear Interrupt Register (INTC_SSCIR23)	8	R/W	00h	17.6.9/452
58	INTC Software Set/Clear Interrupt Register (INTC_SSCIR24)	8	R/W	00h	17.6.9/452
59	INTC Software Set/Clear Interrupt Register (INTC_SSCIR25)	8	R/W	00h	17.6.9/452
5A	INTC Software Set/Clear Interrupt Register (INTC_SSCIR26)	8	R/W	00h	17.6.9/452
5B	INTC Software Set/Clear Interrupt Register (INTC_SSCIR27)	8	R/W	00h	17.6.9/452
5C	INTC Software Set/Clear Interrupt Register (INTC_SSCIR28)	8	R/W	00h	17.6.9/452
5D	INTC Software Set/Clear Interrupt Register (INTC_SSCIR29)	8	R/W	00h	17.6.9/452
5E	INTC Software Set/Clear Interrupt Register (INTC_SSCIR30)	8	R/W	00h	17.6.9/452
5F	INTC Software Set/Clear Interrupt Register (INTC_SSCIR31)	8	R/W	00h	17.6.9/452
60	INTC Priority Select Register (INTC_PSR0)	16	R/W	8000h	17.6.10/454
62	INTC Priority Select Register (INTC_PSR1)	16	R/W	8000h	17.6.10/454
64	INTC Priority Select Register (INTC_PSR2)	16	R/W	8000h	17.6.10/454
66	INTC Priority Select Register (INTC_PSR3)	16	R/W	8000h	17.6.10/454
68	INTC Priority Select Register (INTC_PSR4)	16	R/W	8000h	17.6.10/454
6A	INTC Priority Select Register (INTC_PSR5)	16	R/W	8000h	17.6.10/454
6C	INTC Priority Select Register (INTC_PSR6)	16	R/W	8000h	17.6.10/454
6E	INTC Priority Select Register (INTC_PSR7)	16	R/W	8000h	17.6.10/454
70	INTC Priority Select Register (INTC_PSR8)	16	R/W	8000h	17.6.10/454
72	INTC Priority Select Register (INTC_PSR9)	16	R/W	8000h	17.6.10/454
74	INTC Priority Select Register (INTC_PSR10)	16	R/W	8000h	17.6.10/454
76	INTC Priority Select Register (INTC_PSR11)	16	R/W	8000h	17.6.10/454
78	INTC Priority Select Register (INTC_PSR12)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
7A	INTC Priority Select Register (INTC_PSR13)	16	R/W	8000h	17.6.10/454
7C	INTC Priority Select Register (INTC_PSR14)	16	R/W	8000h	17.6.10/454
7E	INTC Priority Select Register (INTC_PSR15)	16	R/W	8000h	17.6.10/454
80	INTC Priority Select Register (INTC_PSR16)	16	R/W	8000h	17.6.10/454
82	INTC Priority Select Register (INTC_PSR17)	16	R/W	8000h	17.6.10/454
84	INTC Priority Select Register (INTC_PSR18)	16	R/W	8000h	17.6.10/454
86	INTC Priority Select Register (INTC_PSR19)	16	R/W	8000h	17.6.10/454
88	INTC Priority Select Register (INTC_PSR20)	16	R/W	8000h	17.6.10/454
8A	INTC Priority Select Register (INTC_PSR21)	16	R/W	8000h	17.6.10/454
8C	INTC Priority Select Register (INTC_PSR22)	16	R/W	8000h	17.6.10/454
8E	INTC Priority Select Register (INTC_PSR23)	16	R/W	8000h	17.6.10/454
90	INTC Priority Select Register (INTC_PSR24)	16	R/W	8000h	17.6.10/454
92	INTC Priority Select Register (INTC_PSR25)	16	R/W	8000h	17.6.10/454
94	INTC Priority Select Register (INTC_PSR26)	16	R/W	8000h	17.6.10/454
96	INTC Priority Select Register (INTC_PSR27)	16	R/W	8000h	17.6.10/454
98	INTC Priority Select Register (INTC_PSR28)	16	R/W	8000h	17.6.10/454
9A	INTC Priority Select Register (INTC_PSR29)	16	R/W	8000h	17.6.10/454
9C	INTC Priority Select Register (INTC_PSR30)	16	R/W	8000h	17.6.10/454
9E	INTC Priority Select Register (INTC_PSR31)	16	R/W	8000h	17.6.10/454
A0	INTC Priority Select Register (INTC_PSR32)	16	R/W	8000h	17.6.10/454
A2	INTC Priority Select Register (INTC_PSR33)	16	R/W	8000h	17.6.10/454
A4	INTC Priority Select Register (INTC_PSR34)	16	R/W	8000h	17.6.10/454
A6	INTC Priority Select Register (INTC_PSR35)	16	R/W	8000h	17.6.10/454
A8	INTC Priority Select Register (INTC_PSR36)	16	R/W	8000h	17.6.10/454
AA	INTC Priority Select Register (INTC_PSR37)	16	R/W	8000h	17.6.10/454
AC	INTC Priority Select Register (INTC_PSR38)	16	R/W	8000h	17.6.10/454
AE	INTC Priority Select Register (INTC_PSR39)	16	R/W	8000h	17.6.10/454
B0	INTC Priority Select Register (INTC_PSR40)	16	R/W	8000h	17.6.10/454
B2	INTC Priority Select Register (INTC_PSR41)	16	R/W	8000h	17.6.10/454
B4	INTC Priority Select Register (INTC_PSR42)	16	R/W	8000h	17.6.10/454
B6	INTC Priority Select Register (INTC_PSR43)	16	R/W	8000h	17.6.10/454
B8	INTC Priority Select Register (INTC_PSR44)	16	R/W	8000h	17.6.10/454
BA	INTC Priority Select Register (INTC_PSR45)	16	R/W	8000h	17.6.10/454
BC	INTC Priority Select Register (INTC_PSR46)	16	R/W	8000h	17.6.10/454
BE	INTC Priority Select Register (INTC_PSR47)	16	R/W	8000h	17.6.10/454
C0	INTC Priority Select Register (INTC_PSR48)	16	R/W	8000h	17.6.10/454
C2	INTC Priority Select Register (INTC_PSR49)	16	R/W	8000h	17.6.10/454
C4	INTC Priority Select Register (INTC_PSR50)	16	R/W	8000h	17.6.10/454
C6	INTC Priority Select Register (INTC_PSR51)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C8	INTC Priority Select Register (INTC_PSR52)	16	R/W	8000h	17.6.10/454
CA	INTC Priority Select Register (INTC_PSR53)	16	R/W	8000h	17.6.10/454
CC	INTC Priority Select Register (INTC_PSR54)	16	R/W	8000h	17.6.10/454
CE	INTC Priority Select Register (INTC_PSR55)	16	R/W	8000h	17.6.10/454
D0	INTC Priority Select Register (INTC_PSR56)	16	R/W	8000h	17.6.10/454
D2	INTC Priority Select Register (INTC_PSR57)	16	R/W	8000h	17.6.10/454
D4	INTC Priority Select Register (INTC_PSR58)	16	R/W	8000h	17.6.10/454
D6	INTC Priority Select Register (INTC_PSR59)	16	R/W	8000h	17.6.10/454
D8	INTC Priority Select Register (INTC_PSR60)	16	R/W	8000h	17.6.10/454
DA	INTC Priority Select Register (INTC_PSR61)	16	R/W	8000h	17.6.10/454
DC	INTC Priority Select Register (INTC_PSR62)	16	R/W	8000h	17.6.10/454
DE	INTC Priority Select Register (INTC_PSR63)	16	R/W	8000h	17.6.10/454
E0	INTC Priority Select Register (INTC_PSR64)	16	R/W	8000h	17.6.10/454
E2	INTC Priority Select Register (INTC_PSR65)	16	R/W	8000h	17.6.10/454
E4	INTC Priority Select Register (INTC_PSR66)	16	R/W	8000h	17.6.10/454
E6	INTC Priority Select Register (INTC_PSR67)	16	R/W	8000h	17.6.10/454
E8	INTC Priority Select Register (INTC_PSR68)	16	R/W	8000h	17.6.10/454
EA	INTC Priority Select Register (INTC_PSR69)	16	R/W	8000h	17.6.10/454
EC	INTC Priority Select Register (INTC_PSR70)	16	R/W	8000h	17.6.10/454
EE	INTC Priority Select Register (INTC_PSR71)	16	R/W	8000h	17.6.10/454
F0	INTC Priority Select Register (INTC_PSR72)	16	R/W	8000h	17.6.10/454
F2	INTC Priority Select Register (INTC_PSR73)	16	R/W	8000h	17.6.10/454
F4	INTC Priority Select Register (INTC_PSR74)	16	R/W	8000h	17.6.10/454
F6	INTC Priority Select Register (INTC_PSR75)	16	R/W	8000h	17.6.10/454
F8	INTC Priority Select Register (INTC_PSR76)	16	R/W	8000h	17.6.10/454
FA	INTC Priority Select Register (INTC_PSR77)	16	R/W	8000h	17.6.10/454
FC	INTC Priority Select Register (INTC_PSR78)	16	R/W	8000h	17.6.10/454
FE	INTC Priority Select Register (INTC_PSR79)	16	R/W	8000h	17.6.10/454
100	INTC Priority Select Register (INTC_PSR80)	16	R/W	8000h	17.6.10/454
102	INTC Priority Select Register (INTC_PSR81)	16	R/W	8000h	17.6.10/454
104	INTC Priority Select Register (INTC_PSR82)	16	R/W	8000h	17.6.10/454
106	INTC Priority Select Register (INTC_PSR83)	16	R/W	8000h	17.6.10/454
108	INTC Priority Select Register (INTC_PSR84)	16	R/W	8000h	17.6.10/454
10A	INTC Priority Select Register (INTC_PSR85)	16	R/W	8000h	17.6.10/454
10C	INTC Priority Select Register (INTC_PSR86)	16	R/W	8000h	17.6.10/454
10E	INTC Priority Select Register (INTC_PSR87)	16	R/W	8000h	17.6.10/454
110	INTC Priority Select Register (INTC_PSR88)	16	R/W	8000h	17.6.10/454
112	INTC Priority Select Register (INTC_PSR89)	16	R/W	8000h	17.6.10/454
114	INTC Priority Select Register (INTC_PSR90)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
116	INTC Priority Select Register (INTC_PSR91)	16	R/W	8000h	17.6.10/454
118	INTC Priority Select Register (INTC_PSR92)	16	R/W	8000h	17.6.10/454
11A	INTC Priority Select Register (INTC_PSR93)	16	R/W	8000h	17.6.10/454
11C	INTC Priority Select Register (INTC_PSR94)	16	R/W	8000h	17.6.10/454
11E	INTC Priority Select Register (INTC_PSR95)	16	R/W	8000h	17.6.10/454
120	INTC Priority Select Register (INTC_PSR96)	16	R/W	8000h	17.6.10/454
122	INTC Priority Select Register (INTC_PSR97)	16	R/W	8000h	17.6.10/454
124	INTC Priority Select Register (INTC_PSR98)	16	R/W	8000h	17.6.10/454
126	INTC Priority Select Register (INTC_PSR99)	16	R/W	8000h	17.6.10/454
128	INTC Priority Select Register (INTC_PSR100)	16	R/W	8000h	17.6.10/454
12A	INTC Priority Select Register (INTC_PSR101)	16	R/W	8000h	17.6.10/454
12C	INTC Priority Select Register (INTC_PSR102)	16	R/W	8000h	17.6.10/454
12E	INTC Priority Select Register (INTC_PSR103)	16	R/W	8000h	17.6.10/454
130	INTC Priority Select Register (INTC_PSR104)	16	R/W	8000h	17.6.10/454
132	INTC Priority Select Register (INTC_PSR105)	16	R/W	8000h	17.6.10/454
134	INTC Priority Select Register (INTC_PSR106)	16	R/W	8000h	17.6.10/454
136	INTC Priority Select Register (INTC_PSR107)	16	R/W	8000h	17.6.10/454
138	INTC Priority Select Register (INTC_PSR108)	16	R/W	8000h	17.6.10/454
13A	INTC Priority Select Register (INTC_PSR109)	16	R/W	8000h	17.6.10/454
13C	INTC Priority Select Register (INTC_PSR110)	16	R/W	8000h	17.6.10/454
13E	INTC Priority Select Register (INTC_PSR111)	16	R/W	8000h	17.6.10/454
140	INTC Priority Select Register (INTC_PSR112)	16	R/W	8000h	17.6.10/454
142	INTC Priority Select Register (INTC_PSR113)	16	R/W	8000h	17.6.10/454
144	INTC Priority Select Register (INTC_PSR114)	16	R/W	8000h	17.6.10/454
146	INTC Priority Select Register (INTC_PSR115)	16	R/W	8000h	17.6.10/454
148	INTC Priority Select Register (INTC_PSR116)	16	R/W	8000h	17.6.10/454
14A	INTC Priority Select Register (INTC_PSR117)	16	R/W	8000h	17.6.10/454
14C	INTC Priority Select Register (INTC_PSR118)	16	R/W	8000h	17.6.10/454
14E	INTC Priority Select Register (INTC_PSR119)	16	R/W	8000h	17.6.10/454
150	INTC Priority Select Register (INTC_PSR120)	16	R/W	8000h	17.6.10/454
152	INTC Priority Select Register (INTC_PSR121)	16	R/W	8000h	17.6.10/454
154	INTC Priority Select Register (INTC_PSR122)	16	R/W	8000h	17.6.10/454
156	INTC Priority Select Register (INTC_PSR123)	16	R/W	8000h	17.6.10/454
158	INTC Priority Select Register (INTC_PSR124)	16	R/W	8000h	17.6.10/454
15A	INTC Priority Select Register (INTC_PSR125)	16	R/W	8000h	17.6.10/454
15C	INTC Priority Select Register (INTC_PSR126)	16	R/W	8000h	17.6.10/454
15E	INTC Priority Select Register (INTC_PSR127)	16	R/W	8000h	17.6.10/454
160	INTC Priority Select Register (INTC_PSR128)	16	R/W	8000h	17.6.10/454
162	INTC Priority Select Register (INTC_PSR129)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
164	INTC Priority Select Register (INTC_PSR130)	16	R/W	8000h	17.6.10/454
166	INTC Priority Select Register (INTC_PSR131)	16	R/W	8000h	17.6.10/454
168	INTC Priority Select Register (INTC_PSR132)	16	R/W	8000h	17.6.10/454
16A	INTC Priority Select Register (INTC_PSR133)	16	R/W	8000h	17.6.10/454
16C	INTC Priority Select Register (INTC_PSR134)	16	R/W	8000h	17.6.10/454
16E	INTC Priority Select Register (INTC_PSR135)	16	R/W	8000h	17.6.10/454
170	INTC Priority Select Register (INTC_PSR136)	16	R/W	8000h	17.6.10/454
172	INTC Priority Select Register (INTC_PSR137)	16	R/W	8000h	17.6.10/454
174	INTC Priority Select Register (INTC_PSR138)	16	R/W	8000h	17.6.10/454
176	INTC Priority Select Register (INTC_PSR139)	16	R/W	8000h	17.6.10/454
178	INTC Priority Select Register (INTC_PSR140)	16	R/W	8000h	17.6.10/454
17A	INTC Priority Select Register (INTC_PSR141)	16	R/W	8000h	17.6.10/454
17C	INTC Priority Select Register (INTC_PSR142)	16	R/W	8000h	17.6.10/454
17E	INTC Priority Select Register (INTC_PSR143)	16	R/W	8000h	17.6.10/454
180	INTC Priority Select Register (INTC_PSR144)	16	R/W	8000h	17.6.10/454
182	INTC Priority Select Register (INTC_PSR145)	16	R/W	8000h	17.6.10/454
184	INTC Priority Select Register (INTC_PSR146)	16	R/W	8000h	17.6.10/454
186	INTC Priority Select Register (INTC_PSR147)	16	R/W	8000h	17.6.10/454
188	INTC Priority Select Register (INTC_PSR148)	16	R/W	8000h	17.6.10/454
18A	INTC Priority Select Register (INTC_PSR149)	16	R/W	8000h	17.6.10/454
18C	INTC Priority Select Register (INTC_PSR150)	16	R/W	8000h	17.6.10/454
18E	INTC Priority Select Register (INTC_PSR151)	16	R/W	8000h	17.6.10/454
190	INTC Priority Select Register (INTC_PSR152)	16	R/W	8000h	17.6.10/454
192	INTC Priority Select Register (INTC_PSR153)	16	R/W	8000h	17.6.10/454
194	INTC Priority Select Register (INTC_PSR154)	16	R/W	8000h	17.6.10/454
196	INTC Priority Select Register (INTC_PSR155)	16	R/W	8000h	17.6.10/454
198	INTC Priority Select Register (INTC_PSR156)	16	R/W	8000h	17.6.10/454
19A	INTC Priority Select Register (INTC_PSR157)	16	R/W	8000h	17.6.10/454
19C	INTC Priority Select Register (INTC_PSR158)	16	R/W	8000h	17.6.10/454
19E	INTC Priority Select Register (INTC_PSR159)	16	R/W	8000h	17.6.10/454
1A0	INTC Priority Select Register (INTC_PSR160)	16	R/W	8000h	17.6.10/454
1A2	INTC Priority Select Register (INTC_PSR161)	16	R/W	8000h	17.6.10/454
1A4	INTC Priority Select Register (INTC_PSR162)	16	R/W	8000h	17.6.10/454
1A6	INTC Priority Select Register (INTC_PSR163)	16	R/W	8000h	17.6.10/454
1A8	INTC Priority Select Register (INTC_PSR164)	16	R/W	8000h	17.6.10/454
1AA	INTC Priority Select Register (INTC_PSR165)	16	R/W	8000h	17.6.10/454
1AC	INTC Priority Select Register (INTC_PSR166)	16	R/W	8000h	17.6.10/454
1AE	INTC Priority Select Register (INTC_PSR167)	16	R/W	8000h	17.6.10/454
1B0	INTC Priority Select Register (INTC_PSR168)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1B2	INTC Priority Select Register (INTC_PSR169)	16	R/W	8000h	17.6.10/454
1B4	INTC Priority Select Register (INTC_PSR170)	16	R/W	8000h	17.6.10/454
1B6	INTC Priority Select Register (INTC_PSR171)	16	R/W	8000h	17.6.10/454
1B8	INTC Priority Select Register (INTC_PSR172)	16	R/W	8000h	17.6.10/454
1BA	INTC Priority Select Register (INTC_PSR173)	16	R/W	8000h	17.6.10/454
1BC	INTC Priority Select Register (INTC_PSR174)	16	R/W	8000h	17.6.10/454
1BE	INTC Priority Select Register (INTC_PSR175)	16	R/W	8000h	17.6.10/454
1C0	INTC Priority Select Register (INTC_PSR176)	16	R/W	8000h	17.6.10/454
1C2	INTC Priority Select Register (INTC_PSR177)	16	R/W	8000h	17.6.10/454
1C4	INTC Priority Select Register (INTC_PSR178)	16	R/W	8000h	17.6.10/454
1C6	INTC Priority Select Register (INTC_PSR179)	16	R/W	8000h	17.6.10/454
1C8	INTC Priority Select Register (INTC_PSR180)	16	R/W	8000h	17.6.10/454
1CA	INTC Priority Select Register (INTC_PSR181)	16	R/W	8000h	17.6.10/454
1CC	INTC Priority Select Register (INTC_PSR182)	16	R/W	8000h	17.6.10/454
1CE	INTC Priority Select Register (INTC_PSR183)	16	R/W	8000h	17.6.10/454
1D0	INTC Priority Select Register (INTC_PSR184)	16	R/W	8000h	17.6.10/454
1D2	INTC Priority Select Register (INTC_PSR185)	16	R/W	8000h	17.6.10/454
1D4	INTC Priority Select Register (INTC_PSR186)	16	R/W	8000h	17.6.10/454
1D6	INTC Priority Select Register (INTC_PSR187)	16	R/W	8000h	17.6.10/454
1D8	INTC Priority Select Register (INTC_PSR188)	16	R/W	8000h	17.6.10/454
1DA	INTC Priority Select Register (INTC_PSR189)	16	R/W	8000h	17.6.10/454
1DC	INTC Priority Select Register (INTC_PSR190)	16	R/W	8000h	17.6.10/454
1DE	INTC Priority Select Register (INTC_PSR191)	16	R/W	8000h	17.6.10/454
1E0	INTC Priority Select Register (INTC_PSR192)	16	R/W	8000h	17.6.10/454
1E2	INTC Priority Select Register (INTC_PSR193)	16	R/W	8000h	17.6.10/454
1E4	INTC Priority Select Register (INTC_PSR194)	16	R/W	8000h	17.6.10/454
1E6	INTC Priority Select Register (INTC_PSR195)	16	R/W	8000h	17.6.10/454
1E8	INTC Priority Select Register (INTC_PSR196)	16	R/W	8000h	17.6.10/454
1EA	INTC Priority Select Register (INTC_PSR197)	16	R/W	8000h	17.6.10/454
1EC	INTC Priority Select Register (INTC_PSR198)	16	R/W	8000h	17.6.10/454
1EE	INTC Priority Select Register (INTC_PSR199)	16	R/W	8000h	17.6.10/454
1F0	INTC Priority Select Register (INTC_PSR200)	16	R/W	8000h	17.6.10/454
1F2	INTC Priority Select Register (INTC_PSR201)	16	R/W	8000h	17.6.10/454
1F4	INTC Priority Select Register (INTC_PSR202)	16	R/W	8000h	17.6.10/454
1F6	INTC Priority Select Register (INTC_PSR203)	16	R/W	8000h	17.6.10/454
1F8	INTC Priority Select Register (INTC_PSR204)	16	R/W	8000h	17.6.10/454
1FA	INTC Priority Select Register (INTC_PSR205)	16	R/W	8000h	17.6.10/454
1FC	INTC Priority Select Register (INTC_PSR206)	16	R/W	8000h	17.6.10/454
1FE	INTC Priority Select Register (INTC_PSR207)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
200	INTC Priority Select Register (INTC_PSR208)	16	R/W	8000h	17.6.10/454
202	INTC Priority Select Register (INTC_PSR209)	16	R/W	8000h	17.6.10/454
204	INTC Priority Select Register (INTC_PSR210)	16	R/W	8000h	17.6.10/454
206	INTC Priority Select Register (INTC_PSR211)	16	R/W	8000h	17.6.10/454
208	INTC Priority Select Register (INTC_PSR212)	16	R/W	8000h	17.6.10/454
20A	INTC Priority Select Register (INTC_PSR213)	16	R/W	8000h	17.6.10/454
20C	INTC Priority Select Register (INTC_PSR214)	16	R/W	8000h	17.6.10/454
20E	INTC Priority Select Register (INTC_PSR215)	16	R/W	8000h	17.6.10/454
210	INTC Priority Select Register (INTC_PSR216)	16	R/W	8000h	17.6.10/454
212	INTC Priority Select Register (INTC_PSR217)	16	R/W	8000h	17.6.10/454
214	INTC Priority Select Register (INTC_PSR218)	16	R/W	8000h	17.6.10/454
216	INTC Priority Select Register (INTC_PSR219)	16	R/W	8000h	17.6.10/454
218	INTC Priority Select Register (INTC_PSR220)	16	R/W	8000h	17.6.10/454
21A	INTC Priority Select Register (INTC_PSR221)	16	R/W	8000h	17.6.10/454
21C	INTC Priority Select Register (INTC_PSR222)	16	R/W	8000h	17.6.10/454
21E	INTC Priority Select Register (INTC_PSR223)	16	R/W	8000h	17.6.10/454
220	INTC Priority Select Register (INTC_PSR224)	16	R/W	8000h	17.6.10/454
222	INTC Priority Select Register (INTC_PSR225)	16	R/W	8000h	17.6.10/454
224	INTC Priority Select Register (INTC_PSR226)	16	R/W	8000h	17.6.10/454
226	INTC Priority Select Register (INTC_PSR227)	16	R/W	8000h	17.6.10/454
228	INTC Priority Select Register (INTC_PSR228)	16	R/W	8000h	17.6.10/454
22A	INTC Priority Select Register (INTC_PSR229)	16	R/W	8000h	17.6.10/454
22C	INTC Priority Select Register (INTC_PSR230)	16	R/W	8000h	17.6.10/454
22E	INTC Priority Select Register (INTC_PSR231)	16	R/W	8000h	17.6.10/454
230	INTC Priority Select Register (INTC_PSR232)	16	R/W	8000h	17.6.10/454
232	INTC Priority Select Register (INTC_PSR233)	16	R/W	8000h	17.6.10/454
234	INTC Priority Select Register (INTC_PSR234)	16	R/W	8000h	17.6.10/454
236	INTC Priority Select Register (INTC_PSR235)	16	R/W	8000h	17.6.10/454
238	INTC Priority Select Register (INTC_PSR236)	16	R/W	8000h	17.6.10/454
23A	INTC Priority Select Register (INTC_PSR237)	16	R/W	8000h	17.6.10/454
23C	INTC Priority Select Register (INTC_PSR238)	16	R/W	8000h	17.6.10/454
23E	INTC Priority Select Register (INTC_PSR239)	16	R/W	8000h	17.6.10/454
240	INTC Priority Select Register (INTC_PSR240)	16	R/W	8000h	17.6.10/454
242	INTC Priority Select Register (INTC_PSR241)	16	R/W	8000h	17.6.10/454
244	INTC Priority Select Register (INTC_PSR242)	16	R/W	8000h	17.6.10/454
246	INTC Priority Select Register (INTC_PSR243)	16	R/W	8000h	17.6.10/454
248	INTC Priority Select Register (INTC_PSR244)	16	R/W	8000h	17.6.10/454
24A	INTC Priority Select Register (INTC_PSR245)	16	R/W	8000h	17.6.10/454
24C	INTC Priority Select Register (INTC_PSR246)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
24E	INTC Priority Select Register (INTC_PSR247)	16	R/W	8000h	17.6.10/454
250	INTC Priority Select Register (INTC_PSR248)	16	R/W	8000h	17.6.10/454
252	INTC Priority Select Register (INTC_PSR249)	16	R/W	8000h	17.6.10/454
254	INTC Priority Select Register (INTC_PSR250)	16	R/W	8000h	17.6.10/454
256	INTC Priority Select Register (INTC_PSR251)	16	R/W	8000h	17.6.10/454
258	INTC Priority Select Register (INTC_PSR252)	16	R/W	8000h	17.6.10/454
25A	INTC Priority Select Register (INTC_PSR253)	16	R/W	8000h	17.6.10/454
25C	INTC Priority Select Register (INTC_PSR254)	16	R/W	8000h	17.6.10/454
25E	INTC Priority Select Register (INTC_PSR255)	16	R/W	8000h	17.6.10/454
260	INTC Priority Select Register (INTC_PSR256)	16	R/W	8000h	17.6.10/454
262	INTC Priority Select Register (INTC_PSR257)	16	R/W	8000h	17.6.10/454
264	INTC Priority Select Register (INTC_PSR258)	16	R/W	8000h	17.6.10/454
266	INTC Priority Select Register (INTC_PSR259)	16	R/W	8000h	17.6.10/454
268	INTC Priority Select Register (INTC_PSR260)	16	R/W	8000h	17.6.10/454
26A	INTC Priority Select Register (INTC_PSR261)	16	R/W	8000h	17.6.10/454
26C	INTC Priority Select Register (INTC_PSR262)	16	R/W	8000h	17.6.10/454
26E	INTC Priority Select Register (INTC_PSR263)	16	R/W	8000h	17.6.10/454
270	INTC Priority Select Register (INTC_PSR264)	16	R/W	8000h	17.6.10/454
272	INTC Priority Select Register (INTC_PSR265)	16	R/W	8000h	17.6.10/454
274	INTC Priority Select Register (INTC_PSR266)	16	R/W	8000h	17.6.10/454
276	INTC Priority Select Register (INTC_PSR267)	16	R/W	8000h	17.6.10/454
278	INTC Priority Select Register (INTC_PSR268)	16	R/W	8000h	17.6.10/454
27A	INTC Priority Select Register (INTC_PSR269)	16	R/W	8000h	17.6.10/454
27C	INTC Priority Select Register (INTC_PSR270)	16	R/W	8000h	17.6.10/454
27E	INTC Priority Select Register (INTC_PSR271)	16	R/W	8000h	17.6.10/454
280	INTC Priority Select Register (INTC_PSR272)	16	R/W	8000h	17.6.10/454
282	INTC Priority Select Register (INTC_PSR273)	16	R/W	8000h	17.6.10/454
284	INTC Priority Select Register (INTC_PSR274)	16	R/W	8000h	17.6.10/454
286	INTC Priority Select Register (INTC_PSR275)	16	R/W	8000h	17.6.10/454
288	INTC Priority Select Register (INTC_PSR276)	16	R/W	8000h	17.6.10/454
28A	INTC Priority Select Register (INTC_PSR277)	16	R/W	8000h	17.6.10/454
28C	INTC Priority Select Register (INTC_PSR278)	16	R/W	8000h	17.6.10/454
28E	INTC Priority Select Register (INTC_PSR279)	16	R/W	8000h	17.6.10/454
290	INTC Priority Select Register (INTC_PSR280)	16	R/W	8000h	17.6.10/454
292	INTC Priority Select Register (INTC_PSR281)	16	R/W	8000h	17.6.10/454
294	INTC Priority Select Register (INTC_PSR282)	16	R/W	8000h	17.6.10/454
296	INTC Priority Select Register (INTC_PSR283)	16	R/W	8000h	17.6.10/454
298	INTC Priority Select Register (INTC_PSR284)	16	R/W	8000h	17.6.10/454
29A	INTC Priority Select Register (INTC_PSR285)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
29C	INTC Priority Select Register (INTC_PSR286)	16	R/W	8000h	17.6.10/454
29E	INTC Priority Select Register (INTC_PSR287)	16	R/W	8000h	17.6.10/454
2A0	INTC Priority Select Register (INTC_PSR288)	16	R/W	8000h	17.6.10/454
2A2	INTC Priority Select Register (INTC_PSR289)	16	R/W	8000h	17.6.10/454
2A4	INTC Priority Select Register (INTC_PSR290)	16	R/W	8000h	17.6.10/454
2A6	INTC Priority Select Register (INTC_PSR291)	16	R/W	8000h	17.6.10/454
2A8	INTC Priority Select Register (INTC_PSR292)	16	R/W	8000h	17.6.10/454
2AA	INTC Priority Select Register (INTC_PSR293)	16	R/W	8000h	17.6.10/454
2AC	INTC Priority Select Register (INTC_PSR294)	16	R/W	8000h	17.6.10/454
2AE	INTC Priority Select Register (INTC_PSR295)	16	R/W	8000h	17.6.10/454
2B0	INTC Priority Select Register (INTC_PSR296)	16	R/W	8000h	17.6.10/454
2B2	INTC Priority Select Register (INTC_PSR297)	16	R/W	8000h	17.6.10/454
2B4	INTC Priority Select Register (INTC_PSR298)	16	R/W	8000h	17.6.10/454
2B6	INTC Priority Select Register (INTC_PSR299)	16	R/W	8000h	17.6.10/454
2B8	INTC Priority Select Register (INTC_PSR300)	16	R/W	8000h	17.6.10/454
2BA	INTC Priority Select Register (INTC_PSR301)	16	R/W	8000h	17.6.10/454
2BC	INTC Priority Select Register (INTC_PSR302)	16	R/W	8000h	17.6.10/454
2BE	INTC Priority Select Register (INTC_PSR303)	16	R/W	8000h	17.6.10/454
2C0	INTC Priority Select Register (INTC_PSR304)	16	R/W	8000h	17.6.10/454
2C2	INTC Priority Select Register (INTC_PSR305)	16	R/W	8000h	17.6.10/454
2C4	INTC Priority Select Register (INTC_PSR306)	16	R/W	8000h	17.6.10/454
2C6	INTC Priority Select Register (INTC_PSR307)	16	R/W	8000h	17.6.10/454
2C8	INTC Priority Select Register (INTC_PSR308)	16	R/W	8000h	17.6.10/454
2CA	INTC Priority Select Register (INTC_PSR309)	16	R/W	8000h	17.6.10/454
2CC	INTC Priority Select Register (INTC_PSR310)	16	R/W	8000h	17.6.10/454
2CE	INTC Priority Select Register (INTC_PSR311)	16	R/W	8000h	17.6.10/454
2D0	INTC Priority Select Register (INTC_PSR312)	16	R/W	8000h	17.6.10/454
2D2	INTC Priority Select Register (INTC_PSR313)	16	R/W	8000h	17.6.10/454
2D4	INTC Priority Select Register (INTC_PSR314)	16	R/W	8000h	17.6.10/454
2D6	INTC Priority Select Register (INTC_PSR315)	16	R/W	8000h	17.6.10/454
2D8	INTC Priority Select Register (INTC_PSR316)	16	R/W	8000h	17.6.10/454
2DA	INTC Priority Select Register (INTC_PSR317)	16	R/W	8000h	17.6.10/454
2DC	INTC Priority Select Register (INTC_PSR318)	16	R/W	8000h	17.6.10/454
2DE	INTC Priority Select Register (INTC_PSR319)	16	R/W	8000h	17.6.10/454
2E0	INTC Priority Select Register (INTC_PSR320)	16	R/W	8000h	17.6.10/454
2E2	INTC Priority Select Register (INTC_PSR321)	16	R/W	8000h	17.6.10/454
2E4	INTC Priority Select Register (INTC_PSR322)	16	R/W	8000h	17.6.10/454
2E6	INTC Priority Select Register (INTC_PSR323)	16	R/W	8000h	17.6.10/454
2E8	INTC Priority Select Register (INTC_PSR324)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2EA	INTC Priority Select Register (INTC_PSR325)	16	R/W	8000h	17.6.10/454
2EC	INTC Priority Select Register (INTC_PSR326)	16	R/W	8000h	17.6.10/454
2EE	INTC Priority Select Register (INTC_PSR327)	16	R/W	8000h	17.6.10/454
2F0	INTC Priority Select Register (INTC_PSR328)	16	R/W	8000h	17.6.10/454
2F2	INTC Priority Select Register (INTC_PSR329)	16	R/W	8000h	17.6.10/454
2F4	INTC Priority Select Register (INTC_PSR330)	16	R/W	8000h	17.6.10/454
2F6	INTC Priority Select Register (INTC_PSR331)	16	R/W	8000h	17.6.10/454
2F8	INTC Priority Select Register (INTC_PSR332)	16	R/W	8000h	17.6.10/454
2FA	INTC Priority Select Register (INTC_PSR333)	16	R/W	8000h	17.6.10/454
2FC	INTC Priority Select Register (INTC_PSR334)	16	R/W	8000h	17.6.10/454
2FE	INTC Priority Select Register (INTC_PSR335)	16	R/W	8000h	17.6.10/454
300	INTC Priority Select Register (INTC_PSR336)	16	R/W	8000h	17.6.10/454
302	INTC Priority Select Register (INTC_PSR337)	16	R/W	8000h	17.6.10/454
304	INTC Priority Select Register (INTC_PSR338)	16	R/W	8000h	17.6.10/454
306	INTC Priority Select Register (INTC_PSR339)	16	R/W	8000h	17.6.10/454
308	INTC Priority Select Register (INTC_PSR340)	16	R/W	8000h	17.6.10/454
30A	INTC Priority Select Register (INTC_PSR341)	16	R/W	8000h	17.6.10/454
30C	INTC Priority Select Register (INTC_PSR342)	16	R/W	8000h	17.6.10/454
30E	INTC Priority Select Register (INTC_PSR343)	16	R/W	8000h	17.6.10/454
310	INTC Priority Select Register (INTC_PSR344)	16	R/W	8000h	17.6.10/454
312	INTC Priority Select Register (INTC_PSR345)	16	R/W	8000h	17.6.10/454
314	INTC Priority Select Register (INTC_PSR346)	16	R/W	8000h	17.6.10/454
316	INTC Priority Select Register (INTC_PSR347)	16	R/W	8000h	17.6.10/454
318	INTC Priority Select Register (INTC_PSR348)	16	R/W	8000h	17.6.10/454
31A	INTC Priority Select Register (INTC_PSR349)	16	R/W	8000h	17.6.10/454
31C	INTC Priority Select Register (INTC_PSR350)	16	R/W	8000h	17.6.10/454
31E	INTC Priority Select Register (INTC_PSR351)	16	R/W	8000h	17.6.10/454
320	INTC Priority Select Register (INTC_PSR352)	16	R/W	8000h	17.6.10/454
322	INTC Priority Select Register (INTC_PSR353)	16	R/W	8000h	17.6.10/454
324	INTC Priority Select Register (INTC_PSR354)	16	R/W	8000h	17.6.10/454
326	INTC Priority Select Register (INTC_PSR355)	16	R/W	8000h	17.6.10/454
328	INTC Priority Select Register (INTC_PSR356)	16	R/W	8000h	17.6.10/454
32A	INTC Priority Select Register (INTC_PSR357)	16	R/W	8000h	17.6.10/454
32C	INTC Priority Select Register (INTC_PSR358)	16	R/W	8000h	17.6.10/454
32E	INTC Priority Select Register (INTC_PSR359)	16	R/W	8000h	17.6.10/454
330	INTC Priority Select Register (INTC_PSR360)	16	R/W	8000h	17.6.10/454
332	INTC Priority Select Register (INTC_PSR361)	16	R/W	8000h	17.6.10/454
334	INTC Priority Select Register (INTC_PSR362)	16	R/W	8000h	17.6.10/454
336	INTC Priority Select Register (INTC_PSR363)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
338	INTC Priority Select Register (INTC_PSR364)	16	R/W	8000h	17.6.10/454
33A	INTC Priority Select Register (INTC_PSR365)	16	R/W	8000h	17.6.10/454
33C	INTC Priority Select Register (INTC_PSR366)	16	R/W	8000h	17.6.10/454
33E	INTC Priority Select Register (INTC_PSR367)	16	R/W	8000h	17.6.10/454
340	INTC Priority Select Register (INTC_PSR368)	16	R/W	8000h	17.6.10/454
342	INTC Priority Select Register (INTC_PSR369)	16	R/W	8000h	17.6.10/454
344	INTC Priority Select Register (INTC_PSR370)	16	R/W	8000h	17.6.10/454
346	INTC Priority Select Register (INTC_PSR371)	16	R/W	8000h	17.6.10/454
348	INTC Priority Select Register (INTC_PSR372)	16	R/W	8000h	17.6.10/454
34A	INTC Priority Select Register (INTC_PSR373)	16	R/W	8000h	17.6.10/454
34C	INTC Priority Select Register (INTC_PSR374)	16	R/W	8000h	17.6.10/454
34E	INTC Priority Select Register (INTC_PSR375)	16	R/W	8000h	17.6.10/454
350	INTC Priority Select Register (INTC_PSR376)	16	R/W	8000h	17.6.10/454
352	INTC Priority Select Register (INTC_PSR377)	16	R/W	8000h	17.6.10/454
354	INTC Priority Select Register (INTC_PSR378)	16	R/W	8000h	17.6.10/454
356	INTC Priority Select Register (INTC_PSR379)	16	R/W	8000h	17.6.10/454
358	INTC Priority Select Register (INTC_PSR380)	16	R/W	8000h	17.6.10/454
35A	INTC Priority Select Register (INTC_PSR381)	16	R/W	8000h	17.6.10/454
35C	INTC Priority Select Register (INTC_PSR382)	16	R/W	8000h	17.6.10/454
35E	INTC Priority Select Register (INTC_PSR383)	16	R/W	8000h	17.6.10/454
360	INTC Priority Select Register (INTC_PSR384)	16	R/W	8000h	17.6.10/454
362	INTC Priority Select Register (INTC_PSR385)	16	R/W	8000h	17.6.10/454
364	INTC Priority Select Register (INTC_PSR386)	16	R/W	8000h	17.6.10/454
366	INTC Priority Select Register (INTC_PSR387)	16	R/W	8000h	17.6.10/454
368	INTC Priority Select Register (INTC_PSR388)	16	R/W	8000h	17.6.10/454
36A	INTC Priority Select Register (INTC_PSR389)	16	R/W	8000h	17.6.10/454
36C	INTC Priority Select Register (INTC_PSR390)	16	R/W	8000h	17.6.10/454
36E	INTC Priority Select Register (INTC_PSR391)	16	R/W	8000h	17.6.10/454
370	INTC Priority Select Register (INTC_PSR392)	16	R/W	8000h	17.6.10/454
372	INTC Priority Select Register (INTC_PSR393)	16	R/W	8000h	17.6.10/454
374	INTC Priority Select Register (INTC_PSR394)	16	R/W	8000h	17.6.10/454
376	INTC Priority Select Register (INTC_PSR395)	16	R/W	8000h	17.6.10/454
378	INTC Priority Select Register (INTC_PSR396)	16	R/W	8000h	17.6.10/454
37A	INTC Priority Select Register (INTC_PSR397)	16	R/W	8000h	17.6.10/454
37C	INTC Priority Select Register (INTC_PSR398)	16	R/W	8000h	17.6.10/454
37E	INTC Priority Select Register (INTC_PSR399)	16	R/W	8000h	17.6.10/454
380	INTC Priority Select Register (INTC_PSR400)	16	R/W	8000h	17.6.10/454
382	INTC Priority Select Register (INTC_PSR401)	16	R/W	8000h	17.6.10/454
384	INTC Priority Select Register (INTC_PSR402)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
386	INTC Priority Select Register (INTC_PSR403)	16	R/W	8000h	17.6.10/454
388	INTC Priority Select Register (INTC_PSR404)	16	R/W	8000h	17.6.10/454
38A	INTC Priority Select Register (INTC_PSR405)	16	R/W	8000h	17.6.10/454
38C	INTC Priority Select Register (INTC_PSR406)	16	R/W	8000h	17.6.10/454
38E	INTC Priority Select Register (INTC_PSR407)	16	R/W	8000h	17.6.10/454
390	INTC Priority Select Register (INTC_PSR408)	16	R/W	8000h	17.6.10/454
392	INTC Priority Select Register (INTC_PSR409)	16	R/W	8000h	17.6.10/454
394	INTC Priority Select Register (INTC_PSR410)	16	R/W	8000h	17.6.10/454
396	INTC Priority Select Register (INTC_PSR411)	16	R/W	8000h	17.6.10/454
398	INTC Priority Select Register (INTC_PSR412)	16	R/W	8000h	17.6.10/454
39A	INTC Priority Select Register (INTC_PSR413)	16	R/W	8000h	17.6.10/454
39C	INTC Priority Select Register (INTC_PSR414)	16	R/W	8000h	17.6.10/454
39E	INTC Priority Select Register (INTC_PSR415)	16	R/W	8000h	17.6.10/454
3A0	INTC Priority Select Register (INTC_PSR416)	16	R/W	8000h	17.6.10/454
3A2	INTC Priority Select Register (INTC_PSR417)	16	R/W	8000h	17.6.10/454
3A4	INTC Priority Select Register (INTC_PSR418)	16	R/W	8000h	17.6.10/454
3A6	INTC Priority Select Register (INTC_PSR419)	16	R/W	8000h	17.6.10/454
3A8	INTC Priority Select Register (INTC_PSR420)	16	R/W	8000h	17.6.10/454
3AA	INTC Priority Select Register (INTC_PSR421)	16	R/W	8000h	17.6.10/454
3AC	INTC Priority Select Register (INTC_PSR422)	16	R/W	8000h	17.6.10/454
3AE	INTC Priority Select Register (INTC_PSR423)	16	R/W	8000h	17.6.10/454
3B0	INTC Priority Select Register (INTC_PSR424)	16	R/W	8000h	17.6.10/454
3B2	INTC Priority Select Register (INTC_PSR425)	16	R/W	8000h	17.6.10/454
3B4	INTC Priority Select Register (INTC_PSR426)	16	R/W	8000h	17.6.10/454
3B6	INTC Priority Select Register (INTC_PSR427)	16	R/W	8000h	17.6.10/454
3B8	INTC Priority Select Register (INTC_PSR428)	16	R/W	8000h	17.6.10/454
3BA	INTC Priority Select Register (INTC_PSR429)	16	R/W	8000h	17.6.10/454
3BC	INTC Priority Select Register (INTC_PSR430)	16	R/W	8000h	17.6.10/454
3BE	INTC Priority Select Register (INTC_PSR431)	16	R/W	8000h	17.6.10/454
3C0	INTC Priority Select Register (INTC_PSR432)	16	R/W	8000h	17.6.10/454
3C2	INTC Priority Select Register (INTC_PSR433)	16	R/W	8000h	17.6.10/454
3C4	INTC Priority Select Register (INTC_PSR434)	16	R/W	8000h	17.6.10/454
3C6	INTC Priority Select Register (INTC_PSR435)	16	R/W	8000h	17.6.10/454
3C8	INTC Priority Select Register (INTC_PSR436)	16	R/W	8000h	17.6.10/454
3CA	INTC Priority Select Register (INTC_PSR437)	16	R/W	8000h	17.6.10/454
3CC	INTC Priority Select Register (INTC_PSR438)	16	R/W	8000h	17.6.10/454
3CE	INTC Priority Select Register (INTC_PSR439)	16	R/W	8000h	17.6.10/454
3D0	INTC Priority Select Register (INTC_PSR440)	16	R/W	8000h	17.6.10/454
3D2	INTC Priority Select Register (INTC_PSR441)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3D4	INTC Priority Select Register (INTC_PSR442)	16	R/W	8000h	17.6.10/454
3D6	INTC Priority Select Register (INTC_PSR443)	16	R/W	8000h	17.6.10/454
3D8	INTC Priority Select Register (INTC_PSR444)	16	R/W	8000h	17.6.10/454
3DA	INTC Priority Select Register (INTC_PSR445)	16	R/W	8000h	17.6.10/454
3DC	INTC Priority Select Register (INTC_PSR446)	16	R/W	8000h	17.6.10/454
3DE	INTC Priority Select Register (INTC_PSR447)	16	R/W	8000h	17.6.10/454
3E0	INTC Priority Select Register (INTC_PSR448)	16	R/W	8000h	17.6.10/454
3E2	INTC Priority Select Register (INTC_PSR449)	16	R/W	8000h	17.6.10/454
3E4	INTC Priority Select Register (INTC_PSR450)	16	R/W	8000h	17.6.10/454
3E6	INTC Priority Select Register (INTC_PSR451)	16	R/W	8000h	17.6.10/454
3E8	INTC Priority Select Register (INTC_PSR452)	16	R/W	8000h	17.6.10/454
3EA	INTC Priority Select Register (INTC_PSR453)	16	R/W	8000h	17.6.10/454
3EC	INTC Priority Select Register (INTC_PSR454)	16	R/W	8000h	17.6.10/454
3EE	INTC Priority Select Register (INTC_PSR455)	16	R/W	8000h	17.6.10/454
3F0	INTC Priority Select Register (INTC_PSR456)	16	R/W	8000h	17.6.10/454
3F2	INTC Priority Select Register (INTC_PSR457)	16	R/W	8000h	17.6.10/454
3F4	INTC Priority Select Register (INTC_PSR458)	16	R/W	8000h	17.6.10/454
3F6	INTC Priority Select Register (INTC_PSR459)	16	R/W	8000h	17.6.10/454
3F8	INTC Priority Select Register (INTC_PSR460)	16	R/W	8000h	17.6.10/454
3FA	INTC Priority Select Register (INTC_PSR461)	16	R/W	8000h	17.6.10/454
3FC	INTC Priority Select Register (INTC_PSR462)	16	R/W	8000h	17.6.10/454
3FE	INTC Priority Select Register (INTC_PSR463)	16	R/W	8000h	17.6.10/454
400	INTC Priority Select Register (INTC_PSR464)	16	R/W	8000h	17.6.10/454
402	INTC Priority Select Register (INTC_PSR465)	16	R/W	8000h	17.6.10/454
404	INTC Priority Select Register (INTC_PSR466)	16	R/W	8000h	17.6.10/454
406	INTC Priority Select Register (INTC_PSR467)	16	R/W	8000h	17.6.10/454
408	INTC Priority Select Register (INTC_PSR468)	16	R/W	8000h	17.6.10/454
40A	INTC Priority Select Register (INTC_PSR469)	16	R/W	8000h	17.6.10/454
40C	INTC Priority Select Register (INTC_PSR470)	16	R/W	8000h	17.6.10/454
40E	INTC Priority Select Register (INTC_PSR471)	16	R/W	8000h	17.6.10/454
410	INTC Priority Select Register (INTC_PSR472)	16	R/W	8000h	17.6.10/454
412	INTC Priority Select Register (INTC_PSR473)	16	R/W	8000h	17.6.10/454
414	INTC Priority Select Register (INTC_PSR474)	16	R/W	8000h	17.6.10/454
416	INTC Priority Select Register (INTC_PSR475)	16	R/W	8000h	17.6.10/454
418	INTC Priority Select Register (INTC_PSR476)	16	R/W	8000h	17.6.10/454
41A	INTC Priority Select Register (INTC_PSR477)	16	R/W	8000h	17.6.10/454
41C	INTC Priority Select Register (INTC_PSR478)	16	R/W	8000h	17.6.10/454
41E	INTC Priority Select Register (INTC_PSR479)	16	R/W	8000h	17.6.10/454
420	INTC Priority Select Register (INTC_PSR480)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
422	INTC Priority Select Register (INTC_PSR481)	16	R/W	8000h	17.6.10/454
424	INTC Priority Select Register (INTC_PSR482)	16	R/W	8000h	17.6.10/454
426	INTC Priority Select Register (INTC_PSR483)	16	R/W	8000h	17.6.10/454
428	INTC Priority Select Register (INTC_PSR484)	16	R/W	8000h	17.6.10/454
42A	INTC Priority Select Register (INTC_PSR485)	16	R/W	8000h	17.6.10/454
42C	INTC Priority Select Register (INTC_PSR486)	16	R/W	8000h	17.6.10/454
42E	INTC Priority Select Register (INTC_PSR487)	16	R/W	8000h	17.6.10/454
430	INTC Priority Select Register (INTC_PSR488)	16	R/W	8000h	17.6.10/454
432	INTC Priority Select Register (INTC_PSR489)	16	R/W	8000h	17.6.10/454
434	INTC Priority Select Register (INTC_PSR490)	16	R/W	8000h	17.6.10/454
436	INTC Priority Select Register (INTC_PSR491)	16	R/W	8000h	17.6.10/454
438	INTC Priority Select Register (INTC_PSR492)	16	R/W	8000h	17.6.10/454
43A	INTC Priority Select Register (INTC_PSR493)	16	R/W	8000h	17.6.10/454
43C	INTC Priority Select Register (INTC_PSR494)	16	R/W	8000h	17.6.10/454
43E	INTC Priority Select Register (INTC_PSR495)	16	R/W	8000h	17.6.10/454
440	INTC Priority Select Register (INTC_PSR496)	16	R/W	8000h	17.6.10/454
442	INTC Priority Select Register (INTC_PSR497)	16	R/W	8000h	17.6.10/454
444	INTC Priority Select Register (INTC_PSR498)	16	R/W	8000h	17.6.10/454
446	INTC Priority Select Register (INTC_PSR499)	16	R/W	8000h	17.6.10/454
448	INTC Priority Select Register (INTC_PSR500)	16	R/W	8000h	17.6.10/454
44A	INTC Priority Select Register (INTC_PSR501)	16	R/W	8000h	17.6.10/454
44C	INTC Priority Select Register (INTC_PSR502)	16	R/W	8000h	17.6.10/454
44E	INTC Priority Select Register (INTC_PSR503)	16	R/W	8000h	17.6.10/454
450	INTC Priority Select Register (INTC_PSR504)	16	R/W	8000h	17.6.10/454
452	INTC Priority Select Register (INTC_PSR505)	16	R/W	8000h	17.6.10/454
454	INTC Priority Select Register (INTC_PSR506)	16	R/W	8000h	17.6.10/454
456	INTC Priority Select Register (INTC_PSR507)	16	R/W	8000h	17.6.10/454
458	INTC Priority Select Register (INTC_PSR508)	16	R/W	8000h	17.6.10/454
45A	INTC Priority Select Register (INTC_PSR509)	16	R/W	8000h	17.6.10/454
45C	INTC Priority Select Register (INTC_PSR510)	16	R/W	8000h	17.6.10/454
45E	INTC Priority Select Register (INTC_PSR511)	16	R/W	8000h	17.6.10/454
460	INTC Priority Select Register (INTC_PSR512)	16	R/W	8000h	17.6.10/454
462	INTC Priority Select Register (INTC_PSR513)	16	R/W	8000h	17.6.10/454
464	INTC Priority Select Register (INTC_PSR514)	16	R/W	8000h	17.6.10/454
466	INTC Priority Select Register (INTC_PSR515)	16	R/W	8000h	17.6.10/454
468	INTC Priority Select Register (INTC_PSR516)	16	R/W	8000h	17.6.10/454
46A	INTC Priority Select Register (INTC_PSR517)	16	R/W	8000h	17.6.10/454
46C	INTC Priority Select Register (INTC_PSR518)	16	R/W	8000h	17.6.10/454
46E	INTC Priority Select Register (INTC_PSR519)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
470	INTC Priority Select Register (INTC_PSR520)	16	R/W	8000h	17.6.10/454
472	INTC Priority Select Register (INTC_PSR521)	16	R/W	8000h	17.6.10/454
474	INTC Priority Select Register (INTC_PSR522)	16	R/W	8000h	17.6.10/454
476	INTC Priority Select Register (INTC_PSR523)	16	R/W	8000h	17.6.10/454
478	INTC Priority Select Register (INTC_PSR524)	16	R/W	8000h	17.6.10/454
47A	INTC Priority Select Register (INTC_PSR525)	16	R/W	8000h	17.6.10/454
47C	INTC Priority Select Register (INTC_PSR526)	16	R/W	8000h	17.6.10/454
47E	INTC Priority Select Register (INTC_PSR527)	16	R/W	8000h	17.6.10/454
480	INTC Priority Select Register (INTC_PSR528)	16	R/W	8000h	17.6.10/454
482	INTC Priority Select Register (INTC_PSR529)	16	R/W	8000h	17.6.10/454
484	INTC Priority Select Register (INTC_PSR530)	16	R/W	8000h	17.6.10/454
486	INTC Priority Select Register (INTC_PSR531)	16	R/W	8000h	17.6.10/454
488	INTC Priority Select Register (INTC_PSR532)	16	R/W	8000h	17.6.10/454
48A	INTC Priority Select Register (INTC_PSR533)	16	R/W	8000h	17.6.10/454
48C	INTC Priority Select Register (INTC_PSR534)	16	R/W	8000h	17.6.10/454
48E	INTC Priority Select Register (INTC_PSR535)	16	R/W	8000h	17.6.10/454
490	INTC Priority Select Register (INTC_PSR536)	16	R/W	8000h	17.6.10/454
492	INTC Priority Select Register (INTC_PSR537)	16	R/W	8000h	17.6.10/454
494	INTC Priority Select Register (INTC_PSR538)	16	R/W	8000h	17.6.10/454
496	INTC Priority Select Register (INTC_PSR539)	16	R/W	8000h	17.6.10/454
498	INTC Priority Select Register (INTC_PSR540)	16	R/W	8000h	17.6.10/454
49A	INTC Priority Select Register (INTC_PSR541)	16	R/W	8000h	17.6.10/454
49C	INTC Priority Select Register (INTC_PSR542)	16	R/W	8000h	17.6.10/454
49E	INTC Priority Select Register (INTC_PSR543)	16	R/W	8000h	17.6.10/454
4A0	INTC Priority Select Register (INTC_PSR544)	16	R/W	8000h	17.6.10/454
4A2	INTC Priority Select Register (INTC_PSR545)	16	R/W	8000h	17.6.10/454
4A4	INTC Priority Select Register (INTC_PSR546)	16	R/W	8000h	17.6.10/454
4A6	INTC Priority Select Register (INTC_PSR547)	16	R/W	8000h	17.6.10/454
4A8	INTC Priority Select Register (INTC_PSR548)	16	R/W	8000h	17.6.10/454
4AA	INTC Priority Select Register (INTC_PSR549)	16	R/W	8000h	17.6.10/454
4AC	INTC Priority Select Register (INTC_PSR550)	16	R/W	8000h	17.6.10/454
4AE	INTC Priority Select Register (INTC_PSR551)	16	R/W	8000h	17.6.10/454
4B0	INTC Priority Select Register (INTC_PSR552)	16	R/W	8000h	17.6.10/454
4B2	INTC Priority Select Register (INTC_PSR553)	16	R/W	8000h	17.6.10/454
4B4	INTC Priority Select Register (INTC_PSR554)	16	R/W	8000h	17.6.10/454
4B6	INTC Priority Select Register (INTC_PSR555)	16	R/W	8000h	17.6.10/454
4B8	INTC Priority Select Register (INTC_PSR556)	16	R/W	8000h	17.6.10/454
4BA	INTC Priority Select Register (INTC_PSR557)	16	R/W	8000h	17.6.10/454
4BC	INTC Priority Select Register (INTC_PSR558)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4BE	INTC Priority Select Register (INTC_PSR559)	16	R/W	8000h	17.6.10/454
4C0	INTC Priority Select Register (INTC_PSR560)	16	R/W	8000h	17.6.10/454
4C2	INTC Priority Select Register (INTC_PSR561)	16	R/W	8000h	17.6.10/454
4C4	INTC Priority Select Register (INTC_PSR562)	16	R/W	8000h	17.6.10/454
4C6	INTC Priority Select Register (INTC_PSR563)	16	R/W	8000h	17.6.10/454
4C8	INTC Priority Select Register (INTC_PSR564)	16	R/W	8000h	17.6.10/454
4CA	INTC Priority Select Register (INTC_PSR565)	16	R/W	8000h	17.6.10/454
4CC	INTC Priority Select Register (INTC_PSR566)	16	R/W	8000h	17.6.10/454
4CE	INTC Priority Select Register (INTC_PSR567)	16	R/W	8000h	17.6.10/454
4D0	INTC Priority Select Register (INTC_PSR568)	16	R/W	8000h	17.6.10/454
4D2	INTC Priority Select Register (INTC_PSR569)	16	R/W	8000h	17.6.10/454
4D4	INTC Priority Select Register (INTC_PSR570)	16	R/W	8000h	17.6.10/454
4D6	INTC Priority Select Register (INTC_PSR571)	16	R/W	8000h	17.6.10/454
4D8	INTC Priority Select Register (INTC_PSR572)	16	R/W	8000h	17.6.10/454
4DA	INTC Priority Select Register (INTC_PSR573)	16	R/W	8000h	17.6.10/454
4DC	INTC Priority Select Register (INTC_PSR574)	16	R/W	8000h	17.6.10/454
4DE	INTC Priority Select Register (INTC_PSR575)	16	R/W	8000h	17.6.10/454
4E0	INTC Priority Select Register (INTC_PSR576)	16	R/W	8000h	17.6.10/454
4E2	INTC Priority Select Register (INTC_PSR577)	16	R/W	8000h	17.6.10/454
4E4	INTC Priority Select Register (INTC_PSR578)	16	R/W	8000h	17.6.10/454
4E6	INTC Priority Select Register (INTC_PSR579)	16	R/W	8000h	17.6.10/454
4E8	INTC Priority Select Register (INTC_PSR580)	16	R/W	8000h	17.6.10/454
4EA	INTC Priority Select Register (INTC_PSR581)	16	R/W	8000h	17.6.10/454
4EC	INTC Priority Select Register (INTC_PSR582)	16	R/W	8000h	17.6.10/454
4EE	INTC Priority Select Register (INTC_PSR583)	16	R/W	8000h	17.6.10/454
4F0	INTC Priority Select Register (INTC_PSR584)	16	R/W	8000h	17.6.10/454
4F2	INTC Priority Select Register (INTC_PSR585)	16	R/W	8000h	17.6.10/454
4F4	INTC Priority Select Register (INTC_PSR586)	16	R/W	8000h	17.6.10/454
4F6	INTC Priority Select Register (INTC_PSR587)	16	R/W	8000h	17.6.10/454
4F8	INTC Priority Select Register (INTC_PSR588)	16	R/W	8000h	17.6.10/454
4FA	INTC Priority Select Register (INTC_PSR589)	16	R/W	8000h	17.6.10/454
4FC	INTC Priority Select Register (INTC_PSR590)	16	R/W	8000h	17.6.10/454
4FE	INTC Priority Select Register (INTC_PSR591)	16	R/W	8000h	17.6.10/454
500	INTC Priority Select Register (INTC_PSR592)	16	R/W	8000h	17.6.10/454
502	INTC Priority Select Register (INTC_PSR593)	16	R/W	8000h	17.6.10/454
504	INTC Priority Select Register (INTC_PSR594)	16	R/W	8000h	17.6.10/454
506	INTC Priority Select Register (INTC_PSR595)	16	R/W	8000h	17.6.10/454
508	INTC Priority Select Register (INTC_PSR596)	16	R/W	8000h	17.6.10/454
50A	INTC Priority Select Register (INTC_PSR597)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
50C	INTC Priority Select Register (INTC_PSR598)	16	R/W	8000h	17.6.10/454
50E	INTC Priority Select Register (INTC_PSR599)	16	R/W	8000h	17.6.10/454
510	INTC Priority Select Register (INTC_PSR600)	16	R/W	8000h	17.6.10/454
512	INTC Priority Select Register (INTC_PSR601)	16	R/W	8000h	17.6.10/454
514	INTC Priority Select Register (INTC_PSR602)	16	R/W	8000h	17.6.10/454
516	INTC Priority Select Register (INTC_PSR603)	16	R/W	8000h	17.6.10/454
518	INTC Priority Select Register (INTC_PSR604)	16	R/W	8000h	17.6.10/454
51A	INTC Priority Select Register (INTC_PSR605)	16	R/W	8000h	17.6.10/454
51C	INTC Priority Select Register (INTC_PSR606)	16	R/W	8000h	17.6.10/454
51E	INTC Priority Select Register (INTC_PSR607)	16	R/W	8000h	17.6.10/454
520	INTC Priority Select Register (INTC_PSR608)	16	R/W	8000h	17.6.10/454
522	INTC Priority Select Register (INTC_PSR609)	16	R/W	8000h	17.6.10/454
524	INTC Priority Select Register (INTC_PSR610)	16	R/W	8000h	17.6.10/454
526	INTC Priority Select Register (INTC_PSR611)	16	R/W	8000h	17.6.10/454
528	INTC Priority Select Register (INTC_PSR612)	16	R/W	8000h	17.6.10/454
52A	INTC Priority Select Register (INTC_PSR613)	16	R/W	8000h	17.6.10/454
52C	INTC Priority Select Register (INTC_PSR614)	16	R/W	8000h	17.6.10/454
52E	INTC Priority Select Register (INTC_PSR615)	16	R/W	8000h	17.6.10/454
530	INTC Priority Select Register (INTC_PSR616)	16	R/W	8000h	17.6.10/454
532	INTC Priority Select Register (INTC_PSR617)	16	R/W	8000h	17.6.10/454
534	INTC Priority Select Register (INTC_PSR618)	16	R/W	8000h	17.6.10/454
536	INTC Priority Select Register (INTC_PSR619)	16	R/W	8000h	17.6.10/454
538	INTC Priority Select Register (INTC_PSR620)	16	R/W	8000h	17.6.10/454
53A	INTC Priority Select Register (INTC_PSR621)	16	R/W	8000h	17.6.10/454
53C	INTC Priority Select Register (INTC_PSR622)	16	R/W	8000h	17.6.10/454
53E	INTC Priority Select Register (INTC_PSR623)	16	R/W	8000h	17.6.10/454
540	INTC Priority Select Register (INTC_PSR624)	16	R/W	8000h	17.6.10/454
542	INTC Priority Select Register (INTC_PSR625)	16	R/W	8000h	17.6.10/454
544	INTC Priority Select Register (INTC_PSR626)	16	R/W	8000h	17.6.10/454
546	INTC Priority Select Register (INTC_PSR627)	16	R/W	8000h	17.6.10/454
548	INTC Priority Select Register (INTC_PSR628)	16	R/W	8000h	17.6.10/454
54A	INTC Priority Select Register (INTC_PSR629)	16	R/W	8000h	17.6.10/454
54C	INTC Priority Select Register (INTC_PSR630)	16	R/W	8000h	17.6.10/454
54E	INTC Priority Select Register (INTC_PSR631)	16	R/W	8000h	17.6.10/454
550	INTC Priority Select Register (INTC_PSR632)	16	R/W	8000h	17.6.10/454
552	INTC Priority Select Register (INTC_PSR633)	16	R/W	8000h	17.6.10/454
554	INTC Priority Select Register (INTC_PSR634)	16	R/W	8000h	17.6.10/454
556	INTC Priority Select Register (INTC_PSR635)	16	R/W	8000h	17.6.10/454
558	INTC Priority Select Register (INTC_PSR636)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
55A	INTC Priority Select Register (INTC_PSR637)	16	R/W	8000h	17.6.10/454
55C	INTC Priority Select Register (INTC_PSR638)	16	R/W	8000h	17.6.10/454
55E	INTC Priority Select Register (INTC_PSR639)	16	R/W	8000h	17.6.10/454
560	INTC Priority Select Register (INTC_PSR640)	16	R/W	8000h	17.6.10/454
562	INTC Priority Select Register (INTC_PSR641)	16	R/W	8000h	17.6.10/454
564	INTC Priority Select Register (INTC_PSR642)	16	R/W	8000h	17.6.10/454
566	INTC Priority Select Register (INTC_PSR643)	16	R/W	8000h	17.6.10/454
568	INTC Priority Select Register (INTC_PSR644)	16	R/W	8000h	17.6.10/454
56A	INTC Priority Select Register (INTC_PSR645)	16	R/W	8000h	17.6.10/454
56C	INTC Priority Select Register (INTC_PSR646)	16	R/W	8000h	17.6.10/454
56E	INTC Priority Select Register (INTC_PSR647)	16	R/W	8000h	17.6.10/454
570	INTC Priority Select Register (INTC_PSR648)	16	R/W	8000h	17.6.10/454
572	INTC Priority Select Register (INTC_PSR649)	16	R/W	8000h	17.6.10/454
574	INTC Priority Select Register (INTC_PSR650)	16	R/W	8000h	17.6.10/454
576	INTC Priority Select Register (INTC_PSR651)	16	R/W	8000h	17.6.10/454
578	INTC Priority Select Register (INTC_PSR652)	16	R/W	8000h	17.6.10/454
57A	INTC Priority Select Register (INTC_PSR653)	16	R/W	8000h	17.6.10/454
57C	INTC Priority Select Register (INTC_PSR654)	16	R/W	8000h	17.6.10/454
57E	INTC Priority Select Register (INTC_PSR655)	16	R/W	8000h	17.6.10/454
580	INTC Priority Select Register (INTC_PSR656)	16	R/W	8000h	17.6.10/454
582	INTC Priority Select Register (INTC_PSR657)	16	R/W	8000h	17.6.10/454
584	INTC Priority Select Register (INTC_PSR658)	16	R/W	8000h	17.6.10/454
586	INTC Priority Select Register (INTC_PSR659)	16	R/W	8000h	17.6.10/454
588	INTC Priority Select Register (INTC_PSR660)	16	R/W	8000h	17.6.10/454
58A	INTC Priority Select Register (INTC_PSR661)	16	R/W	8000h	17.6.10/454
58C	INTC Priority Select Register (INTC_PSR662)	16	R/W	8000h	17.6.10/454
58E	INTC Priority Select Register (INTC_PSR663)	16	R/W	8000h	17.6.10/454
590	INTC Priority Select Register (INTC_PSR664)	16	R/W	8000h	17.6.10/454
592	INTC Priority Select Register (INTC_PSR665)	16	R/W	8000h	17.6.10/454
594	INTC Priority Select Register (INTC_PSR666)	16	R/W	8000h	17.6.10/454
596	INTC Priority Select Register (INTC_PSR667)	16	R/W	8000h	17.6.10/454
598	INTC Priority Select Register (INTC_PSR668)	16	R/W	8000h	17.6.10/454
59A	INTC Priority Select Register (INTC_PSR669)	16	R/W	8000h	17.6.10/454
59C	INTC Priority Select Register (INTC_PSR670)	16	R/W	8000h	17.6.10/454
59E	INTC Priority Select Register (INTC_PSR671)	16	R/W	8000h	17.6.10/454
5A0	INTC Priority Select Register (INTC_PSR672)	16	R/W	8000h	17.6.10/454
5A2	INTC Priority Select Register (INTC_PSR673)	16	R/W	8000h	17.6.10/454
5A4	INTC Priority Select Register (INTC_PSR674)	16	R/W	8000h	17.6.10/454
5A6	INTC Priority Select Register (INTC_PSR675)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5A8	INTC Priority Select Register (INTC_PSR676)	16	R/W	8000h	17.6.10/454
5AA	INTC Priority Select Register (INTC_PSR677)	16	R/W	8000h	17.6.10/454
5AC	INTC Priority Select Register (INTC_PSR678)	16	R/W	8000h	17.6.10/454
5AE	INTC Priority Select Register (INTC_PSR679)	16	R/W	8000h	17.6.10/454
5B0	INTC Priority Select Register (INTC_PSR680)	16	R/W	8000h	17.6.10/454
5B2	INTC Priority Select Register (INTC_PSR681)	16	R/W	8000h	17.6.10/454
5B4	INTC Priority Select Register (INTC_PSR682)	16	R/W	8000h	17.6.10/454
5B6	INTC Priority Select Register (INTC_PSR683)	16	R/W	8000h	17.6.10/454
5B8	INTC Priority Select Register (INTC_PSR684)	16	R/W	8000h	17.6.10/454
5BA	INTC Priority Select Register (INTC_PSR685)	16	R/W	8000h	17.6.10/454
5BC	INTC Priority Select Register (INTC_PSR686)	16	R/W	8000h	17.6.10/454
5BE	INTC Priority Select Register (INTC_PSR687)	16	R/W	8000h	17.6.10/454
5C0	INTC Priority Select Register (INTC_PSR688)	16	R/W	8000h	17.6.10/454
5C2	INTC Priority Select Register (INTC_PSR689)	16	R/W	8000h	17.6.10/454
5C4	INTC Priority Select Register (INTC_PSR690)	16	R/W	8000h	17.6.10/454
5C6	INTC Priority Select Register (INTC_PSR691)	16	R/W	8000h	17.6.10/454
5C8	INTC Priority Select Register (INTC_PSR692)	16	R/W	8000h	17.6.10/454
5CA	INTC Priority Select Register (INTC_PSR693)	16	R/W	8000h	17.6.10/454
5CC	INTC Priority Select Register (INTC_PSR694)	16	R/W	8000h	17.6.10/454
5CE	INTC Priority Select Register (INTC_PSR695)	16	R/W	8000h	17.6.10/454
5D0	INTC Priority Select Register (INTC_PSR696)	16	R/W	8000h	17.6.10/454
5D2	INTC Priority Select Register (INTC_PSR697)	16	R/W	8000h	17.6.10/454
5D4	INTC Priority Select Register (INTC_PSR698)	16	R/W	8000h	17.6.10/454
5D6	INTC Priority Select Register (INTC_PSR699)	16	R/W	8000h	17.6.10/454
5D8	INTC Priority Select Register (INTC_PSR700)	16	R/W	8000h	17.6.10/454
5DA	INTC Priority Select Register (INTC_PSR701)	16	R/W	8000h	17.6.10/454
5DC	INTC Priority Select Register (INTC_PSR702)	16	R/W	8000h	17.6.10/454
5DE	INTC Priority Select Register (INTC_PSR703)	16	R/W	8000h	17.6.10/454
5E0	INTC Priority Select Register (INTC_PSR704)	16	R/W	8000h	17.6.10/454
5E2	INTC Priority Select Register (INTC_PSR705)	16	R/W	8000h	17.6.10/454
5E4	INTC Priority Select Register (INTC_PSR706)	16	R/W	8000h	17.6.10/454
5E6	INTC Priority Select Register (INTC_PSR707)	16	R/W	8000h	17.6.10/454
5E8	INTC Priority Select Register (INTC_PSR708)	16	R/W	8000h	17.6.10/454
5EA	INTC Priority Select Register (INTC_PSR709)	16	R/W	8000h	17.6.10/454
5EC	INTC Priority Select Register (INTC_PSR710)	16	R/W	8000h	17.6.10/454
5EE	INTC Priority Select Register (INTC_PSR711)	16	R/W	8000h	17.6.10/454
5F0	INTC Priority Select Register (INTC_PSR712)	16	R/W	8000h	17.6.10/454
5F2	INTC Priority Select Register (INTC_PSR713)	16	R/W	8000h	17.6.10/454
5F4	INTC Priority Select Register (INTC_PSR714)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5F6	INTC Priority Select Register (INTC_PSR715)	16	R/W	8000h	17.6.10/454
5F8	INTC Priority Select Register (INTC_PSR716)	16	R/W	8000h	17.6.10/454
5FA	INTC Priority Select Register (INTC_PSR717)	16	R/W	8000h	17.6.10/454
5FC	INTC Priority Select Register (INTC_PSR718)	16	R/W	8000h	17.6.10/454
5FE	INTC Priority Select Register (INTC_PSR719)	16	R/W	8000h	17.6.10/454
600	INTC Priority Select Register (INTC_PSR720)	16	R/W	8000h	17.6.10/454
602	INTC Priority Select Register (INTC_PSR721)	16	R/W	8000h	17.6.10/454
604	INTC Priority Select Register (INTC_PSR722)	16	R/W	8000h	17.6.10/454
606	INTC Priority Select Register (INTC_PSR723)	16	R/W	8000h	17.6.10/454
608	INTC Priority Select Register (INTC_PSR724)	16	R/W	8000h	17.6.10/454
60A	INTC Priority Select Register (INTC_PSR725)	16	R/W	8000h	17.6.10/454
60C	INTC Priority Select Register (INTC_PSR726)	16	R/W	8000h	17.6.10/454
60E	INTC Priority Select Register (INTC_PSR727)	16	R/W	8000h	17.6.10/454
610	INTC Priority Select Register (INTC_PSR728)	16	R/W	8000h	17.6.10/454
612	INTC Priority Select Register (INTC_PSR729)	16	R/W	8000h	17.6.10/454
614	INTC Priority Select Register (INTC_PSR730)	16	R/W	8000h	17.6.10/454
616	INTC Priority Select Register (INTC_PSR731)	16	R/W	8000h	17.6.10/454
618	INTC Priority Select Register (INTC_PSR732)	16	R/W	8000h	17.6.10/454
61A	INTC Priority Select Register (INTC_PSR733)	16	R/W	8000h	17.6.10/454
61C	INTC Priority Select Register (INTC_PSR734)	16	R/W	8000h	17.6.10/454
61E	INTC Priority Select Register (INTC_PSR735)	16	R/W	8000h	17.6.10/454
620	INTC Priority Select Register (INTC_PSR736)	16	R/W	8000h	17.6.10/454
622	INTC Priority Select Register (INTC_PSR737)	16	R/W	8000h	17.6.10/454
624	INTC Priority Select Register (INTC_PSR738)	16	R/W	8000h	17.6.10/454
626	INTC Priority Select Register (INTC_PSR739)	16	R/W	8000h	17.6.10/454
628	INTC Priority Select Register (INTC_PSR740)	16	R/W	8000h	17.6.10/454
62A	INTC Priority Select Register (INTC_PSR741)	16	R/W	8000h	17.6.10/454
62C	INTC Priority Select Register (INTC_PSR742)	16	R/W	8000h	17.6.10/454
62E	INTC Priority Select Register (INTC_PSR743)	16	R/W	8000h	17.6.10/454
630	INTC Priority Select Register (INTC_PSR744)	16	R/W	8000h	17.6.10/454
632	INTC Priority Select Register (INTC_PSR745)	16	R/W	8000h	17.6.10/454
634	INTC Priority Select Register (INTC_PSR746)	16	R/W	8000h	17.6.10/454
636	INTC Priority Select Register (INTC_PSR747)	16	R/W	8000h	17.6.10/454
638	INTC Priority Select Register (INTC_PSR748)	16	R/W	8000h	17.6.10/454
63A	INTC Priority Select Register (INTC_PSR749)	16	R/W	8000h	17.6.10/454
63C	INTC Priority Select Register (INTC_PSR750)	16	R/W	8000h	17.6.10/454
63E	INTC Priority Select Register (INTC_PSR751)	16	R/W	8000h	17.6.10/454
640	INTC Priority Select Register (INTC_PSR752)	16	R/W	8000h	17.6.10/454
642	INTC Priority Select Register (INTC_PSR753)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
644	INTC Priority Select Register (INTC_PSR754)	16	R/W	8000h	17.6.10/454
646	INTC Priority Select Register (INTC_PSR755)	16	R/W	8000h	17.6.10/454
648	INTC Priority Select Register (INTC_PSR756)	16	R/W	8000h	17.6.10/454
64A	INTC Priority Select Register (INTC_PSR757)	16	R/W	8000h	17.6.10/454
64C	INTC Priority Select Register (INTC_PSR758)	16	R/W	8000h	17.6.10/454
64E	INTC Priority Select Register (INTC_PSR759)	16	R/W	8000h	17.6.10/454
650	INTC Priority Select Register (INTC_PSR760)	16	R/W	8000h	17.6.10/454
652	INTC Priority Select Register (INTC_PSR761)	16	R/W	8000h	17.6.10/454
654	INTC Priority Select Register (INTC_PSR762)	16	R/W	8000h	17.6.10/454
656	INTC Priority Select Register (INTC_PSR763)	16	R/W	8000h	17.6.10/454
658	INTC Priority Select Register (INTC_PSR764)	16	R/W	8000h	17.6.10/454
65A	INTC Priority Select Register (INTC_PSR765)	16	R/W	8000h	17.6.10/454
65C	INTC Priority Select Register (INTC_PSR766)	16	R/W	8000h	17.6.10/454
65E	INTC Priority Select Register (INTC_PSR767)	16	R/W	8000h	17.6.10/454
660	INTC Priority Select Register (INTC_PSR768)	16	R/W	8000h	17.6.10/454
662	INTC Priority Select Register (INTC_PSR769)	16	R/W	8000h	17.6.10/454
664	INTC Priority Select Register (INTC_PSR770)	16	R/W	8000h	17.6.10/454
666	INTC Priority Select Register (INTC_PSR771)	16	R/W	8000h	17.6.10/454
668	INTC Priority Select Register (INTC_PSR772)	16	R/W	8000h	17.6.10/454
66A	INTC Priority Select Register (INTC_PSR773)	16	R/W	8000h	17.6.10/454
66C	INTC Priority Select Register (INTC_PSR774)	16	R/W	8000h	17.6.10/454
66E	INTC Priority Select Register (INTC_PSR775)	16	R/W	8000h	17.6.10/454
670	INTC Priority Select Register (INTC_PSR776)	16	R/W	8000h	17.6.10/454
672	INTC Priority Select Register (INTC_PSR777)	16	R/W	8000h	17.6.10/454
674	INTC Priority Select Register (INTC_PSR778)	16	R/W	8000h	17.6.10/454
676	INTC Priority Select Register (INTC_PSR779)	16	R/W	8000h	17.6.10/454
678	INTC Priority Select Register (INTC_PSR780)	16	R/W	8000h	17.6.10/454
67A	INTC Priority Select Register (INTC_PSR781)	16	R/W	8000h	17.6.10/454
67C	INTC Priority Select Register (INTC_PSR782)	16	R/W	8000h	17.6.10/454
67E	INTC Priority Select Register (INTC_PSR783)	16	R/W	8000h	17.6.10/454
680	INTC Priority Select Register (INTC_PSR784)	16	R/W	8000h	17.6.10/454
682	INTC Priority Select Register (INTC_PSR785)	16	R/W	8000h	17.6.10/454
684	INTC Priority Select Register (INTC_PSR786)	16	R/W	8000h	17.6.10/454
686	INTC Priority Select Register (INTC_PSR787)	16	R/W	8000h	17.6.10/454
688	INTC Priority Select Register (INTC_PSR788)	16	R/W	8000h	17.6.10/454
68A	INTC Priority Select Register (INTC_PSR789)	16	R/W	8000h	17.6.10/454
68C	INTC Priority Select Register (INTC_PSR790)	16	R/W	8000h	17.6.10/454
68E	INTC Priority Select Register (INTC_PSR791)	16	R/W	8000h	17.6.10/454
690	INTC Priority Select Register (INTC_PSR792)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
692	INTC Priority Select Register (INTC_PSR793)	16	R/W	8000h	17.6.10/454
694	INTC Priority Select Register (INTC_PSR794)	16	R/W	8000h	17.6.10/454
696	INTC Priority Select Register (INTC_PSR795)	16	R/W	8000h	17.6.10/454
698	INTC Priority Select Register (INTC_PSR796)	16	R/W	8000h	17.6.10/454
69A	INTC Priority Select Register (INTC_PSR797)	16	R/W	8000h	17.6.10/454
69C	INTC Priority Select Register (INTC_PSR798)	16	R/W	8000h	17.6.10/454
69E	INTC Priority Select Register (INTC_PSR799)	16	R/W	8000h	17.6.10/454
6A0	INTC Priority Select Register (INTC_PSR800)	16	R/W	8000h	17.6.10/454
6A2	INTC Priority Select Register (INTC_PSR801)	16	R/W	8000h	17.6.10/454
6A4	INTC Priority Select Register (INTC_PSR802)	16	R/W	8000h	17.6.10/454
6A6	INTC Priority Select Register (INTC_PSR803)	16	R/W	8000h	17.6.10/454
6A8	INTC Priority Select Register (INTC_PSR804)	16	R/W	8000h	17.6.10/454
6AA	INTC Priority Select Register (INTC_PSR805)	16	R/W	8000h	17.6.10/454
6AC	INTC Priority Select Register (INTC_PSR806)	16	R/W	8000h	17.6.10/454
6AE	INTC Priority Select Register (INTC_PSR807)	16	R/W	8000h	17.6.10/454
6B0	INTC Priority Select Register (INTC_PSR808)	16	R/W	8000h	17.6.10/454
6B2	INTC Priority Select Register (INTC_PSR809)	16	R/W	8000h	17.6.10/454
6B4	INTC Priority Select Register (INTC_PSR810)	16	R/W	8000h	17.6.10/454
6B6	INTC Priority Select Register (INTC_PSR811)	16	R/W	8000h	17.6.10/454
6B8	INTC Priority Select Register (INTC_PSR812)	16	R/W	8000h	17.6.10/454
6BA	INTC Priority Select Register (INTC_PSR813)	16	R/W	8000h	17.6.10/454
6BC	INTC Priority Select Register (INTC_PSR814)	16	R/W	8000h	17.6.10/454
6BE	INTC Priority Select Register (INTC_PSR815)	16	R/W	8000h	17.6.10/454
6C0	INTC Priority Select Register (INTC_PSR816)	16	R/W	8000h	17.6.10/454
6C2	INTC Priority Select Register (INTC_PSR817)	16	R/W	8000h	17.6.10/454
6C4	INTC Priority Select Register (INTC_PSR818)	16	R/W	8000h	17.6.10/454
6C6	INTC Priority Select Register (INTC_PSR819)	16	R/W	8000h	17.6.10/454
6C8	INTC Priority Select Register (INTC_PSR820)	16	R/W	8000h	17.6.10/454
6CA	INTC Priority Select Register (INTC_PSR821)	16	R/W	8000h	17.6.10/454
6CC	INTC Priority Select Register (INTC_PSR822)	16	R/W	8000h	17.6.10/454
6CE	INTC Priority Select Register (INTC_PSR823)	16	R/W	8000h	17.6.10/454
6D0	INTC Priority Select Register (INTC_PSR824)	16	R/W	8000h	17.6.10/454
6D2	INTC Priority Select Register (INTC_PSR825)	16	R/W	8000h	17.6.10/454
6D4	INTC Priority Select Register (INTC_PSR826)	16	R/W	8000h	17.6.10/454
6D6	INTC Priority Select Register (INTC_PSR827)	16	R/W	8000h	17.6.10/454
6D8	INTC Priority Select Register (INTC_PSR828)	16	R/W	8000h	17.6.10/454
6DA	INTC Priority Select Register (INTC_PSR829)	16	R/W	8000h	17.6.10/454
6DC	INTC Priority Select Register (INTC_PSR830)	16	R/W	8000h	17.6.10/454
6DE	INTC Priority Select Register (INTC_PSR831)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
6E0	INTC Priority Select Register (INTC_PSR832)	16	R/W	8000h	17.6.10/454
6E2	INTC Priority Select Register (INTC_PSR833)	16	R/W	8000h	17.6.10/454
6E4	INTC Priority Select Register (INTC_PSR834)	16	R/W	8000h	17.6.10/454
6E6	INTC Priority Select Register (INTC_PSR835)	16	R/W	8000h	17.6.10/454
6E8	INTC Priority Select Register (INTC_PSR836)	16	R/W	8000h	17.6.10/454
6EA	INTC Priority Select Register (INTC_PSR837)	16	R/W	8000h	17.6.10/454
6EC	INTC Priority Select Register (INTC_PSR838)	16	R/W	8000h	17.6.10/454
6EE	INTC Priority Select Register (INTC_PSR839)	16	R/W	8000h	17.6.10/454
6F0	INTC Priority Select Register (INTC_PSR840)	16	R/W	8000h	17.6.10/454
6F2	INTC Priority Select Register (INTC_PSR841)	16	R/W	8000h	17.6.10/454
6F4	INTC Priority Select Register (INTC_PSR842)	16	R/W	8000h	17.6.10/454
6F6	INTC Priority Select Register (INTC_PSR843)	16	R/W	8000h	17.6.10/454
6F8	INTC Priority Select Register (INTC_PSR844)	16	R/W	8000h	17.6.10/454
6FA	INTC Priority Select Register (INTC_PSR845)	16	R/W	8000h	17.6.10/454
6FC	INTC Priority Select Register (INTC_PSR846)	16	R/W	8000h	17.6.10/454
6FE	INTC Priority Select Register (INTC_PSR847)	16	R/W	8000h	17.6.10/454
700	INTC Priority Select Register (INTC_PSR848)	16	R/W	8000h	17.6.10/454
702	INTC Priority Select Register (INTC_PSR849)	16	R/W	8000h	17.6.10/454
704	INTC Priority Select Register (INTC_PSR850)	16	R/W	8000h	17.6.10/454
706	INTC Priority Select Register (INTC_PSR851)	16	R/W	8000h	17.6.10/454
708	INTC Priority Select Register (INTC_PSR852)	16	R/W	8000h	17.6.10/454
70A	INTC Priority Select Register (INTC_PSR853)	16	R/W	8000h	17.6.10/454
70C	INTC Priority Select Register (INTC_PSR854)	16	R/W	8000h	17.6.10/454
70E	INTC Priority Select Register (INTC_PSR855)	16	R/W	8000h	17.6.10/454
710	INTC Priority Select Register (INTC_PSR856)	16	R/W	8000h	17.6.10/454
712	INTC Priority Select Register (INTC_PSR857)	16	R/W	8000h	17.6.10/454
714	INTC Priority Select Register (INTC_PSR858)	16	R/W	8000h	17.6.10/454
716	INTC Priority Select Register (INTC_PSR859)	16	R/W	8000h	17.6.10/454
718	INTC Priority Select Register (INTC_PSR860)	16	R/W	8000h	17.6.10/454
71A	INTC Priority Select Register (INTC_PSR861)	16	R/W	8000h	17.6.10/454
71C	INTC Priority Select Register (INTC_PSR862)	16	R/W	8000h	17.6.10/454
71E	INTC Priority Select Register (INTC_PSR863)	16	R/W	8000h	17.6.10/454
720	INTC Priority Select Register (INTC_PSR864)	16	R/W	8000h	17.6.10/454
722	INTC Priority Select Register (INTC_PSR865)	16	R/W	8000h	17.6.10/454
724	INTC Priority Select Register (INTC_PSR866)	16	R/W	8000h	17.6.10/454
726	INTC Priority Select Register (INTC_PSR867)	16	R/W	8000h	17.6.10/454
728	INTC Priority Select Register (INTC_PSR868)	16	R/W	8000h	17.6.10/454
72A	INTC Priority Select Register (INTC_PSR869)	16	R/W	8000h	17.6.10/454
72C	INTC Priority Select Register (INTC_PSR870)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
72E	INTC Priority Select Register (INTC_PSR871)	16	R/W	8000h	17.6.10/454
730	INTC Priority Select Register (INTC_PSR872)	16	R/W	8000h	17.6.10/454
732	INTC Priority Select Register (INTC_PSR873)	16	R/W	8000h	17.6.10/454
734	INTC Priority Select Register (INTC_PSR874)	16	R/W	8000h	17.6.10/454
736	INTC Priority Select Register (INTC_PSR875)	16	R/W	8000h	17.6.10/454
738	INTC Priority Select Register (INTC_PSR876)	16	R/W	8000h	17.6.10/454
73A	INTC Priority Select Register (INTC_PSR877)	16	R/W	8000h	17.6.10/454
73C	INTC Priority Select Register (INTC_PSR878)	16	R/W	8000h	17.6.10/454
73E	INTC Priority Select Register (INTC_PSR879)	16	R/W	8000h	17.6.10/454
740	INTC Priority Select Register (INTC_PSR880)	16	R/W	8000h	17.6.10/454
742	INTC Priority Select Register (INTC_PSR881)	16	R/W	8000h	17.6.10/454
744	INTC Priority Select Register (INTC_PSR882)	16	R/W	8000h	17.6.10/454
746	INTC Priority Select Register (INTC_PSR883)	16	R/W	8000h	17.6.10/454
748	INTC Priority Select Register (INTC_PSR884)	16	R/W	8000h	17.6.10/454
74A	INTC Priority Select Register (INTC_PSR885)	16	R/W	8000h	17.6.10/454
74C	INTC Priority Select Register (INTC_PSR886)	16	R/W	8000h	17.6.10/454
74E	INTC Priority Select Register (INTC_PSR887)	16	R/W	8000h	17.6.10/454
750	INTC Priority Select Register (INTC_PSR888)	16	R/W	8000h	17.6.10/454
752	INTC Priority Select Register (INTC_PSR889)	16	R/W	8000h	17.6.10/454
754	INTC Priority Select Register (INTC_PSR890)	16	R/W	8000h	17.6.10/454
756	INTC Priority Select Register (INTC_PSR891)	16	R/W	8000h	17.6.10/454
758	INTC Priority Select Register (INTC_PSR892)	16	R/W	8000h	17.6.10/454
75A	INTC Priority Select Register (INTC_PSR893)	16	R/W	8000h	17.6.10/454
75C	INTC Priority Select Register (INTC_PSR894)	16	R/W	8000h	17.6.10/454
75E	INTC Priority Select Register (INTC_PSR895)	16	R/W	8000h	17.6.10/454
760	INTC Priority Select Register (INTC_PSR896)	16	R/W	8000h	17.6.10/454
762	INTC Priority Select Register (INTC_PSR897)	16	R/W	8000h	17.6.10/454
764	INTC Priority Select Register (INTC_PSR898)	16	R/W	8000h	17.6.10/454
766	INTC Priority Select Register (INTC_PSR899)	16	R/W	8000h	17.6.10/454
768	INTC Priority Select Register (INTC_PSR900)	16	R/W	8000h	17.6.10/454
76A	INTC Priority Select Register (INTC_PSR901)	16	R/W	8000h	17.6.10/454
76C	INTC Priority Select Register (INTC_PSR902)	16	R/W	8000h	17.6.10/454
76E	INTC Priority Select Register (INTC_PSR903)	16	R/W	8000h	17.6.10/454
770	INTC Priority Select Register (INTC_PSR904)	16	R/W	8000h	17.6.10/454
772	INTC Priority Select Register (INTC_PSR905)	16	R/W	8000h	17.6.10/454
774	INTC Priority Select Register (INTC_PSR906)	16	R/W	8000h	17.6.10/454
776	INTC Priority Select Register (INTC_PSR907)	16	R/W	8000h	17.6.10/454
778	INTC Priority Select Register (INTC_PSR908)	16	R/W	8000h	17.6.10/454
77A	INTC Priority Select Register (INTC_PSR909)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
77C	INTC Priority Select Register (INTC_PSR910)	16	R/W	8000h	17.6.10/454
77E	INTC Priority Select Register (INTC_PSR911)	16	R/W	8000h	17.6.10/454
780	INTC Priority Select Register (INTC_PSR912)	16	R/W	8000h	17.6.10/454
782	INTC Priority Select Register (INTC_PSR913)	16	R/W	8000h	17.6.10/454
784	INTC Priority Select Register (INTC_PSR914)	16	R/W	8000h	17.6.10/454
786	INTC Priority Select Register (INTC_PSR915)	16	R/W	8000h	17.6.10/454
788	INTC Priority Select Register (INTC_PSR916)	16	R/W	8000h	17.6.10/454
78A	INTC Priority Select Register (INTC_PSR917)	16	R/W	8000h	17.6.10/454
78C	INTC Priority Select Register (INTC_PSR918)	16	R/W	8000h	17.6.10/454
78E	INTC Priority Select Register (INTC_PSR919)	16	R/W	8000h	17.6.10/454
790	INTC Priority Select Register (INTC_PSR920)	16	R/W	8000h	17.6.10/454
792	INTC Priority Select Register (INTC_PSR921)	16	R/W	8000h	17.6.10/454
794	INTC Priority Select Register (INTC_PSR922)	16	R/W	8000h	17.6.10/454
796	INTC Priority Select Register (INTC_PSR923)	16	R/W	8000h	17.6.10/454
798	INTC Priority Select Register (INTC_PSR924)	16	R/W	8000h	17.6.10/454
79A	INTC Priority Select Register (INTC_PSR925)	16	R/W	8000h	17.6.10/454
79C	INTC Priority Select Register (INTC_PSR926)	16	R/W	8000h	17.6.10/454
79E	INTC Priority Select Register (INTC_PSR927)	16	R/W	8000h	17.6.10/454
7A0	INTC Priority Select Register (INTC_PSR928)	16	R/W	8000h	17.6.10/454
7A2	INTC Priority Select Register (INTC_PSR929)	16	R/W	8000h	17.6.10/454
7A4	INTC Priority Select Register (INTC_PSR930)	16	R/W	8000h	17.6.10/454
7A6	INTC Priority Select Register (INTC_PSR931)	16	R/W	8000h	17.6.10/454
7A8	INTC Priority Select Register (INTC_PSR932)	16	R/W	8000h	17.6.10/454
7AA	INTC Priority Select Register (INTC_PSR933)	16	R/W	8000h	17.6.10/454
7AC	INTC Priority Select Register (INTC_PSR934)	16	R/W	8000h	17.6.10/454
7AE	INTC Priority Select Register (INTC_PSR935)	16	R/W	8000h	17.6.10/454
7B0	INTC Priority Select Register (INTC_PSR936)	16	R/W	8000h	17.6.10/454
7B2	INTC Priority Select Register (INTC_PSR937)	16	R/W	8000h	17.6.10/454
7B4	INTC Priority Select Register (INTC_PSR938)	16	R/W	8000h	17.6.10/454
7B6	INTC Priority Select Register (INTC_PSR939)	16	R/W	8000h	17.6.10/454
7B8	INTC Priority Select Register (INTC_PSR940)	16	R/W	8000h	17.6.10/454
7BA	INTC Priority Select Register (INTC_PSR941)	16	R/W	8000h	17.6.10/454
7BC	INTC Priority Select Register (INTC_PSR942)	16	R/W	8000h	17.6.10/454
7BE	INTC Priority Select Register (INTC_PSR943)	16	R/W	8000h	17.6.10/454
7C0	INTC Priority Select Register (INTC_PSR944)	16	R/W	8000h	17.6.10/454
7C2	INTC Priority Select Register (INTC_PSR945)	16	R/W	8000h	17.6.10/454
7C4	INTC Priority Select Register (INTC_PSR946)	16	R/W	8000h	17.6.10/454
7C6	INTC Priority Select Register (INTC_PSR947)	16	R/W	8000h	17.6.10/454
7C8	INTC Priority Select Register (INTC_PSR948)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
7CA	INTC Priority Select Register (INTC_PSR949)	16	R/W	8000h	17.6.10/454
7CC	INTC Priority Select Register (INTC_PSR950)	16	R/W	8000h	17.6.10/454
7CE	INTC Priority Select Register (INTC_PSR951)	16	R/W	8000h	17.6.10/454
7D0	INTC Priority Select Register (INTC_PSR952)	16	R/W	8000h	17.6.10/454
7D2	INTC Priority Select Register (INTC_PSR953)	16	R/W	8000h	17.6.10/454
7D4	INTC Priority Select Register (INTC_PSR954)	16	R/W	8000h	17.6.10/454
7D6	INTC Priority Select Register (INTC_PSR955)	16	R/W	8000h	17.6.10/454
7D8	INTC Priority Select Register (INTC_PSR956)	16	R/W	8000h	17.6.10/454
7DA	INTC Priority Select Register (INTC_PSR957)	16	R/W	8000h	17.6.10/454
7DC	INTC Priority Select Register (INTC_PSR958)	16	R/W	8000h	17.6.10/454
7DE	INTC Priority Select Register (INTC_PSR959)	16	R/W	8000h	17.6.10/454
7E0	INTC Priority Select Register (INTC_PSR960)	16	R/W	8000h	17.6.10/454
7E2	INTC Priority Select Register (INTC_PSR961)	16	R/W	8000h	17.6.10/454
7E4	INTC Priority Select Register (INTC_PSR962)	16	R/W	8000h	17.6.10/454
7E6	INTC Priority Select Register (INTC_PSR963)	16	R/W	8000h	17.6.10/454
7E8	INTC Priority Select Register (INTC_PSR964)	16	R/W	8000h	17.6.10/454
7EA	INTC Priority Select Register (INTC_PSR965)	16	R/W	8000h	17.6.10/454
7EC	INTC Priority Select Register (INTC_PSR966)	16	R/W	8000h	17.6.10/454
7EE	INTC Priority Select Register (INTC_PSR967)	16	R/W	8000h	17.6.10/454
7F0	INTC Priority Select Register (INTC_PSR968)	16	R/W	8000h	17.6.10/454
7F2	INTC Priority Select Register (INTC_PSR969)	16	R/W	8000h	17.6.10/454
7F4	INTC Priority Select Register (INTC_PSR970)	16	R/W	8000h	17.6.10/454
7F6	INTC Priority Select Register (INTC_PSR971)	16	R/W	8000h	17.6.10/454
7F8	INTC Priority Select Register (INTC_PSR972)	16	R/W	8000h	17.6.10/454
7FA	INTC Priority Select Register (INTC_PSR973)	16	R/W	8000h	17.6.10/454
7FC	INTC Priority Select Register (INTC_PSR974)	16	R/W	8000h	17.6.10/454
7FE	INTC Priority Select Register (INTC_PSR975)	16	R/W	8000h	17.6.10/454
800	INTC Priority Select Register (INTC_PSR976)	16	R/W	8000h	17.6.10/454
802	INTC Priority Select Register (INTC_PSR977)	16	R/W	8000h	17.6.10/454
804	INTC Priority Select Register (INTC_PSR978)	16	R/W	8000h	17.6.10/454
806	INTC Priority Select Register (INTC_PSR979)	16	R/W	8000h	17.6.10/454
808	INTC Priority Select Register (INTC_PSR980)	16	R/W	8000h	17.6.10/454
80A	INTC Priority Select Register (INTC_PSR981)	16	R/W	8000h	17.6.10/454
80C	INTC Priority Select Register (INTC_PSR982)	16	R/W	8000h	17.6.10/454
80E	INTC Priority Select Register (INTC_PSR983)	16	R/W	8000h	17.6.10/454
810	INTC Priority Select Register (INTC_PSR984)	16	R/W	8000h	17.6.10/454
812	INTC Priority Select Register (INTC_PSR985)	16	R/W	8000h	17.6.10/454
814	INTC Priority Select Register (INTC_PSR986)	16	R/W	8000h	17.6.10/454
816	INTC Priority Select Register (INTC_PSR987)	16	R/W	8000h	17.6.10/454

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
818	INTC Priority Select Register (INTC_PSR988)	16	R/W	8000h	17.6.10/454
81A	INTC Priority Select Register (INTC_PSR989)	16	R/W	8000h	17.6.10/454
81C	INTC Priority Select Register (INTC_PSR990)	16	R/W	8000h	17.6.10/454
81E	INTC Priority Select Register (INTC_PSR991)	16	R/W	8000h	17.6.10/454
820	INTC Priority Select Register (INTC_PSR992)	16	R/W	8000h	17.6.10/454
822	INTC Priority Select Register (INTC_PSR993)	16	R/W	8000h	17.6.10/454
824	INTC Priority Select Register (INTC_PSR994)	16	R/W	8000h	17.6.10/454
826	INTC Priority Select Register (INTC_PSR995)	16	R/W	8000h	17.6.10/454
828	INTC Priority Select Register (INTC_PSR996)	16	R/W	8000h	17.6.10/454
82A	INTC Priority Select Register (INTC_PSR997)	16	R/W	8000h	17.6.10/454
82C	INTC Priority Select Register (INTC_PSR998)	16	R/W	8000h	17.6.10/454
82E	INTC Priority Select Register (INTC_PSR999)	16	R/W	8000h	17.6.10/454
830	INTC Priority Select Register (INTC_PSR1000)	16	R/W	8000h	17.6.10/454
832	INTC Priority Select Register (INTC_PSR1001)	16	R/W	8000h	17.6.10/454
834	INTC Priority Select Register (INTC_PSR1002)	16	R/W	8000h	17.6.10/454
836	INTC Priority Select Register (INTC_PSR1003)	16	R/W	8000h	17.6.10/454
838	INTC Priority Select Register (INTC_PSR1004)	16	R/W	8000h	17.6.10/454
83A	INTC Priority Select Register (INTC_PSR1005)	16	R/W	8000h	17.6.10/454
83C	INTC Priority Select Register (INTC_PSR1006)	16	R/W	8000h	17.6.10/454
83E	INTC Priority Select Register (INTC_PSR1007)	16	R/W	8000h	17.6.10/454
840	INTC Priority Select Register (INTC_PSR1008)	16	R/W	8000h	17.6.10/454
842	INTC Priority Select Register (INTC_PSR1009)	16	R/W	8000h	17.6.10/454
844	INTC Priority Select Register (INTC_PSR1010)	16	R/W	8000h	17.6.10/454
846	INTC Priority Select Register (INTC_PSR1011)	16	R/W	8000h	17.6.10/454
848	INTC Priority Select Register (INTC_PSR1012)	16	R/W	8000h	17.6.10/454
84A	INTC Priority Select Register (INTC_PSR1013)	16	R/W	8000h	17.6.10/454
84C	INTC Priority Select Register (INTC_PSR1014)	16	R/W	8000h	17.6.10/454
84E	INTC Priority Select Register (INTC_PSR1015)	16	R/W	8000h	17.6.10/454
850	INTC Priority Select Register (INTC_PSR1016)	16	R/W	8000h	17.6.10/454
852	INTC Priority Select Register (INTC_PSR1017)	16	R/W	8000h	17.6.10/454
854	INTC Priority Select Register (INTC_PSR1018)	16	R/W	8000h	17.6.10/454
856	INTC Priority Select Register (INTC_PSR1019)	16	R/W	8000h	17.6.10/454
858	INTC Priority Select Register (INTC_PSR1020)	16	R/W	8000h	17.6.10/454
85A	INTC Priority Select Register (INTC_PSR1021)	16	R/W	8000h	17.6.10/454
85C	INTC Priority Select Register (INTC_PSR1022)	16	R/W	8000h	17.6.10/454
85E	INTC Priority Select Register (INTC_PSR1023)	16	R/W	8000h	17.6.10/454
1000	INTC Monitor Mode Register 0 (INTC_MMRC0)	32	R/W	0000_0000h	17.6.11/456
1004	INTC HIPRI Register (INTC_HIPRI0C0)	32	R/W	0000_0000h	17.6.12/457
1008	INTC HIPRI Register (INTC_HIPRI1C0)	32	R/W	0000_0000h	17.6.12/457

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
100C	INTC HIPRI Register (INTC_HIPRI2C0)	32	R/W	0000_0000h	17.6.12/457
1010	INTC HIPRI Register (INTC_HIPRI3C0)	32	R/W	0000_0000h	17.6.12/457
1014	INTC LAT Register (INTC_LAT0C0)	32	R/W	0000_0000h	17.6.13/457
1018	INTC LAT Register (INTC_LAT1C0)	32	R/W	0000_0000h	17.6.13/457
101C	INTC LAT Register (INTC_LAT2C0)	32	R/W	0000_0000h	17.6.13/457
1020	INTC LAT Register (INTC_LAT3C0)	32	R/W	0000_0000h	17.6.13/457
1024	INTC Timer Register (INTC_TIMER0C0)	32	R/W	0000_0000h	17.6.14/458
1028	INTC Timer Register (INTC_TIMER1C0)	32	R/W	0000_0000h	17.6.14/458
102C	INTC Timer Register (INTC_TIMER2C0)	32	R/W	0000_0000h	17.6.14/458
1030	INTC Timer Register (INTC_TIMER3C0)	32	R/W	0000_0000h	17.6.14/458
1034	INTC Monitor Mode Register 1 (INTC_MMRC1)	32	R/W	0000_0000h	17.6.15/458
1038	INTC HIPRI Register (INTC_HIPRI0C1)	32	R/W	0000_0000h	17.6.16/459
103C	INTC HIPRI Register (INTC_HIPRI1C1)	32	R/W	0000_0000h	17.6.16/459
1040	INTC HIPRI Register (INTC_HIPRI2C1)	32	R/W	0000_0000h	17.6.16/459
1044	INTC HIPRI Register (INTC_HIPRI3C1)	32	R/W	0000_0000h	17.6.16/459
1048	INTC LAT Register (INTC_LAT0C1)	32	R/W	0000_0000h	17.6.17/459
104C	INTC LAT Register (INTC_LAT1C1)	32	R/W	0000_0000h	17.6.17/459
1050	INTC LAT Register (INTC_LAT2C1)	32	R/W	0000_0000h	17.6.17/459
1054	INTC LAT Register (INTC_LAT3C1)	32	R/W	0000_0000h	17.6.17/459
1058	INTC Timer Register (INTC_TIMER0C1)	32	R/W	0000_0000h	17.6.18/460
105C	INTC Timer Register (INTC_TIMER1C1)	32	R/W	0000_0000h	17.6.18/460
1060	INTC Timer Register (INTC_TIMER2C1)	32	R/W	0000_0000h	17.6.18/460
1064	INTC Timer Register (INTC_TIMER3C1)	32	R/W	0000_0000h	17.6.18/460

17.6.1 INTC Block Configuration Register (INTC_BCR)

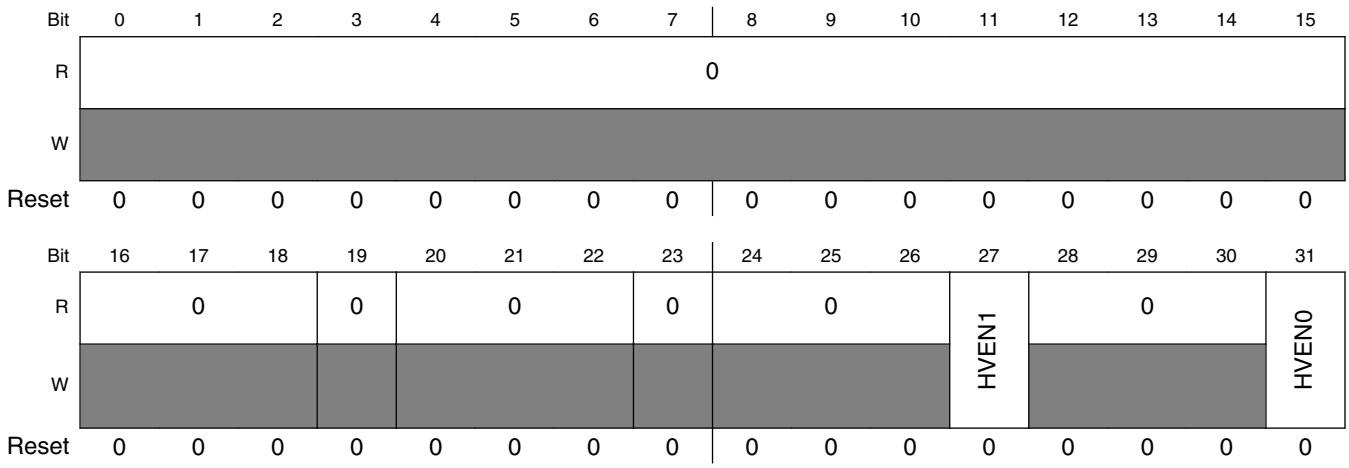
The Block Configuration Register is used to configure options of the INTC.

The master ID is used to determine if this register can be written by a given processor. Only the processor with master ID zero will be allowed write access, otherwise, a termination error is asserted.

If INTC_MPROT[MPROT]=0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] has write access, otherwise, a termination error is asserted.

Address: 0h base + 0h offset = 0h



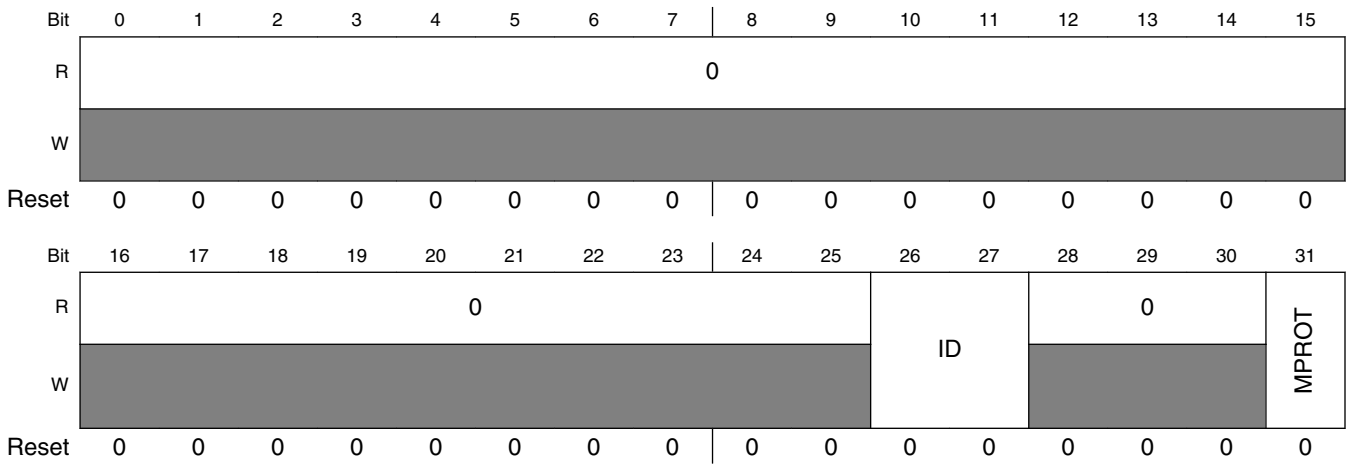
INTC_BCR field descriptions

Field	Description
0–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 HVEN1	Hardware vector enable for processor 1. Controls whether the INTC is in hardware vector mode or software vector mode. See Modes of operation for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode
28–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 HVEN0	Hardware vector enable for processor 0. Controls whether the INTC is in hardware vector mode or software vector mode. See Modes of operation for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode

17.6.2 INTC Master Protection Register (INTC_MPROT)

The Master Protection Register is used to protect cores from writing to other cores INTC registers.

Address: 0h base + 4h offset = 4h



INTC_MPROT field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–27 ID	ID This register can only be updated if MPROT=0 or if processor master ID=ID. When MPROT is set, the ID should also be updated to indicate which master can write to any register in the INTC memory map. Only the field values related to core bus masters can be selected. Non-core bus masters cannot be selected and would result in a bus access error. 00 master0 (reset default) 01 master1 10 master2 11 master3
28–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 MPROT	Master Protection Controls what programming space a given master can write to. If MPROT is disabled, then all masters can write to any valid entry of the INTC memory map. If MPROT is enabled, INTC_MPROT[ID] can write to any valid entry of the INTC memory map but other masters can only write to their assigned resources. Once MPROT is enabled by a master, only that master may modify INTC_MPROT. 0 Master programming space protection disabled 1 Master programming space protection enabled

17.6.3 INTC Current Priority Register for Processor 0 (INTC_CPR0)

The INTC_CPRn masks any peripheral or software-settable interrupt request that is set at the same or lower priority as the current value of the INTC_CPR*n*[PRI] field from generating an interrupt request to the processor. When the INTC Interrupt Acknowledge register (INTC_IACKR*n*) is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode, the value of PRI is pushed onto the LIFO, and PRI is updated with the priority of the preempting interrupt request. When the INTC end-of-interrupt register (INTC_EOIR*n*) is written, the LIFO is popped into the INTC_CPR*n* PRI field. An exception case in hardware vector mode to this behavior is described in Hardware vector mode.

The masking priority can be raised or lowered by writing to the PRI field, supporting the priority ceiling protocol. See Priority ceiling protocol.

The master ID is used to determine if this register can be written by a given processor. The processor with master ID zero will be allowed write access. Only the corresponding processor will have write access, otherwise, a termination error is asserted.

If INTC_MPROT[MPROT]=0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has write access, otherwise, a termination error is asserted.

NOTE

A store to modify the PRI field that closely precedes or follows an access to a shared resource may result in a non-coherent access to that resource. See Ensuring coherency for example code to ensure coherency.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																										PRI					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

INTC_CPR0 field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 PRI	Priority of the currently executing ISR, according to the field values 63 (highest priority) down to 0 (lowest priority).

17.6.4 INTC Current Priority Register for Processor 1 (INTC_CPR1)

The INTC_CPRn masks any peripheral or software-settable interrupt request that is set at the same or lower priority as the current value of the INTC_CPR*n*[PRI] field from generating an interrupt request to the processor. When the INTC Interrupt Acknowledge register (INTC_IACKR*n*) is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode, the value of PRI is pushed onto the LIFO, and PRI is updated with the priority of the preempting interrupt request. When the INTC end-of-interrupt register (INTC_EOIR*n*) is written, the LIFO is popped into the INTC_CPR*n* PRI field. An exception case in hardware vector mode to this behavior is described in Hardware vector mode.

The masking priority can be raised or lowered by writing to the PRI field, supporting the priority ceiling protocol. See Priority ceiling protocol.

The master ID is used to determine if this register can be written by a given processor. The processor with master ID zero will be allowed write access. Only the corresponding processor will have write access, otherwise, a termination error is asserted.

If INTC_MPROT[MPROT]=0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has write access, otherwise, a termination error is asserted.

NOTE

A store to modify the PRI field that closely precedes or follows an access to a shared resource may result in a non-coherent access to that resource. See Ensuring coherency for example code to ensure coherency.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																										PRI					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

INTC_CPR1 field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 PRI	Priority of the currently executing ISR, according to the field values 63 (highest priority) down to 0 (lowest priority).

17.6.5 INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)

The Interrupt Acknowledge Register for Processor *n* provides a value which can be used to load the address of an ISR from a vector table. The vector table can be composed of addresses of the ISRs specific to their respective interrupt vectors.

Also, in software vector mode, the INTC_IACKR*n* has side effects from reads unless INTC_MPROT[MPROT]=1 and processor with master ID (0-3) != INTC_MPROT[ID]. Therefore, it must not be read speculatively while in this mode. The side effects are the same regardless of the size of the read. Reading the INTC_IACKRn does not have side effects in hardware vector mode.

The master ID is used to determine if this register can be written by a given processor. The processor with master ID zero will be allowed write access. Only the corresponding processor will have write access, otherwise, a termination error is asserted.

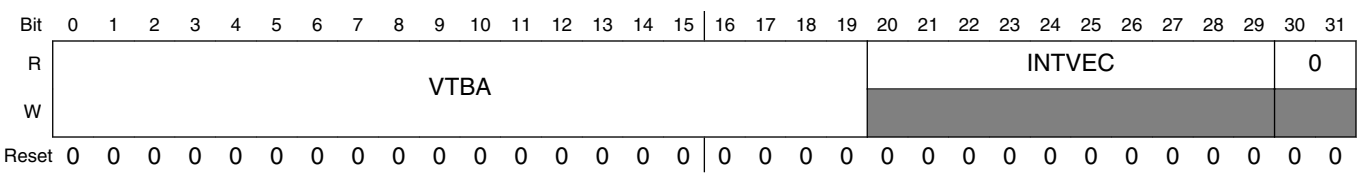
If INTC_MPROT[MPROT]=0 (disabled), all processors have read and write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has read and write access, otherwise, a termination error is asserted.

NOTE

When the corresponding INTC_BCR[HVENn] = 1, a read of the INTC_IACKR has no side effects.

Address: 0h base + 20h offset = 20h



INTC_IACKR0 field descriptions

Field	Description
0–19 VTBA	Vector table base address. Can be the base address of a vector table of addresses of ISRs.
20–29 INTVEC	Interrupt vector. Vector of the peripheral or software-settable interrupt request that caused the interrupt request to the processor. When the interrupt request to the processor asserts, the INTVEC is updated, whether the INTC is in software or hardware vector mode.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

17.6.6 INTC Interrupt Acknowledge Register for Processor 1 (INTC_IACKR1)

The Interrupt Acknowledge Register for Processor *n* provides a value which can be used to load the address of an ISR from a vector table. The vector table can be composed of addresses of the ISRs specific to their respective interrupt vectors.

Also, in software vector mode, the INTC_IACKR*n* has side effects from reads unless INTC_MPROT[MPROT]=1 and processor with master ID (0-3) != INTC_MPROT[ID]. Therefore, it must not be read speculatively while in this mode. The side effects are the same regardless of the size of the read. Reading the INTC_IACKRn does not have side effects in hardware vector mode.

The master ID is used to determine if this register can be written by a given processor. The processor with master ID zero will be allowed write access. Only the corresponding processor will have write access, otherwise, a termination error is asserted.

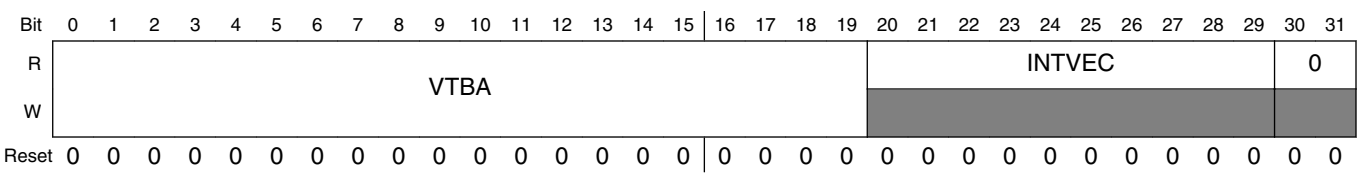
If INTC_MPROT[MPROT]=0 (disabled), all processors have read and write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has read and write access, otherwise, a termination error is asserted.

NOTE

When the corresponding INTC_BCR[HVENn] = 1, a read of the INTC_IACKR has no side effects.

Address: 0h base + 24h offset = 24h



INTC_IACKR1 field descriptions

Field	Description
0–19 VTBA	Vector table base address. Can be the base address of a vector table of addresses of ISRs.
20–29 INTVEC	Interrupt vector. Vector of the peripheral or software-settable interrupt request that caused the interrupt request to the processor. When the interrupt request to the processor asserts, the INTVEC is updated, whether the INTC is in software or hardware vector mode.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

17.6.7 INTC End Of Interrupt Register for Processor 0 (INTC_EOIR0)

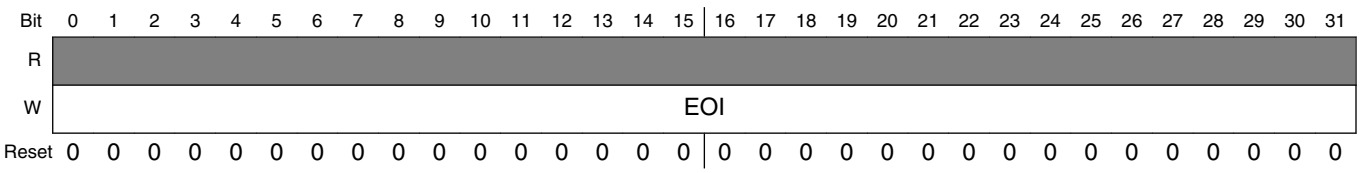
Writing to the INTC_EOIR*n* signals the end of the servicing of the interrupt request. When the INTC_EOIR*n* is written, the priority last pushed on the LIFO is popped into INTC_CPR*n*. The values and size of data written to the INTC_EOIR*n* are ignored. Those values and sizes written to this register neither update the INTC_EOIR*n* contents nor affect whether the LIFO pops. Typically, when you write to this register, write four all-zero bytes.

The master ID is used to determine if this register can be written by a given processor. The processor with master ID zero will be allowed write access. Only the corresponding processor will have write access, otherwise, a termination error is asserted.

If INTC_MPROT[MPROT]=0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has write access, otherwise, a termination error is asserted.

Address: 0h base + 30h offset = 30h



INTC_EOIR0 field descriptions

Field	Description
0–31 EOI	End of Interrupt. Write four all-zero bytes to this field to signal the end of the servicing of an interrupt request.

17.6.8 INTC End Of Interrupt Register for Processor 1 (INTC_EOIR1)

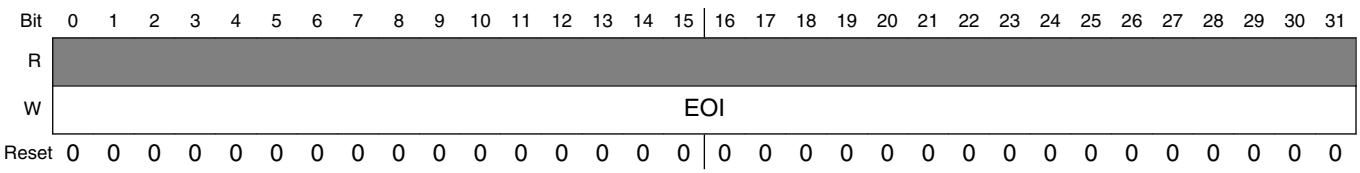
Writing to the INTC_EOIR*n* signals the end of the servicing of the interrupt request. When the INTC_EOIR*n* is written, the priority last pushed on the LIFO is popped into INTC_CPR*n*. The values and size of data written to the INTC_EOIR*n* are ignored. Those values and sizes written to this register neither update the INTC_EOIR*n* contents nor affect whether the LIFO pops. Typically, when you write to this register, write four all-zero bytes.

The master ID is used to determine if this register can be written by a given processor. The processor with master ID zero will be allowed write access. Only the corresponding processor will have write access, otherwise, a termination error is asserted.

If INTC_MPROT[MPROT]=0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has write access, otherwise, a termination error is asserted.

Address: 0h base + 34h offset = 34h



INTC_EOIR1 field descriptions

Field	Description
0–31 EOI	End of Interrupt. Write four all-zero bytes to this field to signal the end of the servicing of an interrupt request.

17.6.9 INTC Software Set/Clear Interrupt Register (INTC_SSCIR*n*)

The INTC_SSCIR registers support the setting or clearing of software-settable interrupt requests. These registers contain independent sets of bits to set and clear a corresponding flag bit by software. Unlike the SWT bit in the PSRs, the INTC_SSCIR registers do not require a read-modify-write. With the exception of being set by software, this flag bit behaves the same as a flag bit set within a peripheral. This flag bit generates an interrupt request within the INTC just like a peripheral interrupt request. Writing a 1 to SET will leave SET unchanged at 0 but will set CLR. Writing a 0 to SET will have no effect. CLR is the flag bit. Writing a 1 to CLR will clear it. Writing a 0 to CLR will have no effect. If a 1 is written to a pair of SET and CLR bits at the same time, CLR is asserted, regardless of whether CLR was asserted before the write.

Any processor will be allowed to write to any of the SET bits in the INTC_SSCIRn registers. The CLR bit will only be writable by the processor which has been assigned to handle that interrupt request, as determined by the setting of the INTC_PSRn[PRC_SEL]. An attempt by any other processor to write the CLR bit and not the SET bit will result in a termination error.

See Figure 17-2 for limitations on writing to this register.

NOTE

Since the four software setable interrupts in a word can be assigned to different processors, the cores with non-zero master IDs must only write the INTC_SSCIRn registers using a byte operation. A write of any other size will result in a termination error.

Any processor will be allowed to write to any of the SET bits in the INTC_SSCIRn registers.

If INTC_MPROT[MPROT]=0 (disabled), all processors can write to the CLR bit.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor can write to the CLR bit, otherwise, a termination error is asserted.

NOTE

If INTC_MPROT[MPROT] is enabled, processors with master ID != INTC_MPROT[ID] must only write the INTC_SSCIRn registers using a byte operation. A write of any other size will result in a termination error.

Address: 0h base + 40h offset + (1d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read	0						0	CLR
Write							SET	w1c
Reset	0	0	0	0	0	0	0	0

INTC_SSCIR*n* field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 SET	Set flag bits. Writing a 1 will set the corresponding CLR bit. Writing a 0 will have no effect. Each SET is read as a 0.
7 CLR	Clear flag bits. CLR is the flag bit. Writing a 1 to CLR will clear it provided that a 1 is not written simultaneously to its corresponding SET bit. Writing a 0 to CLR will have no effect. 0 Interrupt request not pending within INTC. 1 Interrupt request pending within INTC.

17.6.10 INTC Priority Select Register (INTC_PSR*n*)

The Priority Select Registers support the selection of an individual priority for each source of interrupt request, and the routing of the request to one or more processors. The unique vector of each peripheral or software-settable interrupt request determines which INTC_PSRn is assigned to that interrupt request. The software-settable interrupt requests are assigned the lowest numbered vectors, and their priorities are configured in the lowest numbered INTC_PSR registers. The peripheral interrupt requests are assigned to vectors immediately following the software-settable interrupt requests, and their priorities are configured in the immediately following INTC_PSR registers. The peripheral interrupt requests are assigned vectors 32-1023, and their priorities are configured in INTC_PSR32-1023, respectively.

Only the processor with master ID zero will be allowed to write the INTC_PSRn[PRC_SEL,SWT]. When writing the INTC_PSR[PRI], only the processor with master ID 0 and the processor who's master ID matches the setting of the INTC_PSRn[PRC_SEL] will be allowed to write the INTC_PSRn[PRI]. An attempt by any other processor to write the PSRn[PRIn] will result in a termination error.

If INTC_MPROT[MPROT]=0 (disabled), all processors have write access.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has write access to INTC_PSRn[PRC_SEL,SWT], otherwise, a termination error is asserted. When writing the INTC_PSRn[PRI], only the processor with master ID (0-3) = INTC_MPROT[ID] and the processor whose master ID (0-3) matches the setting of the INTC_PSRn[PRC_SEL] will be allowed to write the INTC_PSRn[PRI]. An attempt by any other processor to write the INTC_PSRn[PRI] will result in a termination error.

See Figure 17-2 for limitations on writing to this register.

NOTE

Unlike the peripheral interrupt request PSRs, there is no SWT bit in the PSRs corresponding to the software-settable interrupt requests.

NOTE

Interrupts are not detected while the PSRs are being written.

NOTE

If INTC_MPROT[MPROT] is enabled, the core with master ID (0-3) = INTC_MPROT[ID] will be able to write the INTC_PSRn with either byte, half-word or word operations. All

other master IDs are restricted to byte operations to the INTC_PSRn[PRI] bits (i.e the odd addresses). Any other sized write operation, or a byte operation to the INTC_PSRn[PRC_SEL] (i.e. the even byte) will result in a termination error.

NOTE

The core with master ID 0 will be able to write the INTC_PSRn with either byte, half-word or word operations. All other master IDs are restricted to byte operations to the PSRn[PRIn] bits (i.e the odd addresses). Any other sized write operation, or a byte operation to the INTC_PSRn[PRC_SEL] (i.e. the even byte) will result in a termination error.

Address: 0h base + 60h offset + (2d × i), where i=0d to 1023d

Bit	0	1	2	3	4	5	6	7
Read	PRC_ SELN0	PRC_ SELN1	0	0	0			SWTN
Write								
Reset	1	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0		PRIN					
Write								
Reset	0	0	0	0	0	0	0	0

INTC_PSR*n* field descriptions

Field	Description
0 PRC_SELN0	Processor select 0. If an interrupt source is enabled, this field selects whether the interrupt request is to be sent to processor 0. 0 Interrupt request not sent to processor 0 1 Interrupt request sent to processor 0
1 PRC_SELN1	Processor select 1. If an interrupt source is enabled, this field selects whether the interrupt request is to be sent to processor 1. 0 Interrupt request not sent to processor 1 1 Interrupt request sent to processor 1
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SWTN	This bit is only available in PSRs that do not correspond to the SSCIRs. The SWT bit supports triggering an interrupt by software rather than hardware. This is an alternative to setting a flag bit in a peripheral. This flag bit generates an interrupt request within the INTC just like a peripheral interrupt request. Writing a '1' to SWT*n* sets the bit. On the interrupt acknowledge cycle, the SWT*n* bit is cleared. The SWT function

Table continues on the next page...

INTC_PSR*n* field descriptions (continued)

Field	Description
	works both in hardware and software vector mode. It is only cleared by the enabled Processor(s) defined in the PRC_SEL*n* field of the PSR.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 PRIN	Priority select. Selects the priority for the interrupt requests. PRI has values 63 (highest priority) down to 0 (lowest priority).

17.6.11 INTC Monitor Mode Register 0 (INTC_MMRC0)

The Monitor Mode Register is used to configure measurement options of the INTC.

Address: 0h base + 1000h offset = 1000h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																														MM	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

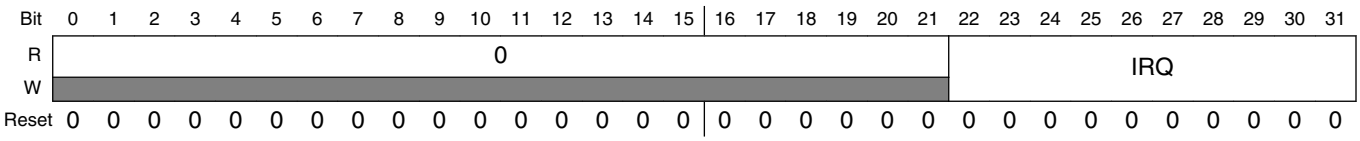
INTC_MMRC0 field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 MM	Monitor Mode. Controls whether the INTC monitors the latency or interrupt handler time 00 Disabled 01 IACK - Interrupt Acknowledge - timer is active from assertion of IRQ until interrupt acknowledge. An error is signaled if timer exceeds the programmed interrupt latency and the programmed interrupt latency is non-zero. 10 EOI - End Of Interrupt - timer is active from assertion of IRQ until the end of the interrupt handler. The timer will continue to run even if interrupt handler is preempted. An error is signaled if timer exceeds the programmed interrupt latency and the programmed interrupt latency is non-zero. 11 SW - Software - timer is active from assertion of IRQ until the end of the interrupt handler unless it is preempted. A return from interrupt will start the timer if it is the timer associated with the active IRQ. No error is signaled; rather software can read the timer at any time.

17.6.12 INTC HIPRI Register (INTC_HIPRI*n*C0)

The HIPRI log2(SOURCES)-bit registers are used to indicate which IRQ is to be monitored or checked. The mnemonic is of the form HIPRImCn, where the 'n' is the corresponding Processor and the ‘m’ is the monitor instance for that Processor.

Address: 0h base + 1004h offset + (4d × i), where i=0d to 3d



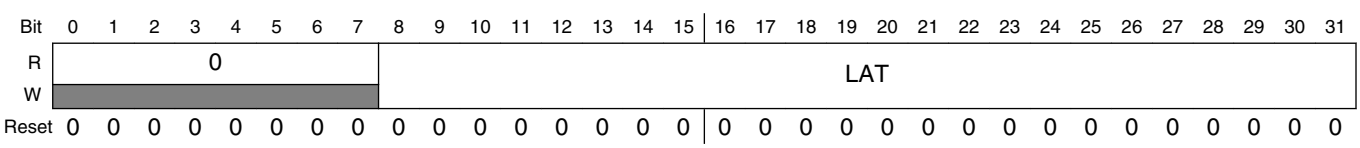
INTC_HIPRI*n*C0 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–31 IRQ	Select the IRQ to monitor (input the hex value of the IRQ)

17.6.13 INTC LAT Register (INTC_LAT*n*C0)

These registers are used to indicate the maximum time before an error signal is asserted for a detected IRQ to either IACK or RFI based on the Mode. The mnemonic is of the form LATmCn, where the n is the corresponding Processor and the ‘m’ is the monitor instance for that Processor.

Address: 0h base + 1014h offset + (4d × i), where i=0d to 3d



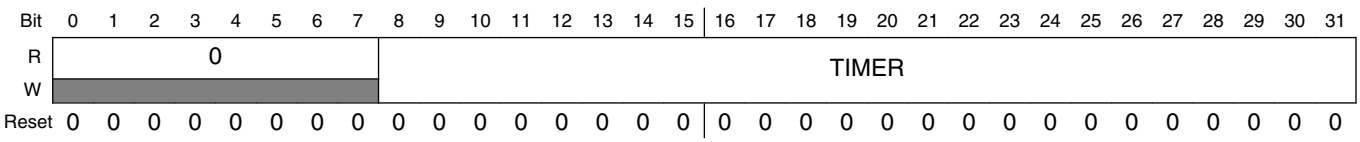
INTC_LAT*n*C0 field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 LAT	Select the maximum number of INTC clock cycles allowed for the monitored IRQ.

17.6.14 INTC Timer Register (INTC_TIMER*n*C0)

These registers are used to indicate the time for a detected IRQ to either IACK or RFI based on the Mode. The mnemonic is of the form TIMERmCn, where the n is the corresponding Processor and the ‘m’ is the monitor instance for that Processor are used to indicate the maximum time for a detected IRQ to either IACK or RFI based on the Mode.

Address: 0h base + 1024h offset + (4d × i), where i=0d to 3d



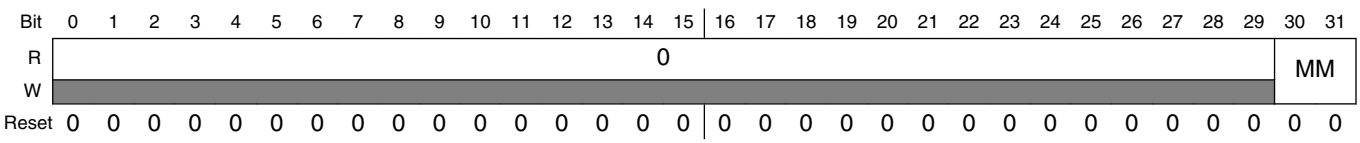
INTC_TIMER*n*C0 field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 TIMER	Timer is used to count the number of INTC clock cycles up to 24 bits of resolution.

17.6.15 INTC Monitor Mode Register 1 (INTC_MMRC1)

The Monitor Mode Register is used to configure measurement options of the INTC.

Address: 0h base + 1034h offset = 1034h



INTC_MMRC1 field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 MM	Monitor Mode. Controls whether the INTC monitors the latency or interrupt handler time 00 Disabled 01 IACK - Interrupt Acknowledge - timer is active from assertion of IRQ until interrupt acknowledge. An error is signaled if timer exceeds the programmed interrupt latency and the programmed interrupt latency is non-zero.

Table continues on the next page...

INTC_MMRC1 field descriptions (continued)

Field	Description
	10 EOI - End Of Interrupt - timer is active from assertion of IRQ until the end of the interrupt handler. The timer will continue to run even if interrupt handler is preempted. An error is signaled if timer exceeds the programmed interrupt latency and the programmed interrupt latency is non-zero. 11 SW - Software - timer is active from assertion of IRQ until the end of the interrupt handler unless it is preempted. A return from interrupt will start the timer if it is the timer associated with the active IRQ. No error is signaled; rather software can read the timer at any time.

17.6.16 INTC HIPRI Register (INTC_HIPRI*n*C1)

The HIPRI log2(SOURCES)-bit registers are used to indicate which IRQ is to be monitored or checked. The mnemonic is of the form HIPRImCn, where the 'n' is the corresponding Processor and the ‘m’ is the monitor instance for that Processor.

Address: 0h base + 1038h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																IRQ															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_HIPRI*n*C1 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–31 IRQ	Select the IRQ to monitor (input the hex value of the IRQ)

17.6.17 INTC LAT Register (INTC_LAT*n*C1)

These registers are used to indicate the maximum time before an error signal is asserted for a detected IRQ to either IACK or RFI based on the Mode. The mnemonic is of the form LATmCn, where the n is the corresponding Processor and the ‘m’ is the monitor instance for that Processor.

Address: 0h base + 1048h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								LAT																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_LAT*n*C1 field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 LAT	Select the maximum number of INTC clock cycles allowed for the monitored IRQ.

17.6.18 INTC Timer Register (INTC_TIMER*n*C1)

These registers are used to indicate the time for a detected IRQ to either IACK or RFI based on the Mode. The mnemonic is of the form TIMERmCn, where the n is the corresponding Processor and the ‘m’ is the monitor instance for that Processor are used to indicate the maximum time for a detected IRQ to either IACK or RFI based on the Mode.

Address: 0h base + 1058h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								TIMER																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_TIMER*n*C1 field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 TIMER	Timer is used to count the number of INTC clock cycles up to 24 bits of resolution.

17.7 Functional description

The functional description involves the areas of interrupt request sources, priority management, and handshaking with the processor. In addition, spaces in the memory map have been reserved for other possible implementations of the INTC.

17.7.1 Interrupt request sources

The INTC has two types of interrupt requests, peripheral and software-settable. These interrupt requests can assert on any clock cycle.

The INTC has no spurious vector support. Therefore, if an asserted peripheral or software-settable interrupt request, whose PRI*n* value in INTC_PSR*n* is higher than the PRI value in INTC_CPR*n*, negates before the interrupt request to the processor for that peripheral or software-settable interrupt request is acknowledged, then the interrupt request to the processor still can assert (or will remain asserted) for that peripheral or software-settable interrupt request. In this case, the interrupt vector will correspond to that peripheral or software-settable interrupt request. Also, the PRI value in the associated INTC_CPR*n* is updated with the corresponding PRI*n* value in INTC_PSR*n*.

Furthermore, clearing the peripheral interrupt request's enable bit in the peripheral, or (alternatively) setting its mask bit, has the same consequences as clearing its flag bit. Setting its enable bit or clearing its mask bit while its flag bit is asserted has the same effect on the INTC as an interrupt event setting the flag bit.

17.7.1.1 Peripheral interrupt requests

An interrupt event in a peripheral's hardware sets a flag bit which resides in that peripheral. The interrupt request from the peripheral is driven by that flag bit.

The time from when the peripheral starts to drive its peripheral interrupt request to the INTC, to the time that the INTC starts to drive the interrupt request to the processor, is three clocks.

17.7.1.2 Software-settable interrupt requests

The software set/clear interrupt registers (INTC_SSCIR*n*) support the setting or clearing of software-settable interrupt requests. These registers contain independent sets of bits to set and clear a corresponding flag bit by software. An interrupt request is triggered by software by writing a 1 to a SET bit in INTC_SSCIR*n*. This write sets the corresponding CLR bit, which is a flag bit, resulting in the interrupt request. The interrupt request is cleared by writing a 1 to the CLR bit. Specific behavior includes the following:

- Writing a 1 to SET leaves SET unchanged at 0 but sets the flag bit (which is the CLR bit).
- Writing a 0 to SET has no effect.
- Writing a 1 to CLR clears the flag (CLR) bit.
- Writing a 0 to CLR has no effect.
- If a 1 is written to a pair of SET and CLR bits at the same time, the flag (CLR) is set, regardless of whether CLR was asserted before the write.

The time from the write to the SET bit, to the time that the INTC starts to drive the interrupt request to the processor, is four clocks.

Any Processor n will be allowed to write to any of the SET bits in the Software Set/Clear Interrupt registers. The CLR bit will only be writable by the Processor n which has been assigned to handle that interrupt request, as determined by the setting of the INTC_PSRn[PRC_SELn] bits.

17.7.1.3 Unique vector for each interrupt request source

Each peripheral and software-settable interrupt request is assigned a hardwired unique 10-bit vector. Software-settable interrupts 0–31 are assigned vectors 0–31, respectively. The peripheral interrupt requests are assigned vectors from 32 to a number as high as needed to cover all peripheral interrupt requests.

17.7.2 Priority management

The asserted interrupt requests are compared to each other based on their PRI*n* and PRC_SEL*n* values set in INTC_PSR*n*. The result of that comparison also is compared to PRI in the associated INTC_CPR*n*. The results of those comparisons are used to manage the priority of the ISR being executed by the associated processor. The associated LIFO also assists in managing that priority.

17.7.2.1 Current priority and preemption

The priority arbitrator, selector, encoder, and comparator logic shown in Figure 17-1 are used to compare the priority of the asserted interrupt requests to the current priority. If the priority of any asserted peripheral or software-settable interrupt request is higher than the current priority for a given processor, then the interrupt request to the processor is asserted. Also, a unique vector for the preempting peripheral or software-settable interrupt request is generated for the associated INTC_IACKR*n*, and if in hardware vector mode, for the interrupt vector provided to the processor.

17.7.2.1.1 Priority tree

The priority tree for each processor compares all the priorities of all of the asserted interrupt requests assigned to that processor, both peripheral and software-settable. The output of the priority tree is the highest of those priorities assigned to a given processor. Also, any interrupt requests which have this highest priority are output as asserted interrupt requests to the associated selector logic.

17.7.2.1.2 Selector

If only one interrupt request from the associated priority tree is asserted, then it is passed as asserted to the associated encoder logic. If multiple interrupt requests from the associated priority tree are asserted, then only the one with the lowest vector is passed as asserted to the associated encoder logic. The lower vector is chosen regardless of the time order of the assertions of the peripheral or software-settable interrupt requests.

17.7.2.1.3 Encoder

The encoder logic generates the unique 10-bit vector for the asserted interrupt request from the request selector subblock for the associated processor.

17.7.2.1.4 Comparator

The comparator logic compares the highest priority output from the associated priority arbitrator subblock with PRI in the associated INTC_CPR*n*. If the comparator detects that this highest priority is higher than the current priority, then it asserts the interrupt request to the associated processor. This interrupt request to the processor asserts whether this highest priority is raised above the value of PRI in the associated INTC_CPR*n*, or the PRI value in the associated INTC_CPR*n* is lowered below this highest priority. This highest priority then becomes the new priority which is written to PRI in the associated INTC_CPR*n* when the interrupt request to the processor is acknowledged. Interrupt requests whose PRI*n* in INTC_PSR*n* are zero will not cause a preemption because their PRI*n* will not be higher than PRI in the associated INTC_CPR*n*.

Another function of the comparator is to signal an update of the INTC_IACKR*n* with the vector number of the first interrupt that arrives that has a priority higher than the current priority. Once the vector number and priority are captured, they cannot be superseded by a higher priority interrupt until an update of the INTC_CPR*n* occurs. In software vector mode, higher priority interrupts can supersede the previously captured interrupt vector number and priority until the time a hardware or software interrupt acknowledge is processed.

17.7.2.2 Stack (LIFO)

The LIFO stores the preempted PRI values from the associated INTC_CPR*n*. Therefore, because these priorities are stacked within the INTC, if interrupts need to be enabled during the ISR, at the beginning of the interrupt exception handler the PRI value in the associated INTC_CPR*n* does not need to be loaded from the associated INTC_CPR*n* and stored onto the context stack. Likewise at the end of the interrupt exception handler, the priority does not need to be loaded from the context stack and stored into the associated INTC_CPR*n*.

The PRI value in the associated INTC_CPR*n* is pushed onto the LIFO when the associated INTC_IACKR*n* is read in software vector mode or the interrupt acknowledge signal from the associated processor is asserted in hardware vector mode. The priority is popped into PRI in the associated INTC_CPR*n* whenever the associated INTC_EOIR*n* is written. An exception case in hardware vector mode to this behavior is described in Hardware vector mode.

Although the INTC supports up to 64 priorities, an ISR executing with PRI in the INTC_CPR*n* equal to 63 will not be preempted. Therefore, the LIFO supports the stacking of 63 priorities. However, the LIFO is only 62 entries deep. An entry for a priority of 0 is not needed because of the ways that pushing onto a full LIFO and popping an empty LIFO are treated. If the LIFO is pushed 63 or more times than it is popped, the priorities first pushed are overwritten. A priority of 0 would be an overwritten priority. However, the LIFO will pop zeroes if it is popped more times than it is pushed. Therefore, although a priority of 0 was overwritten, it is regenerated with the popping of an empty LIFO.

17.7.3 Handshaking with processor

This section describes handshaking with the processor.

17.7.3.1 Software vector mode handshaking

This section describes the software vector mode handshaking.

17.7.3.1.1 Acknowledging interrupt request to processor

The software vector mode handshaking can be used with processors that only support an interrupt request to them, or processors that support both an interrupt request by itself as well as an interrupt request with an interrupt vector. The software vector mode handshaking even supports processors that always expect an interrupt vector with the interrupt request to them. Refer to Figure 17-1101.

A timing diagram of the interrupt request and acknowledge handshaking in software vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in Figure 17-1101. The INTC examines the peripheral and software-settable interrupt requests. When it finds an asserted peripheral or software-settable interrupt request with a higher priority than PRI in the associated INTC_CPR*n*, it asserts the interrupt request to the associated processor. The INTVEC field in the associated INTC_IACKR*n* is updated with the preempting interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the processor is asserted. The rest of the handshaking is described in Software vector mode.

17.7.3.1.2 End of interrupt exception handler

Before the interrupt exception handling completes, INTC_EOIR*n* must be written. When it is written, the associated LIFO is popped so that the preempted priority is restored into PRI of the associated INTC_CPR*n*. Before it is written, the peripheral or software-settable flag bit must be cleared so that the peripheral or software-settable interrupt request is negated.

When returning from the preemption, the INTC does not search for the peripheral or software-settable interrupt request whose ISR was preempted. Depending on how much the ISR has progressed, that interrupt request may no longer even be asserted. When PRI in the associated INTC_CPR*n* is lowered to the priority of the preempted ISR, the interrupt request for the preempted ISR or any other asserted peripheral or software-settable interrupt request at or below that priority will not cause a preemption. Instead, after the restoration of the preempted context, the processor will return to the instruction address that it had been going to execute next, before it was preempted. This next instruction is part of the preempted ISR or the interrupt exception handler's prolog or epilog.

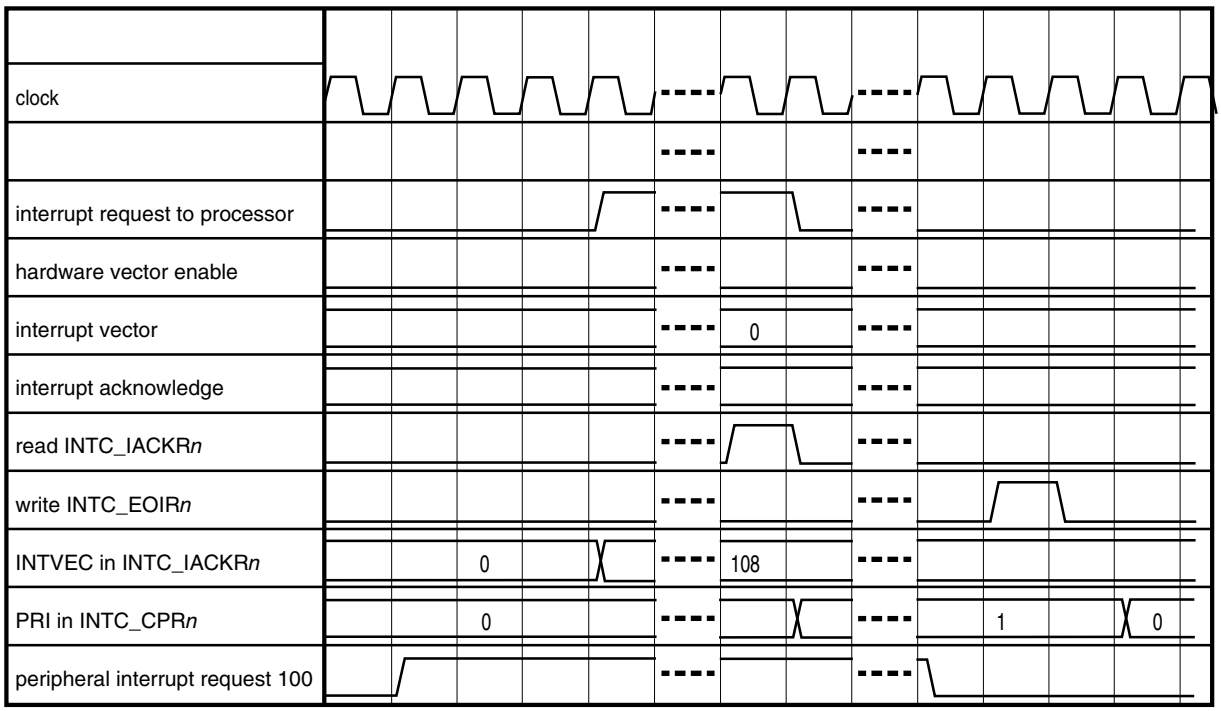


Figure 17-1101. Timing diagram of software vector mode handshaking

17.7.3.2 Hardware vector mode handshaking

A timing diagram of the interrupt request and acknowledge handshaking in hardware vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in Figure 17-1102. As in software vector mode, the INTC examines the peripheral and software-settable interrupt requests, and when it finds an asserted interrupt request with a higher priority than PRI in the associated INTC_CPR*n*, it asserts the interrupt request to the associated processor. The INTVEC field in the associated INTC_IACKR*n* is updated with the preempting peripheral or software-settable interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the associated processor is asserted. In addition, the value of the interrupt vector to the associated processor matches the value of the INTVEC field in the associated INTC_IACKR*n*. The rest of the handshaking is described in Hardware vector mode.

The handshaking near the end of the interrupt exception handler, that is the writing to the associated INTC_EOIR*n*, is the same as in software vector mode. See End of interrupt exception handler.

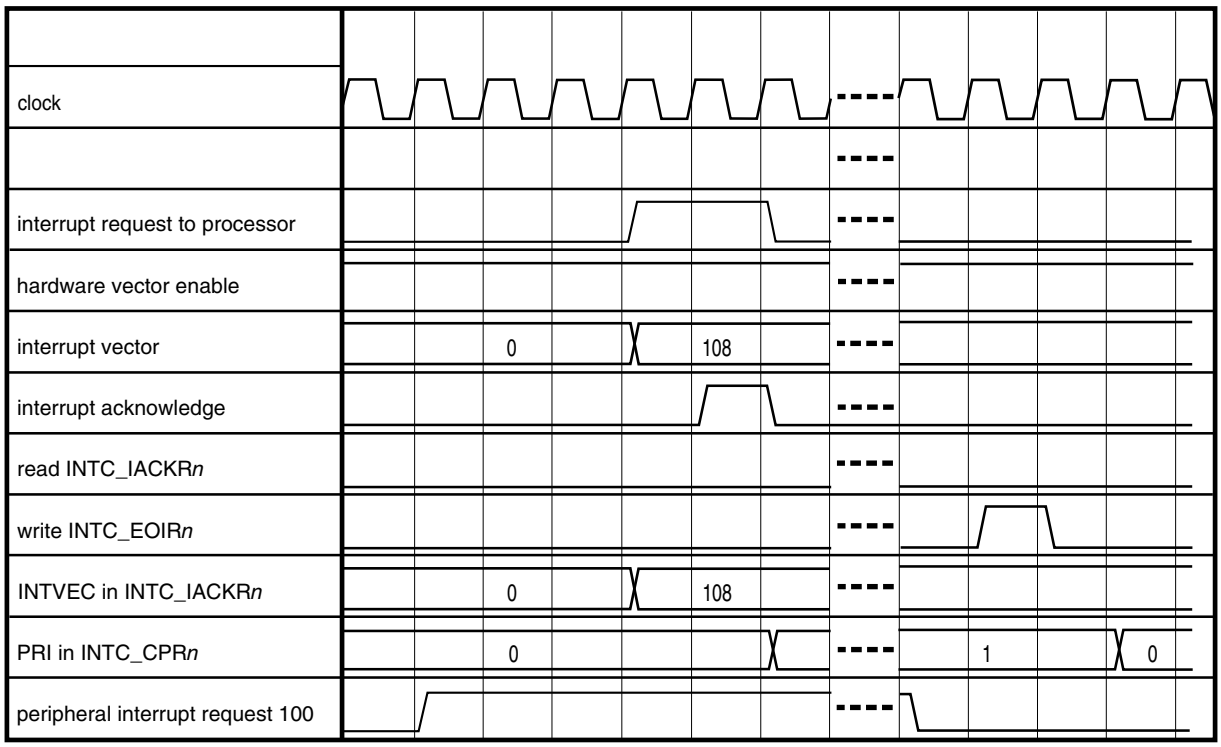


Figure 17-1102. Timing diagram for hardware vector mode handshaking

17.7.4 Interrupt Controller Monitor (INTCM)

Isolated in its own hierarchy, the interrupt latency monitor logic is used to monitor or check interrupt latency for reliability, safety, and interrupt software requirements.

17.7.4.1 Operation

The following section lists the operation of the INTCM.

17.7.4.1.1 TIMERn

A clear of TIMERn will clear the corresponding error.

The TIMERn will not run if there is a corresponding error.

If the INTC_MMRn[MM] is enabled and an error is signaled, disable TIMERn.

If the INTC_MMRn[MM] is enabled, for a transition from 0 to 1 of SOURCE input that is being monitored by INTC_HIPRIn registers, clear and enable TIMERn. Once the timer reaches the maximum count, it will stop and not roll over. Software can read the TIMERn value at any time.

If the INTC_MMRn[MM] is enabled and if the TIMERn is enabled, then it should increment each cycle.

If the INTC_MMRn[MM] is enabled and if INTC_LATn is non-zero, an error will be signaled if the timer exceeds INTC_LATn or if the TIMERn reaches the maximum count.

If the INTC_MMRn[MM] is IACK and an interrupt acknowledge asserts for the monitored source, INTC_HIPRIn[IRQ], then the enabled TIMERn should stop. As an extra check, the IRQ is used to ensure that the acknowledged interrupt matches the monitored interrupt INTC_HIPRIn[IRQ].

If the INTC_MMRn[MM] is SW and an interrupt acknowledge occurs where a monitored interrupt is preempted, the enabled TIMERn should stop.

If the INTC_MMRn[MM] is EOI or SW for a monitored interrupt which has an INTC_EOIRn, the enabled TIMERn should stop.

If the INTC_MMRn[MM] is SW and if INTC_EOIRn, and if the previous one is monitored, it will be re-enabled.

17.8 Initialization/application information

This section contains initialization/application information.

17.8.1 Initialization flow

After exiting reset, all of the PRI*n* and PRC_SEL*n* fields in INTC_PSR*n* are zero, and PRI in the INTC_CPRn registers is 63. These reset values will prevent the INTC from asserting the interrupt request to the processors. The enable or mask bits in the peripherals are reset such that the peripheral interrupt requests are negated. An initialization sequence for allowing the peripheral and software-settable interrupt requests to cause an interrupt request to the processor is:

- interrupt_request_initialization:
  - Configure HVENn in INTC_BCR.
  - Configure VTBAn in INTC_IACKRn.
  - Raise the PRIn fields and set the PRC_SELn fields to the desired processor in INTC_PSRn.
  - Set the enable bits or clear the mask bits for the peripheral interrupt requests.
  - Lower PRI in INTC_CPRn to zero.
  - Enable processor(s) recognition of interrupts.

17.8.2 Interrupt exception handler

These example interrupt exception handlers excerpts use Power Architecture assembly code.

17.8.2.1 Software vector mode

In software vector mode for Power Architecture, there are 16 bytes of vector space available at the single ISR entry point.

```
interrupt_exception_handler:
b        interrupt_exception_handler_continued# 16 bytes available, branch to continue

interrupt_exception_handler_continued:
code to create stack frame, save working register, and save SRR0 and SRR1 (Power
Architecture Save/Restore registers) (not shown)

lis        r3,INTC_IACKRn@ha        # form adjusted upper half of INTC_IACKRn address
lwz        r3,INTC_IACKRn@l(r3)     # load INTC_IACKRn, which clears request to processor
lwz        r3,0x0(r3)                # load address of ISR from vector table
wrteei    1                        # enable processor recognition of interrupts

code to save rest of context required by Power Architecture EABI (Embedded Binary
Application Interface) (not shown)

mtlr    r3                          # move the INTC_IACKRn address into the link register
blrl                              # branch to ISR; link register updated with epilog
                                  # address
epilog:
code to restore most of context required by Power Architecture EABI (not shown)
# Popping the LIFO after the restoration of most of the context and the disabling of
processor
# recognition of interrupts eases the calculation of the maximum stack depth at the cost of
# postponing the servicing of the next interrupt request.
mbar                              # ensure store to clear flag bit has completed
lis        r3,INTC_EOIRn@ha        # form adjusted upper half of INTC_EOIRn address
li        r4,0x0                   # form 0 to write to INTC_EOIRn
wrteei    0                        # disable processor recognition of interrupts
stw        r4,INTC_EOIRn@l(r3)        # store to INTC_EOIRn, informing INTC to lower priority
code to restore SRR0 and SRR1, restore working registers, and delete stack frame (not shown)

rfi

vector_table_base_address:
address of ISR for interrupt with vector 0
address of ISR for interrupt with vector 1
        .
        .
        .
address of ISR for interrupt with vector 1022
address of ISR for interrupt with vector 1023
ISRn:
code to service the interrupt event (not shown)
code to clear flag bit which drives interrupt request to INTC (not shown)
blr                               # return to epilog
```

17.8.2.2 Hardware vector mode

This interrupt exception handler is useful with processor and system bus implementations that support a hardware vector.

```
interrupt_exception_handlern:
b        interrupt_exception_handler_continuedn# 16 bytes available, branch to continue

interrupt_exception_handler_continuedn:
code to create stack frame, save working register, and save SRR0 and SRR1 (not shown)

wrteei    1                    # enable processor recognition of interrupts

code to save rest of context required by Power Architecture EABI (not shown)

bl        ISRn                # branch to ISR for interrupt with vector n

epilog:
code to restore most of context required by Power Architecture EABI (not shown)

# Popping the LIFO after the restoration of most of the context and the disabling of
processor
# recognition of interrupts eases the calculation of the maximum stack depth at the cost of
# postponing the servicing of the next interrupt request.
mbar                           # ensure store to clear flag bit has completed
lis        r3,INTC_EOIRn@ha        # form adjusted upper half of INTC_EOIRn address
li        r4,0x0               # form 0 to write to INTC_EOIRn
wrteei    0                   # disable processor recognition of interrupts
stw        r4,INTC_EOIRn@l(r3)    # store to INTC_EOIRn, informing INTC to lower priority

SRR0 and SRR1, restore working registers, and delete stack frame (not shown)

rfi

ISRn:
code to service the interrupt event
code to clear flag bit which drives interrupt request to INTC

blr        # branch to epilog
```

17.8.3 ISR, RTOS, and task hierarchy

The RTOS and all of the tasks under its control typically execute with PRI in INTC_CPR*n* having a value of 0. The RTOS will execute the tasks according to whatever priority scheme it may have, but that priority scheme is independent and has a lower priority of execution than the priority scheme of the INTC. In other words, the ISRs execute above INTC_CPR*n* priority 0 and outside the control of the RTOS, the RTOS executes at INTC_CPR*n* priority 0, and while the tasks execute at different priorities under the control of the RTOS, they also execute at INTC_CPRn priority 0.

If a task shares a resource with an ISR and the Priority Ceiling Protocol (PCP) is being used to manage that shared resource, then the task's priority can be elevated in the INTC_CPR*n* while the shared resource is being accessed.

An ISR whose PRI*n* in INTC_PSR*n* has a value of 0 will not cause an interrupt request to the selected processor, even if its peripheral or software-settable interrupt request is asserted. For a peripheral interrupt request, not setting its enable bit or disabling the mask bit will cause it to remain negated, which consequently also will not cause an interrupt request to the processor. Since the ISRs are outside the control of the RTOS, this ISR will not run unless called by another ISR or the interrupt exception handler, perhaps after executing another ISR.

17.8.4 Order of execution

An ISR with a higher priority can preempt an ISR with a lower priority, regardless of the unique vectors associated with each of their peripheral or software-settable interrupt requests. However, if multiple peripheral or software-settable interrupt requests are asserted, more than one of these interrupt requests has the highest priority and that priority is high enough to cause preemption, the INTC selects the one with the lowest unique vector regardless of the order in time that they asserted. However, the ability to meet deadlines with this scheduling scheme is no less than if the ISRs execute in the time order that their peripheral or software-settable interrupt requests asserted.

The example in the following table shows the order of execution of both cases, ISRs with different priorities and ISRs with the same priority.

Table 17-1101. Order of ISR execution example

Step #	Step Description	Code executing at end of step						PRI in INTC_CPR at end of step
		RTOS	ISR108 [1]	ISR208	ISR308	ISR408	interrupt exception handler	
1	RTOS at priority 0 is executing.	X						0
2	Peripheral interrupt request 100 at priority 1 asserts. Interrupt taken.		X					1
3	Peripheral interrupt request 400 at priority 4 asserts. Interrupt taken.					X		4
4	Peripheral interrupt request 300 at priority 3 asserts.					X		4
5	Peripheral interrupt request 200 at priority 3 asserts.					X		4
6	ISR408 completes. Interrupt exception handler writes to INTC_EOIR*n*.						X	1
7	Interrupt taken. ISR208 starts to execute, even though peripheral interrupt request 300 asserted first.			X				3
8	ISR208 completes. Interrupt exception handler writes to INTC_EOIR*n*.						X	1
9	Interrupt taken. ISR308 starts to execute.				X			3

Table continues on the next page...

Table 17-1101. Order of ISR execution example (continued)

Step #	Step Description	Code executing at end of step						PRI in INTC_CPR at end of step
		RTOS	ISR108 [1]	ISR208	ISR308	ISR408	interrupt exception handler	
10	ISR308 completes. Interrupt exception handler writes to INTC_EOIR*n*.						X	1
11	ISR108 completes. Interrupt exception handler writes to INTC_EOIR*n*.						X	0
12	RTOS continues execution.	X						0

1. ISR108 executes for peripheral interrupt request 100 because the first eight ISRs are for software-settable interrupt requests.

17.8.5 Priority ceiling protocol

This section describes the priority ceiling protocol.

17.8.5.1 Elevating priority

The PRI field in INTC current priority register (INTC_CPR*n*) is elevated in the OSEK PCP to the ceiling of all of the priorities of the ISRs that share a resource. This protocol therefore allows coherent accesses of the ISRs to that shared resource.

For example, ISR1 has a priority of 1, ISR2 has a priority of 2, and ISR3 has a priority of 3. They all share the same resource. Before ISR1 or ISR2 can access that resource, they must raise the PRI value in INTC_CPR*n* to 3, the ceiling of all of the ISR priorities. After they release the resource, they must lower the PRI value in INTC_CPR*n* to prevent further priority inversion. If they do not raise their priority, then ISR2 can preempt ISR1, and ISR3 can preempt ISR1 or ISR2, possibly corrupting the shared resource. Another possible failure mechanism is deadlock if the higher priority ISR needs the lower priority ISR to release the resource before it can continue, but the lower priority ISR cannot release the resource until the higher priority ISR completes and execution returns to the lower priority ISR.

Using the PCP instead of disabling processor recognition of all interrupts eliminates the time when accessing a shared resource that all higher priority interrupts are blocked. For example, while ISR3 cannot preempt ISR1 while it is accessing the shared resource, all of the ISRs with a priority higher than 3 can preempt ISR1.

17.8.5.2 Ensuring coherency

This section discusses how to ensure coherency.

17.8.5.2.1 Interrupt with blocked priority

A scenario can exist that can cause non-coherent accesses to the shared resource. As an example, ISR1 and ISR2 both share a resource. ISR1 has a lower priority than ISR2. ISR1 is executing, and it writes to the INTC_CPR*n*. The instruction following this store is a store to a value in a shared coherent data block. Either just before or at the same time as the first store, the INTC asserts the interrupt request to the processor because the peripheral interrupt request for ISR2 has asserted. As the processor is responding to the interrupt request from the INTC, and as it is aborting transactions and flushing its pipeline, it is possible (in some implementations) that both of these stores are executed. ISR2 thereby assumes that it can access the data block coherently, but the data block has been corrupted.

OSEK uses the GetResource and ReleaseResource system services to manage access to a shared resource. To prevent this corruption of a coherent data block, modifications to PRI in INTC_CPR*n* can be made by those system services with the code sequence:

```
GetResource:
        wrteei 0                # disable external interrupts to the Processor
        raise PRI                # Write to CPR, cache inhibit, guarded
        mbar                    # flush out writes from store buffer
        wrteei 1                # enable external interrupts to the Processor
        isync                    # re-fetch Processor pipeline

ReleaseResource:
        mbar                    # flush out writes from store buffer
        wrteei 0                # disable external interrupts to the Processor
        lower PRI                # Write to CPR, cache inhibit, guarded
        wrteei 1                # enable external interrupts to the Processor
```

17.8.5.2.1.1 Interrupt request to processor

Referencing Figure 17-1, the interrupt request logic to the processor is shown in the following figure.

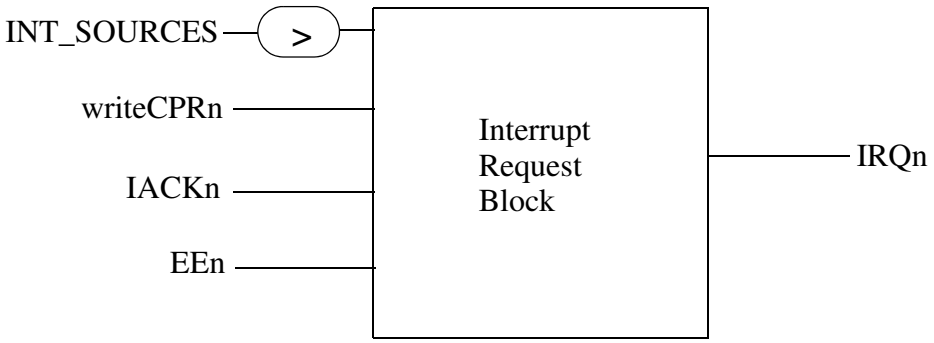


Figure 17-1103. Interrupt Request Block Diagram

Assuming that there are no IRQn deasserted (no pending interrupt requests to processor n), if one of the external interrupt sources (INT_SOURCES) has a priority higher than the INTC_CPRn, then interrupt request to the processor (IRQn) is asserted. It will stay asserted until one of the following conditions is true:

- Interrupt acknowledge (IACK) is asserted
- If the EEn bit has been cleared by the wrteei instruction (see the code sequence in Interrupt with blocked priority), IRQn will be re-evaluated when processor n writes to the INTC_CPRn while processor n ignores IRQn

This provides a safe way to guarantee that no interrupts will be recognized by processor n while updating the INTC_CPRn.

17.8.5.2.2 Raised priority preserved

Before the instruction after the GetResource system service executes, all pending transactions have completed. These pending transactions can include an ISR for a peripheral or software-settable interrupt request whose priority was equal to or lower than the raised priority. The shared coherent data block now can be accessed coherently. The following figure shows the timing diagram for this scenario, and the following table explains the events. The example is for software vector mode. Except for the method of retrieving the vector and acknowledging the interrupt request to the processor, hardware vector mode is identical.

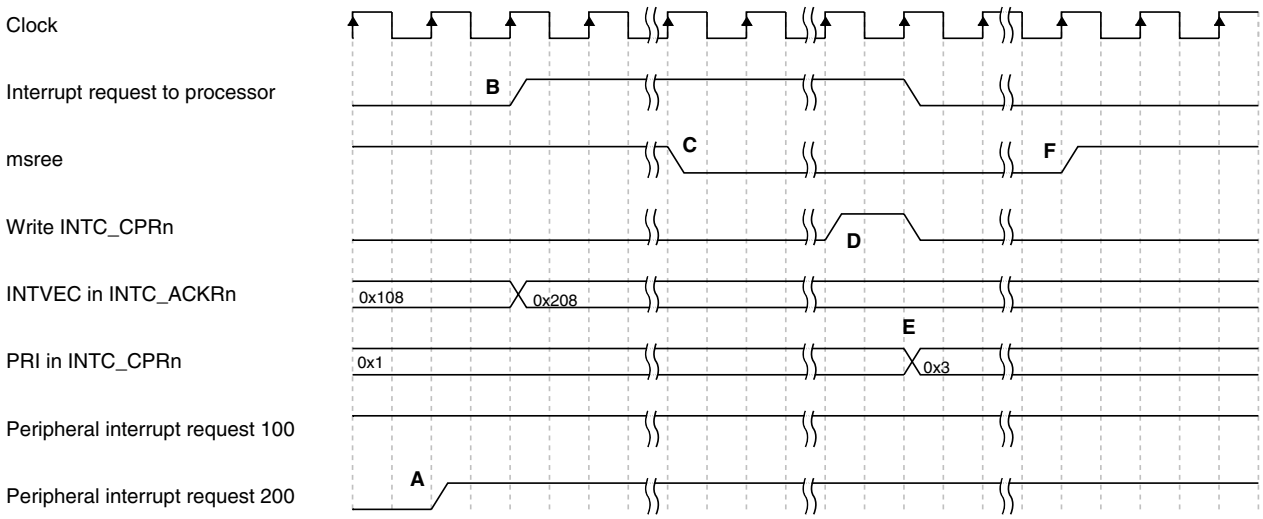


Figure 17-1104. Timing diagram of raised priority preserved

Table 17-1102. Raised priority preserved events

Event	Description
A	Peripheral interrupt request 200 of priority 2 asserts during execution of ISR108 running at priority 1.
B	Interrupt request to processor asserts. INTVEC in INTC_IACKRn updates with vector for that peripheral interrupt request.
C	Write to msree bit to disable external interrupts
D	ISR108 writes to INTC_CPRn to raise priority to 3 before accessing shared coherent data block.
E	PRI in INTC_CPRn now at 3, reflecting the write. This write, just before accessing data block, is the last instruction the processor executes before being interrupted.
F	Write to msree bit to enable external interrupts

17.8.6 Selecting priorities according to request rates and deadlines

The selection of the priorities for the ISRs can be made using Rate Monotonic Scheduling (RMS) or a superset of it, Deadline Monotonic Scheduling (DMS). In RMS, the ISRs which have higher request rates have higher priorities. In DMS, if the deadline is before the next time the ISR is requested, then the ISR is assigned a priority according to the time from the request for the ISR to the deadline, not from the time of the request for the ISR to the next request for it.

For example, ISR1 executes every 100 µs, ISR2 executes every 200 µs, and ISR3 executes every 300 µs. ISR1 has a higher priority than ISR2 which has a higher priority than ISR3. However, if ISR3 has a deadline of 150 µs, then it has a higher priority than ISR2.

The INTC supports 64 priorities, which may be many fewer than the number of ISRs. In this case, the ISRs should be grouped with other ISRs that have similar deadlines. For example, a priority could be allocated for every time the request rate doubles. ISRs with request rates around 1 ms would share a priority, ISRs with request rates around 500 µs would share a priority, ISRs with request rates around 250 µs would share a priority, and so on. With this approach, a range of ISR request rates of 2^{16} could be covered, regardless of the number of ISRs.

Reducing the number of priorities does cause some priority inversion, which reduces the processor's ability to meet its deadlines. However, reducing the number of priorities can reduce the size and latency through the interrupt controller. It also allows easier management of ISRs with similar deadlines that share a resource. They can be placed at the same priority without any further priority inversion, and they do not need to use the PCP to access the shared resource.

17.8.7 Software-settable interrupt requests

The software-settable interrupt requests can be used in two ways. They can be used to schedule a lower priority portion of an ISR and for processors to interrupt other processors in a multiple processor system.

17.8.7.1 Scheduling a lower priority portion of an ISR

A portion of an ISR needs to be executed at the PRIn value in INTC_PSR*n*, which becomes the PRI value in INTC_CPR*n* with the interrupt acknowledge. The ISR, however, can have a portion of it which does not need to be executed at this higher priority. Therefore, executing this later portion which does not need to be executed at this higher priority can prevent the execution of ISRs which do not have a higher priority than the earlier portion of the ISR but do have a higher priority than the later portion of the ISR needs. This preemptive scheduling inefficiency reduces the processor's ability to meet its deadlines.

One option is for the ISR to complete the earlier higher priority portion, but then schedule through the RTOS a task to execute the later lower priority portion. However, some RTOSs can require a large amount of time for an ISR to schedule a task. Therefore, a second option is for the ISR, after completing the higher priority portion, to set a SET bit

in INTC_SSCIR*n*. Writing a '1' to SET causes a software-settable interrupt request. This software-settable interrupt request, which usually will have a lower PRI*n* value in the INTC_PSR*n*, therefore will not cause preemptive scheduling inefficiencies.

17.8.7.2 Scheduling an ISR on another processor

Since the SET bits in the INTC_SSCIR*n* are memory mapped, processors in multiple processor systems can schedule ISRs on the other processors. One possible application is if one processor simply wants to command another processor to perform a piece of work, and the initiating processor does not need to use the results of that work. If the initiating processor needs to know if the processor executing the software-settable ISR has not completed the work before asking it to again execute that ISR, it can check if the corresponding CLR bit in INTC_SSCIR*n* is asserted before again writing a 1 to the SET bit.

Another application is the sharing of a block of data. For example, a first processor has completed accessing a block of data and wants a second processor to then access it. Furthermore, after the second processor has completed accessing the block of data, the first processor again wants to access it. The accesses to the block of data must be done coherently. The procedure is that the first processor writes a 1 to a SET bit on the second processor. The second processor, after accessing the block of data, clears the corresponding CLR bit and then writes 1 to a SET bit on the first processor, informing it that it now can access the block of data.

17.8.8 Lowering priority within an ISR

In implementations without the software-settable interrupt requests in INTC_SSCIR*n*, one way (besides scheduling a task through an RTOS) to prevent preemptive scheduling inefficiencies with an ISR whose work spans multiple priorities (as described in Scheduling a lower priority portion of an ISR) is to lower the current priority. However, the INTC has a LIFO whose depth is determined by the number of priorities.

Note

Lowering the PRI value in INTC_CPR*n* within an ISR to below the ISR's corresponding PRI value in INTC_PSR*n* allows more preemptions than the depth of the LIFO can support.

Therefore, through its use of the LIFO the INTC does not support lowering the current priority within an ISR as a way to avoid preemptive scheduling inefficiencies.

17.8.9 Negating an interrupt request outside of its ISR

This section discusses the negation of an interrupt service request outside of its ISR.

17.8.9.1 Negating an interrupt request as a side effect of an ISR

Some peripherals have flag bits which can be cleared as a side effect of servicing a peripheral interrupt request. For example, reading a specific register can clear the flag bits, and consequently their corresponding interrupt requests too. This clearing as a side effect of servicing a peripheral interrupt request can cause the negation of other peripheral interrupt requests besides the peripheral interrupt request whose ISR presently is executing. This negating of a peripheral interrupt request outside of its ISR can be a desired effect.

17.8.9.2 Negating multiple interrupt requests in one ISR

An ISR can clear other flag bits besides its own flag bit. One reason that an ISR clears multiple flag bits is because it serviced those other flag bits, and therefore the ISRs for these other flag bits do not need to be executed.

17.8.9.3 Proper setting of interrupt request priority

Whether an interrupt request negates outside of its own ISR due to the side effect of an ISR execution or the intentional clearing of a flag bit, the priorities of the peripheral or software-settable interrupt requests for these other flag bits must be selected properly. Their PRI*n* values in INTC_PSR*n* must be selected to be at or lower than the priority of the ISR that cleared their flag bits. Otherwise, those flag bits still can cause the interrupt request to the processor to assert. Furthermore, the clearing of these other flag bits also has the same timing relationship to the writing to INTC_SSCIR*n* as the clearing of the flag bit that caused the present ISR to be executed. Refer to End of interrupt exception handler for more information.

A flag bit whose enable bit or mask bit is negating its peripheral interrupt request can be cleared at any time, regardless of the peripheral interrupt request's PRI*n* value in INTC_PSR*n*.

17.8.10 Examining LIFO contents

There are multiple methods for tracking interrupts in a system, one of which is described here using existing hardware (LIFO) within the INTC. Although the LIFO contents are not memory-mapped, the user can read the contents by popping the LIFO and reading the PRI field in the INTC current priority register (INTC_CPR*n*). To avoid a lower-level interrupt being serviced because the popping of the LIFO updates the INTC_CPR*n*, the processor recognition of interrupts should be disabled when examining LIFO contents. The pseudo-code is as follows:

```
      wrteei 0                # disable processor recognition of external interrupts
      oldCPRn = INTC_CPRn;    # save INTC_CPRn
      (branch to the pop_lifo)

pop_lifo:
      store INTC_EIORn        # pop INTC_CPR from LIFO, examine PRI, etc...
      examine INTC_CPRn[PRI], and store onto stack
      if PRI is not zero or value when interrupts were enabled, branch to pop_lifo
      branch to push_lifo
```

When the examination is complete, the LIFO can be restored using this code sequence:

```
push_lifo:
      load stacked PRI value and store to INTC_CPRn
      load INTC_IACKn         # IACK; push INTC_CPRn into LIFO
      if stacked PRI values are not depleted, branch to push_lifo

      INTC_CPRn    = oldCPRn; # restore original INTC_CPRn and re-evaluate pending external
interrupts
      wrteei 1                # enable processor recognition of interrupts
```

NOTE

Disabling the processor recognition of interrupts during LIFO examination will introduce latency, but none of the interrupt requests will be missed.

17.9 Interrupt sources

The list of interrupt sources is chip-specific. For this list, see the chip-specific INTC information.

Chapter 18
Enhanced Direct Memory Access (eDMA)

18.1 Introduction

The enhanced direct memory access (eDMA) controller is a second-generation module capable of performing complex data transfers with minimal intervention from a host processor. The hardware microarchitecture includes:

- A DMA engine that performs:
  - Source- and destination-address calculations
  - Data-movement operations
- Local memory containing transfer control descriptors for each of the 64 channels

The following figure is a block diagram of the eDMA module.

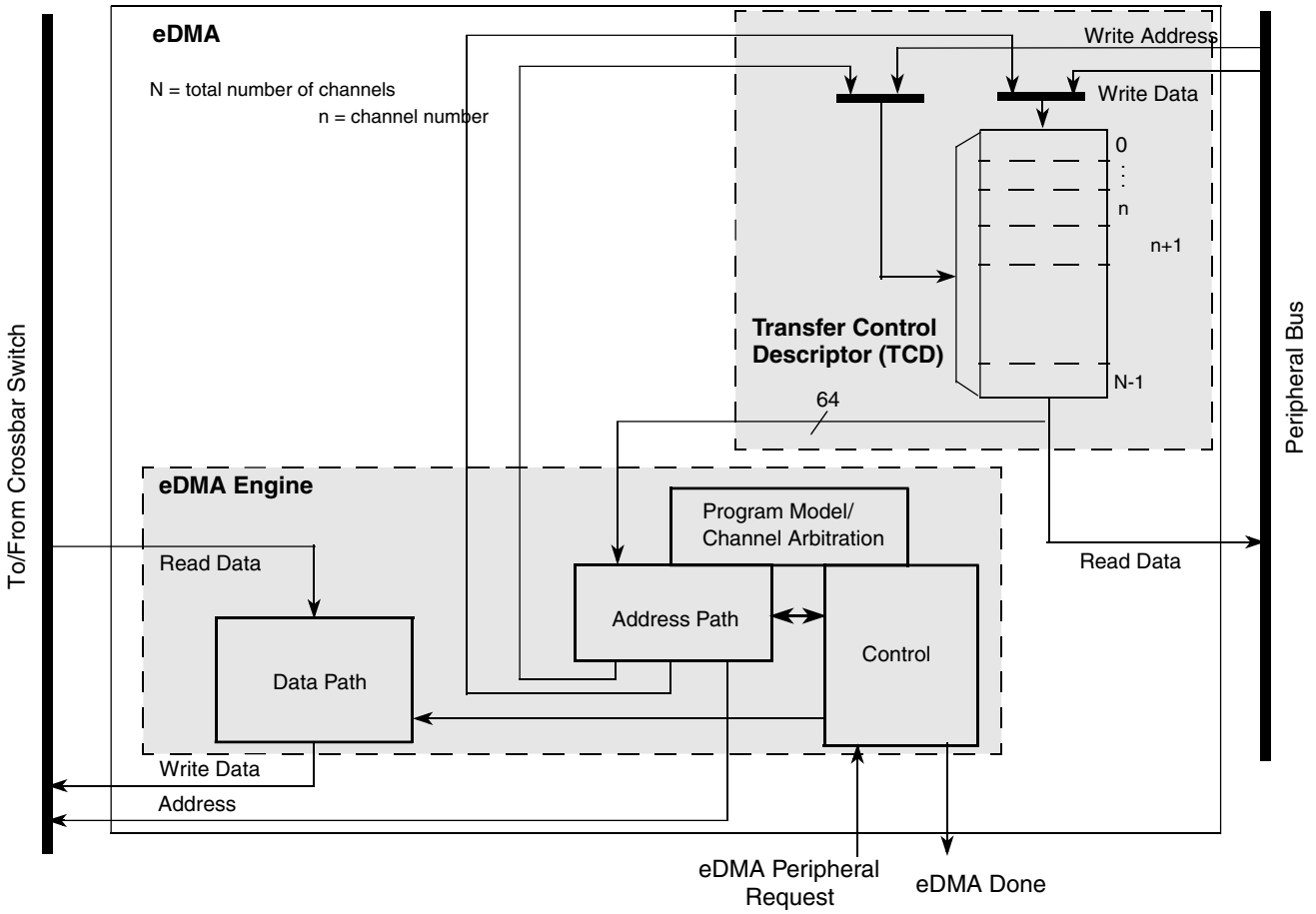


Figure 18-1. eDMA Block Diagram

18.2 Overview

The eDMA is a highly-programmable data-transfer engine optimized to minimize the required intervention from the host processor. It is intended for use in applications where the data size to be transferred is statically known and not defined within the data packet itself.

Throughout this chapter, n is used to reference the channel number. Additionally, data sizes are defined as byte (8-bit), halfword (16-bit), word (32-bit) and doubleword (64-bit).

18.3 Features

The eDMA module has these features:

- All data movement via dual-address transfers: read from source, write to destination

- Programmable source, destination addresses, transfer size, plus support for enhanced addressing modes
- 64-channel implementation that performs complex data transfers with minimal intervention from a host processor
  - Internal data buffer, used as temporary storage to support 16-byte burst transfers
  - Connections to the crossbar switch for bus mastering the data movement
- Transfer control descriptor organized to support two-deep, nested transfer operations
  - An inner data transfer loop defined by a “minor” byte transfer count
  - An outer data transfer loop defined by a “major” iteration count
- Channel service request via one of the three following methods. For all three methods, one service request per execution of the minor loop is required.
  a. Explicit software initiation
  b. Initiation via a channel-to-channel linking mechanism for continuous transfers with independent channel linking at end of minor loop and/or major loop
  c. Peripheral-paced hardware requests (one per channel)
- Support for fixed-priority and round-robin channel arbitration
- Channel completion reported via optional interrupt requests (chip dependent)
  - One interrupt per channel, optionally asserted at completion of major iteration count
  - Error terminations are optionally enabled per channel, and logically summed together to form a small number of error interrupt outputs
- Support for scatter/gather DMA processing
- Support for complex data structures
- Support to cancel transfers via software or hardware

The structure of the transfer control descriptor is fundamental to the operation of the eDMA module. It is defined below in a ‘C’ pseudo-code specification, where int refers to a 32-bit variable (unless noted otherwise) and short is a 16-bit variable.

NOTE

To compile these structures, change any periods '.' in the variable name to underscores '_'.

```
typedef union {
    struct { /* citer.e_link = 1 */
        unsigned short citer.linkch:6; /* link channel number, */
        unsigned short citer:9; /* current (“major”) iteration count */
    } minor_link_enabled;/* channel link at end of the minor loop */
    struct {/* citer.e_link = 0 */
        unsigned short citer:15; /* current (“major”) iteration count */
    } minor_link_disabled;/* no linking at end of the minor loop */
} t_minor_link_citer;

typedef union {
    struct { /* biter.e_link = 1 */
        unsigned short biter.linkch:6; /* link channel number, */
        unsigned short biter:9; /* beginning (“major”) iteration count */
    } init_minor_link_enabled;/* channel link at end of the minor loop */
    struct {/* biter.e_link = 0 */
```

```
        unsigned short biter:15; /* beginning ("major") iteration count */
    } init_minor_link_disabled;/* no linking at end of the minor loop */
} t_minor_link_biter;

typedef struct {
unsigned intsaddr;/* source address */
unsigned intsmod:5;/* source address modulo */
unsigned intssize:3;/* source transfer size */
unsigned intdmod:5;/* destination address modulo */
unsigned intdsize:3;/* destination transfer size */
shortsoff;/* signed source address offset */
unsigned intnbytes;/* inner ("minor") byte count */
intslast;/* last source address adjustment */
unsigned intdaddr;/* destination address */
unsigned shortciter.e_link:1;/* enable channel linking on minor loop */
t_minor_link_citerminor_link_citer;/* conditional current iteration count */
shortdoff;/* signed destination address offset */
intdlast_sga;/* last destination address adjustment, or
         scatter/gather address (if e_sg = 1) */
unsigned shortbiter.e_link:1;/* beginning channel link enable */
t_minor_link_biterminor_link_biter;/* beginning ("major") iteration count */
unsigned intbwc:2;/* bandwidth control */
unsigned intmajor.linkch:6;/* link channel number */
unsigned intdone:1;/* channel done */
unsigned intactive:1;/* channel executing */
unsigned intmajor.e_link:1;/* enable channel linking on major loop*/
unsigned inte_sg:1;/* enable scatter/gather descriptor */
unsigned intd_req:1;/* disable ipd_req when done */
unsigned intint_half:1;/* interrupt on citer = (biter >> 1) */
unsigned intint_maj:1;/* interrupt on major loop completion */
unsigned intstart:1;/* explicit channel start */
} tcd/* transfer_control_descriptor */
```

The basic operation of a channel is defined as:

1. The channel is initialized by software loading the transfer control descriptor into the eDMA's programming model, memory-mapped through the slave bus space, and implemented as local memory.
2. The channel requests service; either explicitly by software, a peripheral request or a linkage from another channel.

NOTE

The major loop executes one iteration per service request.

3. The contents of the transfer control descriptor for the activated channel is read from the local memory and loaded into the eDMA engine's internal register file.
4. The eDMA engine executes the data transfer defined by the transfer control descriptor, reading from the source and writing to the destination. The number of iterations in the minor loop is automatically calculated by the eDMA engine. The number of iterations within the minor loop is a function of the number of bytes to transfer (nbytes), the source size (ssize), and the destination size (dsize). The completion of the minor loop is equal to one iteration of the major loop.
5. At the conclusion of the minor loop's execution, certain fields of the transfer control descriptor are written back to the local TCD memory.

The process (steps 2-5) is repeated until the outer major loop's iteration count is exhausted. At that time, additional processing steps are completed, e.g., the optional assertion of an interrupt request signaling the transfer's completion, final adjustments to the source and destination addresses, etc. A more detailed description of the channel processing is listed in the pseudo-code below. This simplified example is intended to represent basic data transfers. Detailed processing associated with the error handling is omitted.

```
/* the given eDMA channel is requesting service by the software assertion of the
   tcd[channel].start bit, the assertion of an enabled ipd_req from a device, or
   the implicit assertion of a channel-to-channel link */

/* begin by reading the transfer control descriptor from the local RAM
   into the local dma_engine registers */
dma_engine        = read_from_local_memory [channel];
dma_engine.active = 1;                /* set active flag */
dma_engine.done   = 0;                /* clear done flag */

/* check the transfer control descriptor for consistency */
if (dma_engine.config_error == 0) {

    / * begin execution of the inner "minor" loop transfers */
    {

        /* convert the source transfer size into a byte count */
        switch (dma_engine.ssize) {
        case 0:                       /* 8-bit transfer */
            src_xfr_size = 1;
            break;
        case 1:                       /* 16-bit transfer */
            src_xfr_size = 2;
            break;
        case 2:                       /* 32-bit transfer */
            src_xfr_size = 4;
            break;
        case 3:                       /* 64-bit transfer */
            src_xfr_size = 8;
            break;
        case 4:                       /* 16-byte burst transfer */
            src_xfr_size = 16;
            break;
        case 5:                       /* 32-byte burst transfer */
            src_xfr_size = 32;
            break;
        }

        /* convert the destination transfer size into a byte count */
        switch (dma_engine.dsize) {
        case 0:                       /* 8-bit transfer */
            dest_xfr_size = 1;
            break;
        case 1:                       /* 16-bit transfer */
            dest_xfr_size = 2;
            break;
        case 2:                       /* 32-bit transfer */
            dest_xfr_size = 4;
            break;
        case 3:                       /* 64-bit transfer */
            dest_xfr_size = 8;
            break;
        case 4:                       /* 16-byte burst transfer */
            dest_xfr_size = 16;
            break;
        case 5:                       /* 32-byte burst transfer */
            dest_xfr_size = 32;
```

```
        break;
    }

    /* determine the larger of the two transfer sizes, this value reflects */
    /* the number of bytes transferred per read->write sequence. */
    /* number of iterations of the minor loop = nbytes / xfer_size */
    if (dma_engine.ssize < dma_engine.dsize)
        xfr_size = dest_xfer_size;
     else
        xfr_size = src_xfer_size;

    /* process the source address, READ data into the buffer*/

    /* read "xfr_size" bytes from the source */
    /* if the ssize < dsize, do multiple reads to equal the dsize */
    /* if the ssize => dsize, do a single read of source data */
    number_of_source_reads = xfer_size / src_xfer_size;

    for (number_of_source_reads) {
        dma_engine.data = read_from_amba-ahb (dma_engine.saddr, src_xfr_size);

        /* generate the next-state source address */
        /* sum the current saddr with the signed source offset */
        ns_addr = dma_engine.saddr + (int) dma_engine.soff; }

        /* if enabled, apply the power-of-2 modulo to the next-state addr */
        if (dma_engine.smod != 0)
            address_select = (1 << dma_engine.smod) - 1; }
        else
            address_select = 0xffff_ffff;

        dma_engine.saddr = ns_addr           &  address_select
                    | dma_engine.saddr & ~address_select; }
    }


    /* process the destination address, WRITE data from buffer */

    /* write "xfr_size" bytes to the destination */
    /* if the dsize < ssize, do multiple writes to equal the ssize */
    /* if the dsize => ssize, do a single write of dest data */
    number_of_dest_writes = xfer_size / dest_xfer_size;

    for (number_of_dest_writes) {
        write_to_amba-ahb (dma_engine.daddr, dest_xfr_size) = dma_engine.data;

        /* generate the next-state destination address */
        /* sum the current daddr with the signed destination offset */
        ns_addr = dma_engine.daddr + (int) dma_engine.doff;

        /* if enabled, apply the power-of-2 modulo to the next-state dest addr */
        if (dma_engine.dmod != 0)
            address_select = (1 << dma_engine.dmod) - 1;
        else
            address_select = 0xffff_ffff;

        dma_engine.daddr = ns_addr           &  address_select
                    | dma_engine.daddr & ~address_select;
    }
    if (cancel_transfer)
        break;

    /* check for a higher priority channel to service if: */
    /* 1) preemption is enabled */
    /* 2) in fixed arbitration mode */
    /* 3) a higher priority channel is requesting service */
    /* 4) not already servicing a preempting channel */
    if ((DCHPRIn.ecp = 1)  &  fixed_arbitration_mode
        higher_pri_request & ~current_channel_is_preempt)
        service_preempt_channel;
```

```
        /* the bandwidth control field determines when the next read/write occurs */
        if (dma_engine.bwc > 1)
            stall_dma_engine (1 << dma_engine.bwc);

        /* decrement the minor loop byte count */
        dma_engine.nbytes = dma_engine.nbytes - xfr_size;

    }while (dma_engine.nbytes > 0) /* end of minor inner loop */

    dma_engine.citer--;            /* decrement major loop iteration count */

    /* if the major loop is not yet exhausted, update certain TCD values in the RAM */
    if (dma_engine.citer != 0) {
        write_to_local_memory [channel].saddr = dma_engine.saddr;
        write_to_local_memory [channel].daddr = dma_engine.daddr;
        write_to_local_memory [channel].citer = dma_engine.citer;

        /* if minor loop linking is enabled, make the channel link */
        if (dma_engine.citer.e_link)
            TCD[citer.linkch].start = 1;     /* specified channel service req */

        /* check for interrupt assertion if half of the major iterations are done */
        if (dma_engine.int_half && (dma_engine.citer == (dma_engine.biter >> 1)))
            generate_interrupt (channel);

        dma_engine.active = 0;         /* clear the channel busy flag */

     }
     else { /* major loop is complete, dma_engine.citer == 0 */
       /* since the major loop is complete, perform the final address adjustments */

       /* sum the current {src,dst} addresses with "last" adjustment */
       write_to_local_memory [channel].saddr = dma_engine.saddr + dma_engine.slast;
       write_to_local_memory [channel].daddr = dma_engine.daddr + dma_engine.dlast;
       /* restore the major iteration count to the beginning value */
       write_to_local_memory [channel].citer = dma_engine.biter;

       /* check for interrupt assertion at completion of the major iteration */
       if (dma_engine.int_maj)
         generate_interrupt (channel);

        /* check if the ipd_req is to be disabled at completion of the major iteration */
       if (dma_engine.d_req)
           DMAERQ [channel] = 0;

       /* check for a scatter/gather transfer control descriptor */
       if (dma_engine.e_sg) {
         /* load new transfer control descriptor from the address defined by dlast_sga */
           write_to_local_memory [channel] =
                                    read_from_amba-ahb(dma_engine.dlast_sga,32);
       }
       if (dma_engine.major.e_link)
           TCD[major.linkch].start = 1;       /* specified channel service req */

       dma_engine.active = 0;               /* clear the channel busy flag */
       dma_engine.done = 1;                 /* set the channel done flag */

}
else { /* configuration error detected, abort the channel */
       dma_engine.error_status = error_type; /* record the error */
       dma_engine.active = 0;               /* clear the channel busy flag */
       /* check for interrupt assertion on error */
       if (dma_engine.int_err)
           generate_interrupt (channel);
}
```

18.4 Memory map/register definition

The eDMA's programming model is partitioned into two sections, both mapped into the slave bus space: the first region defines a number of registers providing control functions, while the second region corresponds to the local transfer control descriptor memory.

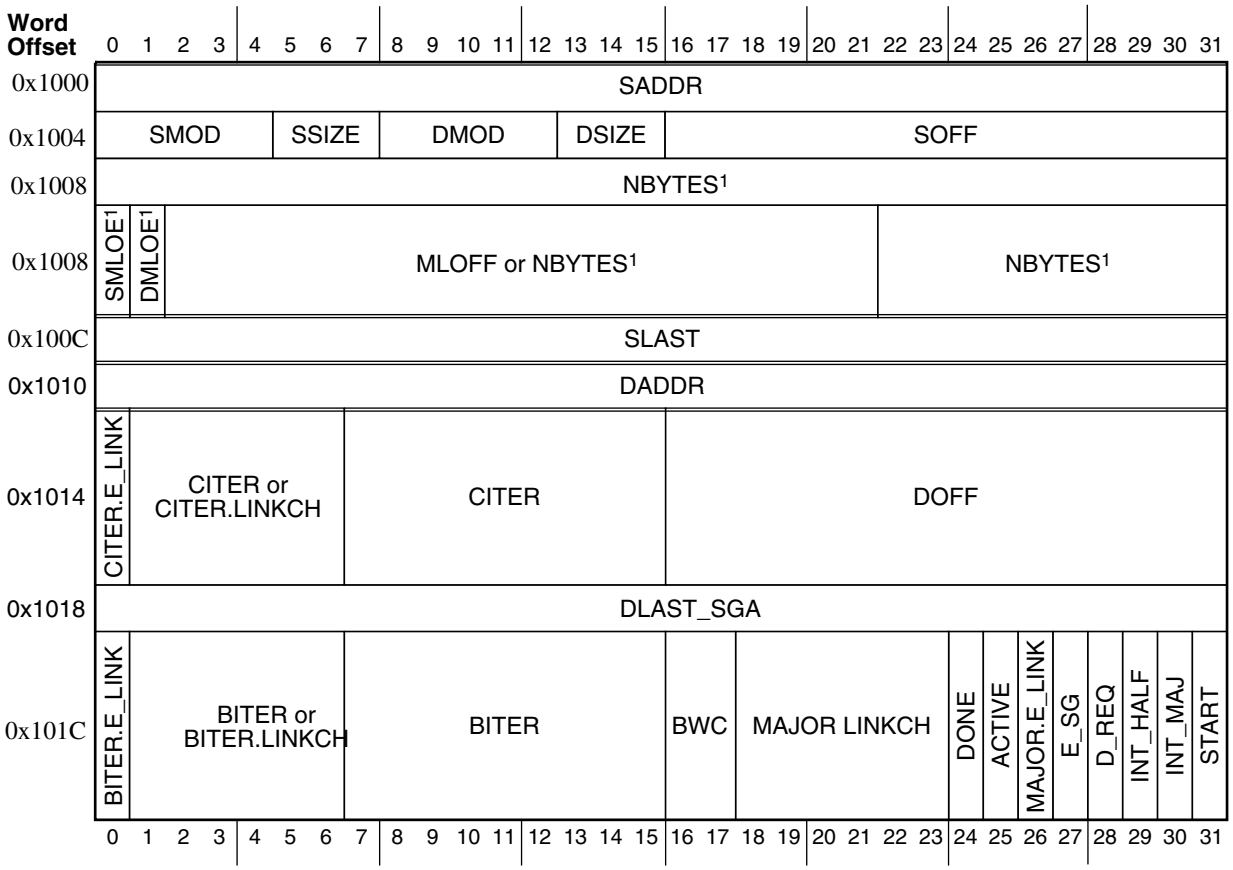
Reading an unimplemented register bit or memory location will return the value of zero. Write the value of zero to unimplemented register bits. Any access to a reserved memory location will result in a bus error. Reserved memory locations are indicated in the memory map. For a 64-channel implementation, reserved memory also includes the high order "H" registers containing channels 63-32 data (for example, the ERQH, EEIH, and INTH registers).

Registers associated with a 64-channel design are implemented as two 32-bit registers, and include an "H" and "L" suffixes, signaling the "high" and "low" portions of the control function. The descriptions in this section define the 64-channel implementation.

The initialization of the TCD memory is performed by the eDMA controller after reset. The initialization runs for 256 eDMA clock cycles. All fields in the TCD memory are initialized to 0. All application read or write accesses to the TCD are delayed until the initialization is finished. Prior to activating a channel, you must initialize its TCD with the appropriate transfer profile.

The TCD structure is shown in the following figure.

- One DMA engine has a number of channels to react to DMA requests
- Each channel has its own TCD



[1] The fields implemented in Word 2 (0x1008) depend on whether EDMA_CR[EMLM] bit is set to 0 or 1.

Figure 18-2. TCD Structure

The following table shows a 32-bit view of the eDMA's memory map.

Table 18-1. eDMA 32-bit memory map

eDMA Offset	Register			
0x0000	eDMA Control Register (CR)			
0x0004	eDMA Error Status (ES)			
0x0008	eDMA Enable Request High (ERQH, Channels 63-32)			
0x000C	eDMA Enable Request Low (ERQL, Channels 31-00)			
0x0010	eDMA Enable Error Interrupt High (EEIH, Channels 63-32)			
0x0014	eDMA Enable Error Interrupt Low (EEIL, Channels 31-00)			
0x0018	eDMA Set Enable Request (SERQ)	eDMA Clear Enable Request (CERQ)	eDMA Set Enable Error Interrupt (SEEI)	eDMA Clear Enable Error Interrupt (CEEI)
0x001C	eDMA Clear Interrupt Request (CINT)	eDMA Clear Error (CERR)	eDMA Set Start Bit (SSRT)	eDMA Clear Done Status Bit (CDNE)
0x0020	eDMA Interrupt Request High (INTH, Channels 63-32)			
0x0024	eDMA Interrupt Request Low (INTL, Channels 31-00)			
0x0028	eDMA Error High (ERRH, Channels 63-32)			

Table continues on the next page...

Table 18-1. eDMA 32-bit memory map (continued)

eDMA Offset	Register			
0x002C	eDMA Error Low (ERRL, Channels 31-00)			
0x0030	eDMA Hardware Request Status High (HRSH, Channels 63-32)			
0x0034	eDMA Hardware Request Status Low (HRSL, Channels 31-00)			
0x0040 – 0x00FC	Reserved			
0x0100	eDMA Channel 0 Priority (DCHPRI0)	eDMA Channel 1 Priority (DCHPRI1)	eDMA Channel 2 Priority (DCHPRI2)	eDMA Channel 3 Priority (DCHPRI3)
0x0104	eDMA Channel 4 Priority (DCHPRI4)	eDMA Channel 5 Priority (DCHPRI5)	eDMA Channel 6 Priority (DCHPRI6)	eDMA Channel 7 Priority (DCHPRI7)
0x0108	eDMA Channel 8 Priority (DCHPRI8)	eDMA Channel 9 Priority (DCHPRI9)	eDMA Channel 10 Priority (DCHPRI10)	eDMA Channel 11 Priority (DCHPRI11)
0x010C	eDMA Channel 12 Priority (DCHPRI12)	eDMA Channel 13 Priority (DCHPRI13)	eDMA Channel 14 Priority (DCHPRI14)	eDMA Channel 15 Priority (DCHPRI15)
0x0110	eDMA Channel 16 Priority (DCHPRI16)	eDMA Channel 17 Priority (DCHPRI17)	eDMA Channel 18 Priority (DCHPRI18)	eDMA Channel 19 Priority (DCHPRI18)
0x0114	eDMA Channel 20 Priority (DCHPRI20)	eDMA Channel 21 Priority (DCHPRI21)	eDMA Channel 22 Priority (DCHPRI21)	eDMA Channel 23 Priority (DCHPRI23)
0x0118	eDMA Channel 24 Priority (DCHPRI23)	eDMA Channel 25 Priority (DCHPRI25)	eDMA Channel 26 Priority (DCHPRI25)	eDMA Channel 27 Priority (DCHPRI27)
0x011C	eDMA Channel 28 Priority (DCHPRI28)	eDMA Channel 29 Priority (DCHPRI29)	eDMA Channel 30 Priority (DCHPRI30)	eDMA Channel 31 Priority (DCHPRI30)
0x0120	eDMA Channel 32 Priority (DCHPRI32)	eDMA Channel 33 Priority (DCHPRI32)	eDMA Channel 34 Priority (DCHPRI34)	eDMA Channel 35 Priority (DCHPRI34)
0x0124	eDMA Channel 36 Priority (DCHPRI36)	eDMA Channel 37 Priority (DCHPRI37)	eDMA Channel 38 Priority (DCHPRI38)	eDMA Channel 39 Priority (DCHPRI39)
0x0128	eDMA Channel 40 Priority (DCHPRI40)	eDMA Channel 41 Priority (DCHPRI41)	eDMA Channel 42 Priority (DCHPRI42)	eDMA Channel 43 Priority (DCHPRI42)
0x012C	eDMA Channel 44 Priority (DCHPRI44)	eDMA Channel 45 Priority (DCHPRI45)	eDMA Channel 46 Priority (DCHPRI46)	eDMA Channel 47 Priority (DCHPRI47)
0x0130	eDMA Channel 48 Priority (DCHPRI48)	eDMA Channel 49 Priority (DCHPRI49)	eDMA Channel 50 Priority (DCHPRI50)	eDMA Channel 51 Priority (DCHPRI51)
0x0134	eDMA Channel 52 Priority (DCHPRI52)	eDMA Channel 53 Priority (DCHPRI52)	eDMA Channel 54 Priority (DCHPRI54)	eDMA Channel 55 Priority (DCHPRI55)
0x0138	eDMA Channel 56 Priority (DCHPRI56)	eDMA Channel 57 Priority (DCHPRI56)	eDMA Channel 58 Priority (DCHPRI58)	eDMA Channel 59 Priority (DCHPRI59)
0x013C	eDMA Channel 60 Priority (DCHPRI60)	eDMA Channel 61 Priority (DCHPRI60)	eDMA Channel 62 Priority (DCHPRI62)	eDMA Channel 63 Priority (DCHPRI62)
0x0140 – 0x017C	Channel n Master ID Register (DCHMID)			
0x0180 – 0x0FFC	Reserved			
0x1000 – 0x11FC	TCD00 – TCD15			
0x1200 – 0x13FC	TCD16 – TCD31			
0x1400 – 0x15FC	TCD32 – TCD47			
0x1600 – 0x17FC	TCD48 – TCD63			

18.5 Memory Map and Registers

DMA memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control Register (DMA_CR)	32	R/W	0000_E400h	18.5.1/520
4	Error Status Register (DMA_ES)	32	R	0000_0000h	18.5.2/523
8	Enable Request Register High (DMA_ERQH)	32	R/W	0000_0000h	18.5.3/526
C	Enable Request Register Low (DMA_ERQL)	32	R/W	0000_0000h	18.5.4/530
10	Enable Error Interrupt Register High (DMA_EEIH)	32	R/W	0000_0000h	18.5.5/533
14	Enable Error Interrupt Register Low (DMA_EEIL)	32	R/W	0000_0000h	18.5.6/537
18	Set Enable Request Register (DMA_SERQ)	8	W (always reads 0)	00h	18.5.7/540
19	Clear Enable Request Register (DMA_CERQ)	8	W (always reads 0)	00h	18.5.8/541
1A	Set Enable Error Interrupt Register (DMA_SEEI)	8	W (always reads 0)	00h	18.5.9/542
1B	Clear Enable Error Interrupt Register (DMA_CEEI)	8	W (always reads 0)	00h	18.5.10/543
1C	Clear Interrupt Request Register (DMA_CINT)	8	W (always reads 0)	00h	18.5.11/544
1D	Clear Error Register (DMA_CERR)	8	W (always reads 0)	00h	18.5.12/545
1E	Set START Bit Register (DMA_SSRT)	8	W (always reads 0)	00h	18.5.13/546
1F	Clear DONE Status Bit Register (DMA_CDNE)	8	W (always reads 0)	00h	18.5.14/547
20	Interrupt Request Register High (DMA_INTH)	32	R/W	0000_0000h	18.5.15/547
24	Interrupt Request Register Low (DMA_INTL)	32	R/W	0000_0000h	18.5.16/551
28	Enable Error Interrupt Register High (DMA_ERRH)	32	R/W	0000_0000h	18.5.17/555
2C	Enable Error Interrupt RegisterLow (DMA_ERRL)	32	R/W	0000_0000h	18.5.18/559
30	Hardware Request Status Register High (DMA_HRSH)	32	R/W	0000_0000h	18.5.19/563
34	Hardware Request Status Register Low (DMA_HRSL)	32	R/W	0000_0000h	18.5.20/566
100	Channel n Priority Register (DMA_DCHPRI0)	8	R/W	See section	18.5.21/569
101	Channel n Priority Register (DMA_DCHPRI1)	8	R/W	See section	18.5.21/569
102	Channel n Priority Register (DMA_DCHPRI2)	8	R/W	See section	18.5.21/569

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
103	Channel n Priority Register (DMA_DCHPRI3)	8	R/W	See section	18.5.21/569
104	Channel n Priority Register (DMA_DCHPRI4)	8	R/W	See section	18.5.21/569
105	Channel n Priority Register (DMA_DCHPRI5)	8	R/W	See section	18.5.21/569
106	Channel n Priority Register (DMA_DCHPRI6)	8	R/W	See section	18.5.21/569
107	Channel n Priority Register (DMA_DCHPRI7)	8	R/W	See section	18.5.21/569
108	Channel n Priority Register (DMA_DCHPRI8)	8	R/W	See section	18.5.21/569
109	Channel n Priority Register (DMA_DCHPRI9)	8	R/W	See section	18.5.21/569
10A	Channel n Priority Register (DMA_DCHPRI10)	8	R/W	See section	18.5.21/569
10B	Channel n Priority Register (DMA_DCHPRI11)	8	R/W	See section	18.5.21/569
10C	Channel n Priority Register (DMA_DCHPRI12)	8	R/W	See section	18.5.21/569
10D	Channel n Priority Register (DMA_DCHPRI13)	8	R/W	See section	18.5.21/569
10E	Channel n Priority Register (DMA_DCHPRI14)	8	R/W	See section	18.5.21/569
10F	Channel n Priority Register (DMA_DCHPRI15)	8	R/W	See section	18.5.21/569
110	Channel n Priority Register (DMA_DCHPRI16)	8	R/W	See section	18.5.21/569
111	Channel n Priority Register (DMA_DCHPRI17)	8	R/W	See section	18.5.21/569
112	Channel n Priority Register (DMA_DCHPRI18)	8	R/W	See section	18.5.21/569
113	Channel n Priority Register (DMA_DCHPRI19)	8	R/W	See section	18.5.21/569
114	Channel n Priority Register (DMA_DCHPRI20)	8	R/W	See section	18.5.21/569
115	Channel n Priority Register (DMA_DCHPRI21)	8	R/W	See section	18.5.21/569
116	Channel n Priority Register (DMA_DCHPRI22)	8	R/W	See section	18.5.21/569
117	Channel n Priority Register (DMA_DCHPRI23)	8	R/W	See section	18.5.21/569
118	Channel n Priority Register (DMA_DCHPRI24)	8	R/W	See section	18.5.21/569
119	Channel n Priority Register (DMA_DCHPRI25)	8	R/W	See section	18.5.21/569
11A	Channel n Priority Register (DMA_DCHPRI26)	8	R/W	See section	18.5.21/569
11B	Channel n Priority Register (DMA_DCHPRI27)	8	R/W	See section	18.5.21/569
11C	Channel n Priority Register (DMA_DCHPRI28)	8	R/W	See section	18.5.21/569
11D	Channel n Priority Register (DMA_DCHPRI29)	8	R/W	See section	18.5.21/569
11E	Channel n Priority Register (DMA_DCHPRI30)	8	R/W	See section	18.5.21/569
11F	Channel n Priority Register (DMA_DCHPRI31)	8	R/W	See section	18.5.21/569
120	Channel n Priority Register (DMA_DCHPRI32)	8	R/W	See section	18.5.21/569
121	Channel n Priority Register (DMA_DCHPRI33)	8	R/W	See section	18.5.21/569
122	Channel n Priority Register (DMA_DCHPRI34)	8	R/W	See section	18.5.21/569
123	Channel n Priority Register (DMA_DCHPRI35)	8	R/W	See section	18.5.21/569
124	Channel n Priority Register (DMA_DCHPRI36)	8	R/W	See section	18.5.21/569
125	Channel n Priority Register (DMA_DCHPRI37)	8	R/W	See section	18.5.21/569
126	Channel n Priority Register (DMA_DCHPRI38)	8	R/W	See section	18.5.21/569
127	Channel n Priority Register (DMA_DCHPRI39)	8	R/W	See section	18.5.21/569
128	Channel n Priority Register (DMA_DCHPRI40)	8	R/W	See section	18.5.21/569
129	Channel n Priority Register (DMA_DCHPRI41)	8	R/W	See section	18.5.21/569

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12A	Channel n Priority Register (DMA_DCHPRI42)	8	R/W	See section	18.5.21/569
12B	Channel n Priority Register (DMA_DCHPRI43)	8	R/W	See section	18.5.21/569
12C	Channel n Priority Register (DMA_DCHPRI44)	8	R/W	See section	18.5.21/569
12D	Channel n Priority Register (DMA_DCHPRI45)	8	R/W	See section	18.5.21/569
12E	Channel n Priority Register (DMA_DCHPRI46)	8	R/W	See section	18.5.21/569
12F	Channel n Priority Register (DMA_DCHPRI47)	8	R/W	See section	18.5.21/569
130	Channel n Priority Register (DMA_DCHPRI48)	8	R/W	See section	18.5.21/569
131	Channel n Priority Register (DMA_DCHPRI49)	8	R/W	See section	18.5.21/569
132	Channel n Priority Register (DMA_DCHPRI50)	8	R/W	See section	18.5.21/569
133	Channel n Priority Register (DMA_DCHPRI51)	8	R/W	See section	18.5.21/569
134	Channel n Priority Register (DMA_DCHPRI52)	8	R/W	See section	18.5.21/569
135	Channel n Priority Register (DMA_DCHPRI53)	8	R/W	See section	18.5.21/569
136	Channel n Priority Register (DMA_DCHPRI54)	8	R/W	See section	18.5.21/569
137	Channel n Priority Register (DMA_DCHPRI55)	8	R/W	See section	18.5.21/569
138	Channel n Priority Register (DMA_DCHPRI56)	8	R/W	See section	18.5.21/569
139	Channel n Priority Register (DMA_DCHPRI57)	8	R/W	See section	18.5.21/569
13A	Channel n Priority Register (DMA_DCHPRI58)	8	R/W	See section	18.5.21/569
13B	Channel n Priority Register (DMA_DCHPRI59)	8	R/W	See section	18.5.21/569
13C	Channel n Priority Register (DMA_DCHPRI60)	8	R/W	See section	18.5.21/569
13D	Channel n Priority Register (DMA_DCHPRI61)	8	R/W	See section	18.5.21/569
13E	Channel n Priority Register (DMA_DCHPRI62)	8	R/W	See section	18.5.21/569
13F	Channel n Priority Register (DMA_DCHPRI63)	8	R/W	See section	18.5.21/569
140	Channel n Master ID Register (DMA_DCHMID0)	8	R/W	00h	18.5.22/571
141	Channel n Master ID Register (DMA_DCHMID1)	8	R/W	00h	18.5.22/571
142	Channel n Master ID Register (DMA_DCHMID2)	8	R/W	00h	18.5.22/571
143	Channel n Master ID Register (DMA_DCHMID3)	8	R/W	00h	18.5.22/571
144	Channel n Master ID Register (DMA_DCHMID4)	8	R/W	00h	18.5.22/571
145	Channel n Master ID Register (DMA_DCHMID5)	8	R/W	00h	18.5.22/571
146	Channel n Master ID Register (DMA_DCHMID6)	8	R/W	00h	18.5.22/571
147	Channel n Master ID Register (DMA_DCHMID7)	8	R/W	00h	18.5.22/571
148	Channel n Master ID Register (DMA_DCHMID8)	8	R/W	00h	18.5.22/571
149	Channel n Master ID Register (DMA_DCHMID9)	8	R/W	00h	18.5.22/571
14A	Channel n Master ID Register (DMA_DCHMID10)	8	R/W	00h	18.5.22/571
14B	Channel n Master ID Register (DMA_DCHMID11)	8	R/W	00h	18.5.22/571
14C	Channel n Master ID Register (DMA_DCHMID12)	8	R/W	00h	18.5.22/571
14D	Channel n Master ID Register (DMA_DCHMID13)	8	R/W	00h	18.5.22/571
14E	Channel n Master ID Register (DMA_DCHMID14)	8	R/W	00h	18.5.22/571
14F	Channel n Master ID Register (DMA_DCHMID15)	8	R/W	00h	18.5.22/571
150	Channel n Master ID Register (DMA_DCHMID16)	8	R/W	00h	18.5.22/571

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
151	Channel n Master ID Register (DMA_DCHMID17)	8	R/W	00h	18.5.22/571
152	Channel n Master ID Register (DMA_DCHMID18)	8	R/W	00h	18.5.22/571
153	Channel n Master ID Register (DMA_DCHMID19)	8	R/W	00h	18.5.22/571
154	Channel n Master ID Register (DMA_DCHMID20)	8	R/W	00h	18.5.22/571
155	Channel n Master ID Register (DMA_DCHMID21)	8	R/W	00h	18.5.22/571
156	Channel n Master ID Register (DMA_DCHMID22)	8	R/W	00h	18.5.22/571
157	Channel n Master ID Register (DMA_DCHMID23)	8	R/W	00h	18.5.22/571
158	Channel n Master ID Register (DMA_DCHMID24)	8	R/W	00h	18.5.22/571
159	Channel n Master ID Register (DMA_DCHMID25)	8	R/W	00h	18.5.22/571
15A	Channel n Master ID Register (DMA_DCHMID26)	8	R/W	00h	18.5.22/571
15B	Channel n Master ID Register (DMA_DCHMID27)	8	R/W	00h	18.5.22/571
15C	Channel n Master ID Register (DMA_DCHMID28)	8	R/W	00h	18.5.22/571
15D	Channel n Master ID Register (DMA_DCHMID29)	8	R/W	00h	18.5.22/571
15E	Channel n Master ID Register (DMA_DCHMID30)	8	R/W	00h	18.5.22/571
15F	Channel n Master ID Register (DMA_DCHMID31)	8	R/W	00h	18.5.22/571
160	Channel n Master ID Register (DMA_DCHMID32)	8	R/W	00h	18.5.22/571
161	Channel n Master ID Register (DMA_DCHMID33)	8	R/W	00h	18.5.22/571
162	Channel n Master ID Register (DMA_DCHMID34)	8	R/W	00h	18.5.22/571
163	Channel n Master ID Register (DMA_DCHMID35)	8	R/W	00h	18.5.22/571
164	Channel n Master ID Register (DMA_DCHMID36)	8	R/W	00h	18.5.22/571
165	Channel n Master ID Register (DMA_DCHMID37)	8	R/W	00h	18.5.22/571
166	Channel n Master ID Register (DMA_DCHMID38)	8	R/W	00h	18.5.22/571
167	Channel n Master ID Register (DMA_DCHMID39)	8	R/W	00h	18.5.22/571
168	Channel n Master ID Register (DMA_DCHMID40)	8	R/W	00h	18.5.22/571
169	Channel n Master ID Register (DMA_DCHMID41)	8	R/W	00h	18.5.22/571
16A	Channel n Master ID Register (DMA_DCHMID42)	8	R/W	00h	18.5.22/571
16B	Channel n Master ID Register (DMA_DCHMID43)	8	R/W	00h	18.5.22/571
16C	Channel n Master ID Register (DMA_DCHMID44)	8	R/W	00h	18.5.22/571
16D	Channel n Master ID Register (DMA_DCHMID45)	8	R/W	00h	18.5.22/571
16E	Channel n Master ID Register (DMA_DCHMID46)	8	R/W	00h	18.5.22/571
16F	Channel n Master ID Register (DMA_DCHMID47)	8	R/W	00h	18.5.22/571
170	Channel n Master ID Register (DMA_DCHMID48)	8	R/W	00h	18.5.22/571
171	Channel n Master ID Register (DMA_DCHMID49)	8	R/W	00h	18.5.22/571
172	Channel n Master ID Register (DMA_DCHMID50)	8	R/W	00h	18.5.22/571
173	Channel n Master ID Register (DMA_DCHMID51)	8	R/W	00h	18.5.22/571
174	Channel n Master ID Register (DMA_DCHMID52)	8	R/W	00h	18.5.22/571
175	Channel n Master ID Register (DMA_DCHMID53)	8	R/W	00h	18.5.22/571
176	Channel n Master ID Register (DMA_DCHMID54)	8	R/W	00h	18.5.22/571
177	Channel n Master ID Register (DMA_DCHMID55)	8	R/W	00h	18.5.22/571

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
178	Channel n Master ID Register (DMA_DCHMID56)	8	R/W	00h	18.5.22/571
179	Channel n Master ID Register (DMA_DCHMID57)	8	R/W	00h	18.5.22/571
17A	Channel n Master ID Register (DMA_DCHMID58)	8	R/W	00h	18.5.22/571
17B	Channel n Master ID Register (DMA_DCHMID59)	8	R/W	00h	18.5.22/571
17C	Channel n Master ID Register (DMA_DCHMID60)	8	R/W	00h	18.5.22/571
17D	Channel n Master ID Register (DMA_DCHMID61)	8	R/W	00h	18.5.22/571
17E	Channel n Master ID Register (DMA_DCHMID62)	8	R/W	00h	18.5.22/571
17F	Channel n Master ID Register (DMA_DCHMID63)	8	R/W	00h	18.5.22/571
1000	TCD Source Address (DMA_TCD0_WORD_0)	32	R/W	Undefined	18.5.23/572
1004	TCD Transfer Attributes (DMA_TCD0_WORD_1)	32	R/W	Undefined	18.5.24/573
1008	TCD Number of bytes to be transferred (DMA_TCD0_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/574
1008	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD0_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/575
100C	TCD Last Source Address Adjustment (DMA_TCD0_WORD_3)	32	R/W	Undefined	18.5.27/576
1010	TCD Destination Address (DMA_TCD0_WORD_4)	32	R/W	Undefined	18.5.28/576
1014	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD0_WORD_5)	32	R/W	Undefined	18.5.29/577
1018	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD0_WORD_6)	32	R/W	Undefined	18.5.30/578
101C	TCD Control and Status (DMA_TCD0_WORD_7)	32	R/W	Undefined	18.5.31/579
1020	TCD Source Address (DMA_TCD1_WORD_0)	32	R/W	Undefined	18.5.23/572
1024	TCD Transfer Attributes (DMA_TCD1_WORD_1)	32	R/W	Undefined	18.5.24/573
1028	TCD Number of bytes to be transferred (DMA_TCD1_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/574
1028	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD1_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/575
102C	TCD Last Source Address Adjustment (DMA_TCD1_WORD_3)	32	R/W	Undefined	18.5.27/576
1030	TCD Destination Address (DMA_TCD1_WORD_4)	32	R/W	Undefined	18.5.28/576
1034	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD1_WORD_5)	32	R/W	Undefined	18.5.29/577
1038	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD1_WORD_6)	32	R/W	Undefined	18.5.30/578
103C	TCD Control and Status (DMA_TCD1_WORD_7)	32	R/W	Undefined	18.5.31/579
1040	TCD Source Address (DMA_TCD2_WORD_0)	32	R/W	Undefined	18.5.23/572
1044	TCD Transfer Attributes (DMA_TCD2_WORD_1)	32	R/W	Undefined	18.5.24/573
1048	TCD Number of bytes to be transferred (DMA_TCD2_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/574
1048	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD2_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/575

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
104C	TCD Last Source Address Adjustment (DMA_TCD2_WORD_3)	32	R/W	Undefined	18.5.27/576
1050	TCD Destination Address (DMA_TCD2_WORD_4)	32	R/W	Undefined	18.5.28/576
1054	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD2_WORD_5)	32	R/W	Undefined	18.5.29/577
1058	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD2_WORD_6)	32	R/W	Undefined	18.5.30/578
105C	TCD Control and Status (DMA_TCD2_WORD_7)	32	R/W	Undefined	18.5.31/579
1060	TCD Source Address (DMA_TCD3_WORD_0)	32	R/W	Undefined	18.5.23/572
1064	TCD Transfer Attributes (DMA_TCD3_WORD_1)	32	R/W	Undefined	18.5.24/573
1068	TCD Number of bytes to be transferred (DMA_TCD3_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/574
1068	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD3_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/575
106C	TCD Last Source Address Adjustment (DMA_TCD3_WORD_3)	32	R/W	Undefined	18.5.27/576
1070	TCD Destination Address (DMA_TCD3_WORD_4)	32	R/W	Undefined	18.5.28/576
1074	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD3_WORD_5)	32	R/W	Undefined	18.5.29/577
1078	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD3_WORD_6)	32	R/W	Undefined	18.5.30/578
107C	TCD Control and Status (DMA_TCD3_WORD_7)	32	R/W	Undefined	18.5.31/579
1080	TCD Source Address (DMA_TCD4_WORD_0)	32	R/W	Undefined	18.5.23/572
1084	TCD Transfer Attributes (DMA_TCD4_WORD_1)	32	R/W	Undefined	18.5.24/573
1088	TCD Number of bytes to be transferred (DMA_TCD4_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/574
1088	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD4_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/575
108C	TCD Last Source Address Adjustment (DMA_TCD4_WORD_3)	32	R/W	Undefined	18.5.27/576
1090	TCD Destination Address (DMA_TCD4_WORD_4)	32	R/W	Undefined	18.5.28/576
1094	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD4_WORD_5)	32	R/W	Undefined	18.5.29/577
1098	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD4_WORD_6)	32	R/W	Undefined	18.5.30/578
109C	TCD Control and Status (DMA_TCD4_WORD_7)	32	R/W	Undefined	18.5.31/579
10A0	TCD Source Address (DMA_TCD5_WORD_0)	32	R/W	Undefined	18.5.23/572
10A4	TCD Transfer Attributes (DMA_TCD5_WORD_1)	32	R/W	Undefined	18.5.24/573
10A8	TCD Number of bytes to be transferred (DMA_TCD5_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/574
10A8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD5_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/575

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10AC	TCD Last Source Address Adjustment (DMA_TCD5_WORD_3)	32	R/W	Undefined	18.5.27/576
10B0	TCD Destination Address (DMA_TCD5_WORD_4)	32	R/W	Undefined	18.5.28/576
10B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD5_WORD_5)	32	R/W	Undefined	18.5.29/577
10B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD5_WORD_6)	32	R/W	Undefined	18.5.30/578
10BC	TCD Control and Status (DMA_TCD5_WORD_7)	32	R/W	Undefined	18.5.31/579
10C0	TCD Source Address (DMA_TCD6_WORD_0)	32	R/W	Undefined	18.5.23/572
10C4	TCD Transfer Attributes (DMA_TCD6_WORD_1)	32	R/W	Undefined	18.5.24/573
10C8	TCD Number of bytes to be transferred (DMA_TCD6_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/574
10C8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD6_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/575
10CC	TCD Last Source Address Adjustment (DMA_TCD6_WORD_3)	32	R/W	Undefined	18.5.27/576
10D0	TCD Destination Address (DMA_TCD6_WORD_4)	32	R/W	Undefined	18.5.28/576
10D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD6_WORD_5)	32	R/W	Undefined	18.5.29/577
10D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD6_WORD_6)	32	R/W	Undefined	18.5.30/578
10DC	TCD Control and Status (DMA_TCD6_WORD_7)	32	R/W	Undefined	18.5.31/579
10E0	TCD Source Address (DMA_TCD7_WORD_0)	32	R/W	Undefined	18.5.23/572
10E4	TCD Transfer Attributes (DMA_TCD7_WORD_1)	32	R/W	Undefined	18.5.24/573
10E8	TCD Number of bytes to be transferred (DMA_TCD7_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/574
10E8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD7_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/575
10EC	TCD Last Source Address Adjustment (DMA_TCD7_WORD_3)	32	R/W	Undefined	18.5.27/576
10F0	TCD Destination Address (DMA_TCD7_WORD_4)	32	R/W	Undefined	18.5.28/576
10F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD7_WORD_5)	32	R/W	Undefined	18.5.29/577
10F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD7_WORD_6)	32	R/W	Undefined	18.5.30/578
10FC	TCD Control and Status (DMA_TCD7_WORD_7)	32	R/W	Undefined	18.5.31/579
1100	TCD Source Address (DMA_TCD8_WORD_0)	32	R/W	Undefined	18.5.23/572
1104	TCD Transfer Attributes (DMA_TCD8_WORD_1)	32	R/W	Undefined	18.5.24/573
1108	TCD Number of bytes to be transferred (DMA_TCD8_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/574
1108	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD8_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/575

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
110C	TCD Last Source Address Adjustment (DMA_TCD8_WORD_3)	32	R/W	Undefined	18.5.27/576
1110	TCD Destination Address (DMA_TCD8_WORD_4)	32	R/W	Undefined	18.5.28/576
1114	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD8_WORD_5)	32	R/W	Undefined	18.5.29/577
1118	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD8_WORD_6)	32	R/W	Undefined	18.5.30/578
111C	TCD Control and Status (DMA_TCD8_WORD_7)	32	R/W	Undefined	18.5.31/579
1120	TCD Source Address (DMA_TCD9_WORD_0)	32	R/W	Undefined	18.5.23/572
1124	TCD Transfer Attributes (DMA_TCD9_WORD_1)	32	R/W	Undefined	18.5.24/573
1128	TCD Number of bytes to be transferred (DMA_TCD9_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/574
1128	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD9_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/575
112C	TCD Last Source Address Adjustment (DMA_TCD9_WORD_3)	32	R/W	Undefined	18.5.27/576
1130	TCD Destination Address (DMA_TCD9_WORD_4)	32	R/W	Undefined	18.5.28/576
1134	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD9_WORD_5)	32	R/W	Undefined	18.5.29/577
1138	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD9_WORD_6)	32	R/W	Undefined	18.5.30/578
113C	TCD Control and Status (DMA_TCD9_WORD_7)	32	R/W	Undefined	18.5.31/579
1140	TCD Source Address (DMA_TCD10_WORD_0)	32	R/W	Undefined	18.5.23/572
1144	TCD Transfer Attributes (DMA_TCD10_WORD_1)	32	R/W	Undefined	18.5.24/573
1148	TCD Number of bytes to be transferred (DMA_TCD10_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/574
1148	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD10_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/575
114C	TCD Last Source Address Adjustment (DMA_TCD10_WORD_3)	32	R/W	Undefined	18.5.27/576
1150	TCD Destination Address (DMA_TCD10_WORD_4)	32	R/W	Undefined	18.5.28/576
1154	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD10_WORD_5)	32	R/W	Undefined	18.5.29/577
1158	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD10_WORD_6)	32	R/W	Undefined	18.5.30/578
115C	TCD Control and Status (DMA_TCD10_WORD_7)	32	R/W	Undefined	18.5.31/579
1160	TCD Source Address (DMA_TCD11_WORD_0)	32	R/W	Undefined	18.5.23/572
1164	TCD Transfer Attributes (DMA_TCD11_WORD_1)	32	R/W	Undefined	18.5.24/573
1168	TCD Number of bytes to be transferred (DMA_TCD11_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/574
1168	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD11_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/575

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
116C	TCD Last Source Address Adjustment (DMA_TCD11_WORD_3)	32	R/W	Undefined	18.5.27/576
1170	TCD Destination Address (DMA_TCD11_WORD_4)	32	R/W	Undefined	18.5.28/576
1174	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD11_WORD_5)	32	R/W	Undefined	18.5.29/577
1178	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD11_WORD_6)	32	R/W	Undefined	18.5.30/578
117C	TCD Control and Status (DMA_TCD11_WORD_7)	32	R/W	Undefined	18.5.31/579
1180	TCD Source Address (DMA_TCD12_WORD_0)	32	R/W	Undefined	18.5.23/572
1184	TCD Transfer Attributes (DMA_TCD12_WORD_1)	32	R/W	Undefined	18.5.24/573
1188	TCD Number of bytes to be transferred (DMA_TCD12_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/574
1188	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD12_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/575
118C	TCD Last Source Address Adjustment (DMA_TCD12_WORD_3)	32	R/W	Undefined	18.5.27/576
1190	TCD Destination Address (DMA_TCD12_WORD_4)	32	R/W	Undefined	18.5.28/576
1194	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD12_WORD_5)	32	R/W	Undefined	18.5.29/577
1198	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD12_WORD_6)	32	R/W	Undefined	18.5.30/578
119C	TCD Control and Status (DMA_TCD12_WORD_7)	32	R/W	Undefined	18.5.31/579
11A0	TCD Source Address (DMA_TCD13_WORD_0)	32	R/W	Undefined	18.5.23/572
11A4	TCD Transfer Attributes (DMA_TCD13_WORD_1)	32	R/W	Undefined	18.5.24/573
11A8	TCD Number of bytes to be transferred (DMA_TCD13_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/574
11A8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD13_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/575
11AC	TCD Last Source Address Adjustment (DMA_TCD13_WORD_3)	32	R/W	Undefined	18.5.27/576
11B0	TCD Destination Address (DMA_TCD13_WORD_4)	32	R/W	Undefined	18.5.28/576
11B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD13_WORD_5)	32	R/W	Undefined	18.5.29/577
11B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD13_WORD_6)	32	R/W	Undefined	18.5.30/578
11BC	TCD Control and Status (DMA_TCD13_WORD_7)	32	R/W	Undefined	18.5.31/579
11C0	TCD Source Address (DMA_TCD14_WORD_0)	32	R/W	Undefined	18.5.23/572
11C4	TCD Transfer Attributes (DMA_TCD14_WORD_1)	32	R/W	Undefined	18.5.24/573
11C8	TCD Number of bytes to be transferred (DMA_TCD14_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/574
11C8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD14_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/575

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11CC	TCD Last Source Address Adjustment (DMA_TCD14_WORD_3)	32	R/W	Undefined	18.5.27/ 576
11D0	TCD Destination Address (DMA_TCD14_WORD_4)	32	R/W	Undefined	18.5.28/ 576
11D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD14_WORD_5)	32	R/W	Undefined	18.5.29/ 577
11D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD14_WORD_6)	32	R/W	Undefined	18.5.30/ 578
11DC	TCD Control and Status (DMA_TCD14_WORD_7)	32	R/W	Undefined	18.5.31/ 579
11E0	TCD Source Address (DMA_TCD15_WORD_0)	32	R/W	Undefined	18.5.23/ 572
11E4	TCD Transfer Attributes (DMA_TCD15_WORD_1)	32	R/W	Undefined	18.5.24/ 573
11E8	TCD Number of bytes to be transferred (DMA_TCD15_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
11E8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD15_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
11EC	TCD Last Source Address Adjustment (DMA_TCD15_WORD_3)	32	R/W	Undefined	18.5.27/ 576
11F0	TCD Destination Address (DMA_TCD15_WORD_4)	32	R/W	Undefined	18.5.28/ 576
11F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD15_WORD_5)	32	R/W	Undefined	18.5.29/ 577
11F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD15_WORD_6)	32	R/W	Undefined	18.5.30/ 578
11FC	TCD Control and Status (DMA_TCD15_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1200	TCD Source Address (DMA_TCD16_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1204	TCD Transfer Attributes (DMA_TCD16_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1208	TCD Number of bytes to be transferred (DMA_TCD16_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1208	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD16_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
120C	TCD Last Source Address Adjustment (DMA_TCD16_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1210	TCD Destination Address (DMA_TCD16_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1214	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD16_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1218	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD16_WORD_6)	32	R/W	Undefined	18.5.30/ 578

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
121C	TCD Control and Status (DMA_TCD16_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1220	TCD Source Address (DMA_TCD17_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1224	TCD Transfer Attributes (DMA_TCD17_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1228	TCD Number of bytes to be transferred (DMA_TCD17_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1228	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD17_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
122C	TCD Last Source Address Adjustment (DMA_TCD17_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1230	TCD Destination Address (DMA_TCD17_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1234	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD17_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1238	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD17_WORD_6)	32	R/W	Undefined	18.5.30/ 578
123C	TCD Control and Status (DMA_TCD17_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1240	TCD Source Address (DMA_TCD18_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1244	TCD Transfer Attributes (DMA_TCD18_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1248	TCD Number of bytes to be transferred (DMA_TCD18_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1248	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD18_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
124C	TCD Last Source Address Adjustment (DMA_TCD18_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1250	TCD Destination Address (DMA_TCD18_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1254	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD18_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1258	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD18_WORD_6)	32	R/W	Undefined	18.5.30/ 578
125C	TCD Control and Status (DMA_TCD18_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1260	TCD Source Address (DMA_TCD19_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1264	TCD Transfer Attributes (DMA_TCD19_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1268	TCD Number of bytes to be transferred (DMA_TCD19_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1268	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD19_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
126C	TCD Last Source Address Adjustment (DMA_TCD19_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1270	TCD Destination Address (DMA_TCD19_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1274	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD19_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1278	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD19_WORD_6)	32	R/W	Undefined	18.5.30/ 578
127C	TCD Control and Status (DMA_TCD19_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1280	TCD Source Address (DMA_TCD20_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1284	TCD Transfer Attributes (DMA_TCD20_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1288	TCD Number of bytes to be transferred (DMA_TCD20_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1288	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD20_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
128C	TCD Last Source Address Adjustment (DMA_TCD20_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1290	TCD Destination Address (DMA_TCD20_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1294	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD20_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1298	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD20_WORD_6)	32	R/W	Undefined	18.5.30/ 578
129C	TCD Control and Status (DMA_TCD20_WORD_7)	32	R/W	Undefined	18.5.31/ 579
12A0	TCD Source Address (DMA_TCD21_WORD_0)	32	R/W	Undefined	18.5.23/ 572
12A4	TCD Transfer Attributes (DMA_TCD21_WORD_1)	32	R/W	Undefined	18.5.24/ 573
12A8	TCD Number of bytes to be transferred (DMA_TCD21_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
12A8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD21_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
12AC	TCD Last Source Address Adjustment (DMA_TCD21_WORD_3)	32	R/W	Undefined	18.5.27/ 576
12B0	TCD Destination Address (DMA_TCD21_WORD_4)	32	R/W	Undefined	18.5.28/ 576
12B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD21_WORD_5)	32	R/W	Undefined	18.5.29/ 577

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD21_WORD_6)	32	R/W	Undefined	18.5.30/ 578
12BC	TCD Control and Status (DMA_TCD21_WORD_7)	32	R/W	Undefined	18.5.31/ 579
12C0	TCD Source Address (DMA_TCD22_WORD_0)	32	R/W	Undefined	18.5.23/ 572
12C4	TCD Transfer Attributes (DMA_TCD22_WORD_1)	32	R/W	Undefined	18.5.24/ 573
12C8	TCD Number of bytes to be transferred (DMA_TCD22_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
12C8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD22_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
12CC	TCD Last Source Address Adjustment (DMA_TCD22_WORD_3)	32	R/W	Undefined	18.5.27/ 576
12D0	TCD Destination Address (DMA_TCD22_WORD_4)	32	R/W	Undefined	18.5.28/ 576
12D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD22_WORD_5)	32	R/W	Undefined	18.5.29/ 577
12D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD22_WORD_6)	32	R/W	Undefined	18.5.30/ 578
12DC	TCD Control and Status (DMA_TCD22_WORD_7)	32	R/W	Undefined	18.5.31/ 579
12E0	TCD Source Address (DMA_TCD23_WORD_0)	32	R/W	Undefined	18.5.23/ 572
12E4	TCD Transfer Attributes (DMA_TCD23_WORD_1)	32	R/W	Undefined	18.5.24/ 573
12E8	TCD Number of bytes to be transferred (DMA_TCD23_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
12E8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD23_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
12EC	TCD Last Source Address Adjustment (DMA_TCD23_WORD_3)	32	R/W	Undefined	18.5.27/ 576
12F0	TCD Destination Address (DMA_TCD23_WORD_4)	32	R/W	Undefined	18.5.28/ 576
12F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD23_WORD_5)	32	R/W	Undefined	18.5.29/ 577
12F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD23_WORD_6)	32	R/W	Undefined	18.5.30/ 578
12FC	TCD Control and Status (DMA_TCD23_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1300	TCD Source Address (DMA_TCD24_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1304	TCD Transfer Attributes (DMA_TCD24_WORD_1)	32	R/W	Undefined	18.5.24/ 573

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1308	TCD Number of bytes to be transferred (DMA_TCD24_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1308	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD24_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
130C	TCD Last Source Address Adjustment (DMA_TCD24_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1310	TCD Destination Address (DMA_TCD24_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1314	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD24_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1318	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD24_WORD_6)	32	R/W	Undefined	18.5.30/ 578
131C	TCD Control and Status (DMA_TCD24_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1320	TCD Source Address (DMA_TCD25_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1324	TCD Transfer Attributes (DMA_TCD25_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1328	TCD Number of bytes to be transferred (DMA_TCD25_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1328	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD25_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
132C	TCD Last Source Address Adjustment (DMA_TCD25_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1330	TCD Destination Address (DMA_TCD25_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1334	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD25_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1338	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD25_WORD_6)	32	R/W	Undefined	18.5.30/ 578
133C	TCD Control and Status (DMA_TCD25_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1340	TCD Source Address (DMA_TCD26_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1344	TCD Transfer Attributes (DMA_TCD26_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1348	TCD Number of bytes to be transferred (DMA_TCD26_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1348	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD26_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
134C	TCD Last Source Address Adjustment (DMA_TCD26_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1350	TCD Destination Address (DMA_TCD26_WORD_4)	32	R/W	Undefined	18.5.28/ 576

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1354	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD26_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1358	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD26_WORD_6)	32	R/W	Undefined	18.5.30/ 578
135C	TCD Control and Status (DMA_TCD26_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1360	TCD Source Address (DMA_TCD27_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1364	TCD Transfer Attributes (DMA_TCD27_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1368	TCD Number of bytes to be transferred (DMA_TCD27_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1368	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD27_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
136C	TCD Last Source Address Adjustment (DMA_TCD27_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1370	TCD Destination Address (DMA_TCD27_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1374	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD27_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1378	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD27_WORD_6)	32	R/W	Undefined	18.5.30/ 578
137C	TCD Control and Status (DMA_TCD27_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1380	TCD Source Address (DMA_TCD28_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1384	TCD Transfer Attributes (DMA_TCD28_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1388	TCD Number of bytes to be transferred (DMA_TCD28_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1388	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD28_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
138C	TCD Last Source Address Adjustment (DMA_TCD28_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1390	TCD Destination Address (DMA_TCD28_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1394	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD28_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1398	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD28_WORD_6)	32	R/W	Undefined	18.5.30/ 578
139C	TCD Control and Status (DMA_TCD28_WORD_7)	32	R/W	Undefined	18.5.31/ 579
13A0	TCD Source Address (DMA_TCD29_WORD_0)	32	R/W	Undefined	18.5.23/ 572

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13A4	TCD Transfer Attributes (DMA_TCD29_WORD_1)	32	R/W	Undefined	18.5.24/ 573
13A8	TCD Number of bytes to be transferred (DMA_TCD29_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
13A8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD29_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
13AC	TCD Last Source Address Adjustment (DMA_TCD29_WORD_3)	32	R/W	Undefined	18.5.27/ 576
13B0	TCD Destination Address (DMA_TCD29_WORD_4)	32	R/W	Undefined	18.5.28/ 576
13B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD29_WORD_5)	32	R/W	Undefined	18.5.29/ 577
13B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD29_WORD_6)	32	R/W	Undefined	18.5.30/ 578
13BC	TCD Control and Status (DMA_TCD29_WORD_7)	32	R/W	Undefined	18.5.31/ 579
13C0	TCD Source Address (DMA_TCD30_WORD_0)	32	R/W	Undefined	18.5.23/ 572
13C4	TCD Transfer Attributes (DMA_TCD30_WORD_1)	32	R/W	Undefined	18.5.24/ 573
13C8	TCD Number of bytes to be transferred (DMA_TCD30_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
13C8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD30_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
13CC	TCD Last Source Address Adjustment (DMA_TCD30_WORD_3)	32	R/W	Undefined	18.5.27/ 576
13D0	TCD Destination Address (DMA_TCD30_WORD_4)	32	R/W	Undefined	18.5.28/ 576
13D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD30_WORD_5)	32	R/W	Undefined	18.5.29/ 577
13D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD30_WORD_6)	32	R/W	Undefined	18.5.30/ 578
13DC	TCD Control and Status (DMA_TCD30_WORD_7)	32	R/W	Undefined	18.5.31/ 579
13E0	TCD Source Address (DMA_TCD31_WORD_0)	32	R/W	Undefined	18.5.23/ 572
13E4	TCD Transfer Attributes (DMA_TCD31_WORD_1)	32	R/W	Undefined	18.5.24/ 573
13E8	TCD Number of bytes to be transferred (DMA_TCD31_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
13E8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD31_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
13EC	TCD Last Source Address Adjustment (DMA_TCD31_WORD_3)	32	R/W	Undefined	18.5.27/ 576

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13F0	TCD Destination Address (DMA_TCD31_WORD_4)	32	R/W	Undefined	18.5.28/ 576
13F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD31_WORD_5)	32	R/W	Undefined	18.5.29/ 577
13F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD31_WORD_6)	32	R/W	Undefined	18.5.30/ 578
13FC	TCD Control and Status (DMA_TCD31_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1400	TCD Source Address (DMA_TCD32_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1404	TCD Transfer Attributes (DMA_TCD32_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1408	TCD Number of bytes to be transferred (DMA_TCD32_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1408	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD32_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
140C	TCD Last Source Address Adjustment (DMA_TCD32_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1410	TCD Destination Address (DMA_TCD32_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1414	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD32_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1418	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD32_WORD_6)	32	R/W	Undefined	18.5.30/ 578
141C	TCD Control and Status (DMA_TCD32_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1420	TCD Source Address (DMA_TCD33_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1424	TCD Transfer Attributes (DMA_TCD33_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1428	TCD Number of bytes to be transferred (DMA_TCD33_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1428	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD33_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
142C	TCD Last Source Address Adjustment (DMA_TCD33_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1430	TCD Destination Address (DMA_TCD33_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1434	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD33_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1438	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD33_WORD_6)	32	R/W	Undefined	18.5.30/ 578
143C	TCD Control and Status (DMA_TCD33_WORD_7)	32	R/W	Undefined	18.5.31/ 579

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1440	TCD Source Address (DMA_TCD34_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1444	TCD Transfer Attributes (DMA_TCD34_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1448	TCD Number of bytes to be transferred (DMA_TCD34_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1448	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD34_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
144C	TCD Last Source Address Adjustment (DMA_TCD34_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1450	TCD Destination Address (DMA_TCD34_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1454	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD34_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1458	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD34_WORD_6)	32	R/W	Undefined	18.5.30/ 578
145C	TCD Control and Status (DMA_TCD34_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1460	TCD Source Address (DMA_TCD35_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1464	TCD Transfer Attributes (DMA_TCD35_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1468	TCD Number of bytes to be transferred (DMA_TCD35_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1468	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD35_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
146C	TCD Last Source Address Adjustment (DMA_TCD35_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1470	TCD Destination Address (DMA_TCD35_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1474	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD35_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1478	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD35_WORD_6)	32	R/W	Undefined	18.5.30/ 578
147C	TCD Control and Status (DMA_TCD35_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1480	TCD Source Address (DMA_TCD36_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1484	TCD Transfer Attributes (DMA_TCD36_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1488	TCD Number of bytes to be transferred (DMA_TCD36_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1488	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD36_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
148C	TCD Last Source Address Adjustment (DMA_TCD36_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1490	TCD Destination Address (DMA_TCD36_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1494	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD36_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1498	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD36_WORD_6)	32	R/W	Undefined	18.5.30/ 578
149C	TCD Control and Status (DMA_TCD36_WORD_7)	32	R/W	Undefined	18.5.31/ 579
14A0	TCD Source Address (DMA_TCD37_WORD_0)	32	R/W	Undefined	18.5.23/ 572
14A4	TCD Transfer Attributes (DMA_TCD37_WORD_1)	32	R/W	Undefined	18.5.24/ 573
14A8	TCD Number of bytes to be transferred (DMA_TCD37_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
14A8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD37_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
14AC	TCD Last Source Address Adjustment (DMA_TCD37_WORD_3)	32	R/W	Undefined	18.5.27/ 576
14B0	TCD Destination Address (DMA_TCD37_WORD_4)	32	R/W	Undefined	18.5.28/ 576
14B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD37_WORD_5)	32	R/W	Undefined	18.5.29/ 577
14B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD37_WORD_6)	32	R/W	Undefined	18.5.30/ 578
14BC	TCD Control and Status (DMA_TCD37_WORD_7)	32	R/W	Undefined	18.5.31/ 579
14C0	TCD Source Address (DMA_TCD38_WORD_0)	32	R/W	Undefined	18.5.23/ 572
14C4	TCD Transfer Attributes (DMA_TCD38_WORD_1)	32	R/W	Undefined	18.5.24/ 573
14C8	TCD Number of bytes to be transferred (DMA_TCD38_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
14C8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD38_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
14CC	TCD Last Source Address Adjustment (DMA_TCD38_WORD_3)	32	R/W	Undefined	18.5.27/ 576
14D0	TCD Destination Address (DMA_TCD38_WORD_4)	32	R/W	Undefined	18.5.28/ 576
14D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD38_WORD_5)	32	R/W	Undefined	18.5.29/ 577
14D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD38_WORD_6)	32	R/W	Undefined	18.5.30/ 578

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
14DC	TCD Control and Status (DMA_TCD38_WORD_7)	32	R/W	Undefined	18.5.31/ 579
14E0	TCD Source Address (DMA_TCD39_WORD_0)	32	R/W	Undefined	18.5.23/ 572
14E4	TCD Transfer Attributes (DMA_TCD39_WORD_1)	32	R/W	Undefined	18.5.24/ 573
14E8	TCD Number of bytes to be transferred (DMA_TCD39_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
14E8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD39_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
14EC	TCD Last Source Address Adjustment (DMA_TCD39_WORD_3)	32	R/W	Undefined	18.5.27/ 576
14F0	TCD Destination Address (DMA_TCD39_WORD_4)	32	R/W	Undefined	18.5.28/ 576
14F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD39_WORD_5)	32	R/W	Undefined	18.5.29/ 577
14F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD39_WORD_6)	32	R/W	Undefined	18.5.30/ 578
14FC	TCD Control and Status (DMA_TCD39_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1500	TCD Source Address (DMA_TCD40_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1504	TCD Transfer Attributes (DMA_TCD40_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1508	TCD Number of bytes to be transferred (DMA_TCD40_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1508	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD40_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
150C	TCD Last Source Address Adjustment (DMA_TCD40_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1510	TCD Destination Address (DMA_TCD40_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1514	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD40_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1518	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD40_WORD_6)	32	R/W	Undefined	18.5.30/ 578
151C	TCD Control and Status (DMA_TCD40_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1520	TCD Source Address (DMA_TCD41_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1524	TCD Transfer Attributes (DMA_TCD41_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1528	TCD Number of bytes to be transferred (DMA_TCD41_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1528	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD41_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
152C	TCD Last Source Address Adjustment (DMA_TCD41_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1530	TCD Destination Address (DMA_TCD41_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1534	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD41_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1538	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD41_WORD_6)	32	R/W	Undefined	18.5.30/ 578
153C	TCD Control and Status (DMA_TCD41_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1540	TCD Source Address (DMA_TCD42_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1544	TCD Transfer Attributes (DMA_TCD42_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1548	TCD Number of bytes to be transferred (DMA_TCD42_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1548	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD42_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
154C	TCD Last Source Address Adjustment (DMA_TCD42_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1550	TCD Destination Address (DMA_TCD42_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1554	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD42_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1558	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD42_WORD_6)	32	R/W	Undefined	18.5.30/ 578
155C	TCD Control and Status (DMA_TCD42_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1560	TCD Source Address (DMA_TCD43_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1564	TCD Transfer Attributes (DMA_TCD43_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1568	TCD Number of bytes to be transferred (DMA_TCD43_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1568	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD43_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
156C	TCD Last Source Address Adjustment (DMA_TCD43_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1570	TCD Destination Address (DMA_TCD43_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1574	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD43_WORD_5)	32	R/W	Undefined	18.5.29/ 577

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1578	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD43_WORD_6)	32	R/W	Undefined	18.5.30/ 578
157C	TCD Control and Status (DMA_TCD43_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1580	TCD Source Address (DMA_TCD44_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1584	TCD Transfer Attributes (DMA_TCD44_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1588	TCD Number of bytes to be transferred (DMA_TCD44_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1588	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD44_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
158C	TCD Last Source Address Adjustment (DMA_TCD44_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1590	TCD Destination Address (DMA_TCD44_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1594	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD44_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1598	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD44_WORD_6)	32	R/W	Undefined	18.5.30/ 578
159C	TCD Control and Status (DMA_TCD44_WORD_7)	32	R/W	Undefined	18.5.31/ 579
15A0	TCD Source Address (DMA_TCD45_WORD_0)	32	R/W	Undefined	18.5.23/ 572
15A4	TCD Transfer Attributes (DMA_TCD45_WORD_1)	32	R/W	Undefined	18.5.24/ 573
15A8	TCD Number of bytes to be transferred (DMA_TCD45_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
15A8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD45_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
15AC	TCD Last Source Address Adjustment (DMA_TCD45_WORD_3)	32	R/W	Undefined	18.5.27/ 576
15B0	TCD Destination Address (DMA_TCD45_WORD_4)	32	R/W	Undefined	18.5.28/ 576
15B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD45_WORD_5)	32	R/W	Undefined	18.5.29/ 577
15B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD45_WORD_6)	32	R/W	Undefined	18.5.30/ 578
15BC	TCD Control and Status (DMA_TCD45_WORD_7)	32	R/W	Undefined	18.5.31/ 579
15C0	TCD Source Address (DMA_TCD46_WORD_0)	32	R/W	Undefined	18.5.23/ 572
15C4	TCD Transfer Attributes (DMA_TCD46_WORD_1)	32	R/W	Undefined	18.5.24/ 573

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
15C8	TCD Number of bytes to be transferred (DMA_TCD46_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
15C8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD46_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
15CC	TCD Last Source Address Adjustment (DMA_TCD46_WORD_3)	32	R/W	Undefined	18.5.27/ 576
15D0	TCD Destination Address (DMA_TCD46_WORD_4)	32	R/W	Undefined	18.5.28/ 576
15D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD46_WORD_5)	32	R/W	Undefined	18.5.29/ 577
15D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD46_WORD_6)	32	R/W	Undefined	18.5.30/ 578
15DC	TCD Control and Status (DMA_TCD46_WORD_7)	32	R/W	Undefined	18.5.31/ 579
15E0	TCD Source Address (DMA_TCD47_WORD_0)	32	R/W	Undefined	18.5.23/ 572
15E4	TCD Transfer Attributes (DMA_TCD47_WORD_1)	32	R/W	Undefined	18.5.24/ 573
15E8	TCD Number of bytes to be transferred (DMA_TCD47_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
15E8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD47_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
15EC	TCD Last Source Address Adjustment (DMA_TCD47_WORD_3)	32	R/W	Undefined	18.5.27/ 576
15F0	TCD Destination Address (DMA_TCD47_WORD_4)	32	R/W	Undefined	18.5.28/ 576
15F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD47_WORD_5)	32	R/W	Undefined	18.5.29/ 577
15F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD47_WORD_6)	32	R/W	Undefined	18.5.30/ 578
15FC	TCD Control and Status (DMA_TCD47_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1600	TCD Source Address (DMA_TCD48_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1604	TCD Transfer Attributes (DMA_TCD48_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1608	TCD Number of bytes to be transferred (DMA_TCD48_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1608	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD48_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
160C	TCD Last Source Address Adjustment (DMA_TCD48_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1610	TCD Destination Address (DMA_TCD48_WORD_4)	32	R/W	Undefined	18.5.28/ 576

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1614	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD48_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1618	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD48_WORD_6)	32	R/W	Undefined	18.5.30/ 578
161C	TCD Control and Status (DMA_TCD48_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1620	TCD Source Address (DMA_TCD49_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1624	TCD Transfer Attributes (DMA_TCD49_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1628	TCD Number of bytes to be transferred (DMA_TCD49_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1628	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD49_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
162C	TCD Last Source Address Adjustment (DMA_TCD49_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1630	TCD Destination Address (DMA_TCD49_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1634	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD49_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1638	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD49_WORD_6)	32	R/W	Undefined	18.5.30/ 578
163C	TCD Control and Status (DMA_TCD49_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1640	TCD Source Address (DMA_TCD50_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1644	TCD Transfer Attributes (DMA_TCD50_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1648	TCD Number of bytes to be transferred (DMA_TCD50_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1648	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD50_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
164C	TCD Last Source Address Adjustment (DMA_TCD50_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1650	TCD Destination Address (DMA_TCD50_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1654	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD50_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1658	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD50_WORD_6)	32	R/W	Undefined	18.5.30/ 578
165C	TCD Control and Status (DMA_TCD50_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1660	TCD Source Address (DMA_TCD51_WORD_0)	32	R/W	Undefined	18.5.23/ 572

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1664	TCD Transfer Attributes (DMA_TCD51_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1668	TCD Number of bytes to be transferred (DMA_TCD51_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1668	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD51_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
166C	TCD Last Source Address Adjustment (DMA_TCD51_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1670	TCD Destination Address (DMA_TCD51_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1674	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD51_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1678	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD51_WORD_6)	32	R/W	Undefined	18.5.30/ 578
167C	TCD Control and Status (DMA_TCD51_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1680	TCD Source Address (DMA_TCD52_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1684	TCD Transfer Attributes (DMA_TCD52_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1688	TCD Number of bytes to be transferred (DMA_TCD52_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1688	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD52_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
168C	TCD Last Source Address Adjustment (DMA_TCD52_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1690	TCD Destination Address (DMA_TCD52_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1694	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD52_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1698	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD52_WORD_6)	32	R/W	Undefined	18.5.30/ 578
169C	TCD Control and Status (DMA_TCD52_WORD_7)	32	R/W	Undefined	18.5.31/ 579
16A0	TCD Source Address (DMA_TCD53_WORD_0)	32	R/W	Undefined	18.5.23/ 572
16A4	TCD Transfer Attributes (DMA_TCD53_WORD_1)	32	R/W	Undefined	18.5.24/ 573
16A8	TCD Number of bytes to be transferred (DMA_TCD53_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
16A8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD53_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
16AC	TCD Last Source Address Adjustment (DMA_TCD53_WORD_3)	32	R/W	Undefined	18.5.27/ 576

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
16B0	TCD Destination Address (DMA_TCD53_WORD_4)	32	R/W	Undefined	18.5.28/ 576
16B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD53_WORD_5)	32	R/W	Undefined	18.5.29/ 577
16B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD53_WORD_6)	32	R/W	Undefined	18.5.30/ 578
16BC	TCD Control and Status (DMA_TCD53_WORD_7)	32	R/W	Undefined	18.5.31/ 579
16C0	TCD Source Address (DMA_TCD54_WORD_0)	32	R/W	Undefined	18.5.23/ 572
16C4	TCD Transfer Attributes (DMA_TCD54_WORD_1)	32	R/W	Undefined	18.5.24/ 573
16C8	TCD Number of bytes to be transferred (DMA_TCD54_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
16C8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD54_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
16CC	TCD Last Source Address Adjustment (DMA_TCD54_WORD_3)	32	R/W	Undefined	18.5.27/ 576
16D0	TCD Destination Address (DMA_TCD54_WORD_4)	32	R/W	Undefined	18.5.28/ 576
16D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD54_WORD_5)	32	R/W	Undefined	18.5.29/ 577
16D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD54_WORD_6)	32	R/W	Undefined	18.5.30/ 578
16DC	TCD Control and Status (DMA_TCD54_WORD_7)	32	R/W	Undefined	18.5.31/ 579
16E0	TCD Source Address (DMA_TCD55_WORD_0)	32	R/W	Undefined	18.5.23/ 572
16E4	TCD Transfer Attributes (DMA_TCD55_WORD_1)	32	R/W	Undefined	18.5.24/ 573
16E8	TCD Number of bytes to be transferred (DMA_TCD55_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
16E8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD55_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
16EC	TCD Last Source Address Adjustment (DMA_TCD55_WORD_3)	32	R/W	Undefined	18.5.27/ 576
16F0	TCD Destination Address (DMA_TCD55_WORD_4)	32	R/W	Undefined	18.5.28/ 576
16F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD55_WORD_5)	32	R/W	Undefined	18.5.29/ 577
16F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD55_WORD_6)	32	R/W	Undefined	18.5.30/ 578
16FC	TCD Control and Status (DMA_TCD55_WORD_7)	32	R/W	Undefined	18.5.31/ 579

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1700	TCD Source Address (DMA_TCD56_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1704	TCD Transfer Attributes (DMA_TCD56_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1708	TCD Number of bytes to be transferred (DMA_TCD56_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1708	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD56_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
170C	TCD Last Source Address Adjustment (DMA_TCD56_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1710	TCD Destination Address (DMA_TCD56_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1714	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD56_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1718	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD56_WORD_6)	32	R/W	Undefined	18.5.30/ 578
171C	TCD Control and Status (DMA_TCD56_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1720	TCD Source Address (DMA_TCD57_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1724	TCD Transfer Attributes (DMA_TCD57_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1728	TCD Number of bytes to be transferred (DMA_TCD57_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1728	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD57_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
172C	TCD Last Source Address Adjustment (DMA_TCD57_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1730	TCD Destination Address (DMA_TCD57_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1734	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD57_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1738	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD57_WORD_6)	32	R/W	Undefined	18.5.30/ 578
173C	TCD Control and Status (DMA_TCD57_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1740	TCD Source Address (DMA_TCD58_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1744	TCD Transfer Attributes (DMA_TCD58_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1748	TCD Number of bytes to be transferred (DMA_TCD58_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1748	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD58_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
174C	TCD Last Source Address Adjustment (DMA_TCD58_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1750	TCD Destination Address (DMA_TCD58_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1754	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD58_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1758	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD58_WORD_6)	32	R/W	Undefined	18.5.30/ 578
175C	TCD Control and Status (DMA_TCD58_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1760	TCD Source Address (DMA_TCD59_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1764	TCD Transfer Attributes (DMA_TCD59_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1768	TCD Number of bytes to be transferred (DMA_TCD59_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1768	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD59_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
176C	TCD Last Source Address Adjustment (DMA_TCD59_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1770	TCD Destination Address (DMA_TCD59_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1774	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD59_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1778	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD59_WORD_6)	32	R/W	Undefined	18.5.30/ 578
177C	TCD Control and Status (DMA_TCD59_WORD_7)	32	R/W	Undefined	18.5.31/ 579
1780	TCD Source Address (DMA_TCD60_WORD_0)	32	R/W	Undefined	18.5.23/ 572
1784	TCD Transfer Attributes (DMA_TCD60_WORD_1)	32	R/W	Undefined	18.5.24/ 573
1788	TCD Number of bytes to be transferred (DMA_TCD60_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
1788	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD60_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
178C	TCD Last Source Address Adjustment (DMA_TCD60_WORD_3)	32	R/W	Undefined	18.5.27/ 576
1790	TCD Destination Address (DMA_TCD60_WORD_4)	32	R/W	Undefined	18.5.28/ 576
1794	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD60_WORD_5)	32	R/W	Undefined	18.5.29/ 577
1798	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD60_WORD_6)	32	R/W	Undefined	18.5.30/ 578

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
179C	TCD Control and Status (DMA_TCD60_WORD_7)	32	R/W	Undefined	18.5.31/ 579
17A0	TCD Source Address (DMA_TCD61_WORD_0)	32	R/W	Undefined	18.5.23/ 572
17A4	TCD Transfer Attributes (DMA_TCD61_WORD_1)	32	R/W	Undefined	18.5.24/ 573
17A8	TCD Number of bytes to be transferred (DMA_TCD61_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
17A8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD61_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
17AC	TCD Last Source Address Adjustment (DMA_TCD61_WORD_3)	32	R/W	Undefined	18.5.27/ 576
17B0	TCD Destination Address (DMA_TCD61_WORD_4)	32	R/W	Undefined	18.5.28/ 576
17B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD61_WORD_5)	32	R/W	Undefined	18.5.29/ 577
17B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD61_WORD_6)	32	R/W	Undefined	18.5.30/ 578
17BC	TCD Control and Status (DMA_TCD61_WORD_7)	32	R/W	Undefined	18.5.31/ 579
17C0	TCD Source Address (DMA_TCD62_WORD_0)	32	R/W	Undefined	18.5.23/ 572
17C4	TCD Transfer Attributes (DMA_TCD62_WORD_1)	32	R/W	Undefined	18.5.24/ 573
17C8	TCD Number of bytes to be transferred (DMA_TCD62_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574
17C8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD62_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
17CC	TCD Last Source Address Adjustment (DMA_TCD62_WORD_3)	32	R/W	Undefined	18.5.27/ 576
17D0	TCD Destination Address (DMA_TCD62_WORD_4)	32	R/W	Undefined	18.5.28/ 576
17D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD62_WORD_5)	32	R/W	Undefined	18.5.29/ 577
17D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD62_WORD_6)	32	R/W	Undefined	18.5.30/ 578
17DC	TCD Control and Status (DMA_TCD62_WORD_7)	32	R/W	Undefined	18.5.31/ 579
17E0	TCD Source Address (DMA_TCD63_WORD_0)	32	R/W	Undefined	18.5.23/ 572
17E4	TCD Transfer Attributes (DMA_TCD63_WORD_1)	32	R/W	Undefined	18.5.24/ 573
17E8	TCD Number of bytes to be transferred (DMA_TCD63_WORD_2_NBYTES)	32	R/W	Undefined	18.5.25/ 574

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
17E8	TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD63_WORD_2_MLOOP)	32	R/W	Undefined	18.5.26/ 575
17EC	TCD Last Source Address Adjustment (DMA_TCD63_WORD_3)	32	R/W	Undefined	18.5.27/ 576
17F0	TCD Destination Address (DMA_TCD63_WORD_4)	32	R/W	Undefined	18.5.28/ 576
17F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD63_WORD_5)	32	R/W	Undefined	18.5.29/ 577
17F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD63_WORD_6)	32	R/W	Undefined	18.5.30/ 578
17FC	TCD Control and Status (DMA_TCD63_WORD_7)	32	R/W	Undefined	18.5.31/ 579

18.5.1 Control Register (DMA_CR)

This register defines the basic operating configuration of the eDMA.

The eDMA arbitrates channel service requests in groups of 16 channels. The 64- and 32-channel configurations have four groups (3,2,1,0) and two groups (1,0), respectively; the 16 channel configuration has only one group (0). Group 3 contains channels 63-48, group 2 contains channels 47-32, group 1 contains channels 31-16, and group 0 contains channels 15-0.

Arbitration within a group can be configured to use either a fixed-priority or a round-robin selection. In fixed-priority arbitration, the highest priority channel requesting service is selected to execute. The priorities are assigned by the channel priority registers (see Channel n Priority Register (DMA_DCHPRI*n*). In round-robin arbitration mode, the channel priorities are ignored and the channels within each group are cycled through without regard to priority.

The group priorities operate in a similar fashion. In group fixed-priority arbitration mode, channel service requests in the highest priority group are executed first where priority level 3 is the highest and priority level 0 is the lowest. The group priorities are assigned in the GRPnPri registers. All group priorities must have unique values prior to any channel service requests occur, otherwise a configuration error will be reported. Unused group priority registers, per configuration, are unimplemented in the CR. In group round-robin mode, the group priorities are ignored and the groups are cycled through without regard to priority.

Minor loop offsets are address offset values added to the final source address (SADDR) or destination address (DADDR) upon minor loop completion. When minor loop offsets are enabled, the minor loop offset (MLOFF) is added to the final source address (SADDR), or the final destination address (DADDR), or both prior to the addresses being written back into the TCD. If the major loop is complete, the minor loop offset is ignored and the major loop address offsets (SLAST and DLAST_SGA) are used to compute the next SADDR and DADDR values.

When minor loop mapping is enabled (CR[EMLM] = 1), TCDn_WORD_2 is redefined. A portion of TCDn_WORD_2 is used to specify multiple fields: a source enable bit (smloe) to specify the minor loop offset should be applied to the source address (saddr) upon minor loop completion, a destination enable bit (DMLOE) to specify the minor loop offset should be applied to the destination address (DADDR) upon minor loop completion, and the sign extended minor loop offset value (MLOFF). The same offset value (MLOFF) is used for both source and destination minor loop offsets. When either minor loop offset is enabled (SMLOE set or DMLOE set), the NBYTES field is reduced to 10 bits. When both minor loop offsets are disabled (SMLOE cleared and DMLOE cleared), the NBYTES field is a 30-bit vector.

When minor loop mapping is disabled (CR[EMLM] = 0), all 32 bits of TCDn_WORD_2 are assigned to the NBYTES field.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0														CX	ECX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GRP3PRI		GRP2PRI		GRP1PRI		GRP0PRI		EMLM	CLM	HALT	HOE	ERGA	ERCA	EDBG	EBW
W																
Reset	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0

DMA_CR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 CX	Cancel Transfer

Table continues on the next page...

DMA_CR field descriptions (continued)

Field	Description
	0 Normal operation 1 Cancel the remaining data transfer. Stop the executing channel and force the minor loop to be finished. The cancel takes effect after the last write of the current read/write sequence. The CX bit clears itself after the cancel has been honored. This cancel retires the channel normally as if the minor loop was completed.
15 ECX	Error Cancel Transfer 0 Normal operation 1 Cancel the remaining data transfer in the same fashion as the CX bit. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The ECX bit clears itself after the cancel is honored. In addition to canceling the transfer, ECX treats the cancel as an error condition, thus updating the Error Status register (ES) and generating an optional error interrupt.
16–17 GRP3PRI	Channel Group 3 Priority Group 3 priority level when fixed priority group arbitration is enabled.
18–19 GRP2PRI	Channel Group 2 Priority Group 2 priority level when fixed priority group arbitration is enabled.
20–21 GRP1PRI	Channel Group 1 Priority Group 1 priority level when fixed priority group arbitration is enabled.
22–23 GRP0PRI	Channel Group 0 Priority Group 0 priority level when fixed priority group arbitration is enabled.
24 EMLM	Enable Minor Loop Mapping. 0 Minor loop mapping disabled. TCDn_WORD_2 is defined as a 32-bit nbytes field. 1 Minor loop mapping enabled. When set, TCDn_WORD_2 is redefined to include individual enable fields, an offset field and the nbytes field. The individual enable fields allow the minor loop offset to be applied to the source address, the destination address, or both. The nbytes field is reduced when either offset is enabled.
25 CLM	Continuous Link Mode 0 A minor loop channel link made to itself will go through channel arbitration before being activated again. 1 A minor loop channel link made to itself will not go through channel arbitration before being activated again. Upon minor loop completion, the channel will be active again if that channel has a minor loop channel link enabled and the link channel is itself. This effectively applies the minor loop offsets and restarts the next minor loop.
26 HALT	Halt DMA Operations 0 Normal operation 1 Stall the start of any new channels. Executing channels are allowed to complete. Channel execution resumes when this bit is cleared.
27 HOE	Halt On Error 0 Normal operation 1 Any error causes the HALT bit to set. Subsequently, all service requests are ignored until the HALT bit is cleared.

Table continues on the next page...

DMA_CR field descriptions (continued)

Field	Description
28 ERGA	Enable Round Robin Group Arbitration 0 Fixed priority arbitration is used for selection among the groups. 1 Round robin arbitration is used for selection among the groups.
29 ERCA	Enable Round Robin Channel Arbitration 0 Fixed priority arbitration is used for channel selection within each group. 1 Round robin arbitration is used for channel selection within each group.
30 EDBG	Enable Debug 0 The assertion of the ipg_debug input is ignored. 1 The assertion of the ipg_debug input causes the eDMA to stall the start of a new channel. Executing channels are allowed to complete. Channel execution will resume when either the ipg_debug input is negated or the EDBG bit is cleared.
31 EBW	Enable Buffered Writes 0 The bufferable write signal (hprot[2]) is not asserted during AMBA AHB writes. 1 The bufferable write signal (hprot[2]) is asserted on all AMBA AHB writes except for the last write sequence.

18.5.2 Error Status Register (DMA_ES)

This register provides information concerning the last recorded channel error. Channel errors can be caused by a configuration error (an illegal setting in the transfer control descriptor or an illegal priority register setting in fixed arbitration mode) or an error termination to a bus master read or write cycle.

A configuration error is caused when the starting source or destination address, source or destination offsets, minor loop byte count, and the transfer size represent an inconsistent state. The addresses and offsets must be aligned on 0-modulo-transfer_size boundaries, and the minor loop byte count must be a multiple of the source and destination transfer sizes. All source reads and destination writes must be configured to the natural boundary of the programmed transfer size respectively. In fixed arbitration mode, a configuration error is caused by any two channel priorities being equal within a group, or any group priority levels being equal among the groups. All channel priority levels within a group must be unique and all group priority levels among the groups must be unique when fixed arbitration mode is enabled. If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST_SGA) is not aligned on a 32-byte boundary. If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCDn_WORD_5[CITER_E_LINK] bit does not equal the TCDn_WORD_7[BITER_E_LINK] bit. All configuration error conditions except scatter/gather and minor loop link error are reported as the channel is activated and assert an

error interrupt request, if enabled. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion when properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address and current iteration count at the point of the fault. When a system bus error occurs, the channel is terminated after the read or write transaction, which is already pipelined after errant access, has completed. If a bus error occurs on the last read prior to beginning the write sequence, the write will execute using the data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence will execute before the channel is terminated due to the destination bus error.

A transfer may be canceled by software via the CR[CX] bit. When a cancel transfer request is recognized, the eDMA engine stops processing the channel. The current read-write sequence is allowed to finish. If the cancel occurs on the last read-write sequence of a major or minor loop, the cancel request is discarded and the channel retires normally.

The error cancel transfer is the same as a cancel transfer except the ES register is updated with the canceled channel number and error cancel bit is set. The TCD of a canceled channel has the source address and destination address of the last transfer saved in the TCD. The user should initialize the TCD again if the channel needs to be restarted because the aforementioned fields have been modified by the eDMA engine and no longer represent the original parameters. When a transfer is canceled via the error cancel transfer mechanism (setting the CR[ECX] or asserting the dma_err_cancel_xfer input), the channel number is loaded into the ERRCHN field and the ECX and VLD bits are set in the ES register. In addition, an error interrupt may be generated if enabled. See the Enable Error Interrupt Register High (DMA_ERRH) and Enable Error Interrupt RegisterLow (DMA_ERRL) registers for error interrupt details.

The occurrence of any type of error causes the eDMA engine to immediately stop, and the appropriate channel bit in the eDMA Error register to be asserted. At the same time, the details of the error condition are loaded into the ES register. The major loop complete indicators, setting the transfer control descriptor done flag and the possible assertion of an interrupt request, are not affected when an error is detected.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0													UCE	ECX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GPE	CPE	ERRCHN						SAE	SOE	DAE	DOE	NCE	SGE	SBE	DBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_ES field descriptions

Field	Description
0 VLD	Logical OR of all ERRH and ERRL status bits. 0 No ERR bits are set 1 At least one ERR bit is set indicating a valid error exists that has not been cleared
1–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 UCE	Uncorrectable ECC error during channel execution. The UCE bit is set when an uncorrectable ECC error occurs on an access generated by the DMA only. If a CPU access to the TCD causes an uncorrectable ECC error, that access will receive a bus error response.” 0 No uncorrectable ECC error 1 The last recorded error was an uncorrectable TCD RAM error
15 ECX	Transfer Canceled. 0 No canceled transfers 1 The last recorded entry was a canceled transfer by the error cancel transfer input
16 GPE	Group Priority Error. 0 No group priority error 1 The last recorded error was a configuration error among the group priorities. All group priorities are not unique.
17 CPE	Channel Priority Error. 0 No channel priority error 1 The last recorded error was a configuration error in the channel priorities within a group. All channel priorities within a group are not unique.
18–23 ERRCHN	Error/Cancelled Channel Number. The channel number of the last recorded error (excluding GPE and CPE errors) or last recorded transfer that was canceled with error exit.
24 SAE	Source Address Error. 0 No source address configuration error. 1 The last recorded error was a configuration error detected in the TCDn_WORD_0[SADDR] field. TCDn_WORD_0[SADDR] is inconsistent with TCn_TCDn_WORD_1[SSIZE].
25 SOE	Source Offset Error 0 No source offset configuration error 1 The last recorded error was a configuration error detected in the TCD_TCDn_WORD_1[SOFF] field. TCDTCDn_WORD_1[SOFF] is inconsistent with TCDTCDn_WORD_1[SSIZE].
26 DAE	Destination Address Error.

Table continues on the next page...

DMA_ES field descriptions (continued)

Field	Description
	0 No destination address configuration error 1 The last recorded error was a configuration error detected in the TCDTCDn_WORD_4[DADDR] field. TCDTCDn_WORD_4[DADDR] is inconsistent with TCDTCDn_WORD_1[DSIZE].
27 DOE	Destination Offset Error. 0 No destination offset configuration error 1 The last recorded error was a configuration error detected in the TCDTCDn_WORD_5[DOFF] field. TCDTCDn_WORD_5[DOFF] is inconsistent with TCDTCDn_WORD_1[DSIZE].
28 NCE	NBYTES/CITER Configuration Error. 0 No NBYTES/CITER configuration error 1 The last recorded error was a configuration error detected in the TCDTCDn_WORD_2[NBYTES] or TCDTCDn_WORD_5[CITER] fields. TCDTCDn_WORD_2[NBYTES] is not a multiple of TCDTCDn_WORD_1[SSIZE] and TCDTCDn_WORD_1[DSIZE], or TCDTCDn_WORD_5[CITER] is equal to zero, or TCD[CITER_E_LINK] is not equal to TCDTCDn_WORD_7[BITER_E_LINK].
29 SGE	Scatter/Gather Configuration Error. 0 No scatter/gather configuration error 1 The last recorded error was a configuration error detected in the TCDTCDn_WORD_6[DLAST_SGA] field. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCDn_WORD_7[ESG] is enabled. TCDn_WORD_6[DLAST_SGA] is not on a 32 byte boundary.
30 SBE	Source Bus Error. 0 No source bus error 1 The last recorded error was a bus error on a source read
31 DBE	Destination Bus Error. 0 No destination bus error 1 The last recorded error was a bus error on a destination write

18.5.3 Enable Request Register High (DMA_ERQH)

The ERQ{H,L} registers provide a bit map to enable the request signal for each of the 64 channels. ERQH supports channels 63-32, while ERQL covers channels 31-00. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ registers. The SERQ and CERQ registers are provided so that the request enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the ERQ{H,L} registers.

Both the eDMA request input signal and enable request flag must be asserted before a channel's hardware service request is accepted. The state of the eDMA enable request flag does not affect a channel service request made explicitly through software or a linked channel request.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERQ63	ERQ62	ERQ61	ERQ60	ERQ59	ERQ58	ERQ57	ERQ56	ERQ55	ERQ54	ERQ53	ERQ52	ERQ51	ERQ50	ERQ49	ERQ48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERQ47	ERQ46	ERQ45	ERQ44	ERQ43	ERQ42	ERQ41	ERQ40	ERQ39	ERQ38	ERQ37	ERQ36	ERQ35	ERQ34	ERQ33	ERQ32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_ERQH field descriptions

Field	Description
0 ERQ63	Enable DMA Request 63 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
1 ERQ62	Enable DMA Request 62 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
2 ERQ61	Enable DMA Request 61 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
3 ERQ60	Enable DMA Request 60 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
4 ERQ59	Enable DMA Request 59 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
5 ERQ58	Enable DMA Request 58 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
6 ERQ57	Enable DMA Request 57 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
7 ERQ56	Enable DMA Request 56 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

Table continues on the next page...

DMA_ERQH field descriptions (continued)

Field	Description
8 ERQ55	Enable DMA Request 55 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
9 ERQ54	Enable DMA Request 54 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
10 ERQ53	Enable DMA Request 53 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
11 ERQ52	Enable DMA Request 52 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
12 ERQ51	Enable DMA Request 51 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
13 ERQ50	Enable DMA Request 50 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
14 ERQ49	Enable DMA Request 49 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
15 ERQ48	Enable DMA Request 48 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
16 ERQ47	Enable DMA Request 47 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
17 ERQ46	Enable DMA Request 46 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
18 ERQ45	Enable DMA Request 45 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
19 ERQ44	Enable DMA Request 44 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

Table continues on the next page...

DMA_ERQH field descriptions (continued)

Field	Description
20 ERQ43	Enable DMA Request 43 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
21 ERQ42	Enable DMA Request 42 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
22 ERQ41	Enable DMA Request 41 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
23 ERQ40	Enable DMA Request 40 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
24 ERQ39	Enable DMA Request 39 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
25 ERQ38	Enable DMA Request 38 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
26 ERQ37	Enable DMA Request 37 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
27 ERQ36	Enable DMA Request 36 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
28 ERQ35	Enable DMA Request 35 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
29 ERQ34	Enable DMA Request 34 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
30 ERQ33	Enable DMA Request 33 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
31 ERQ32	Enable DMA Request 32 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

18.5.4 Enable Request Register Low (DMA_ERQL)

The ERQ{H,L} registers provide a bit map to enable the request signal for each of the 64 channels. ERQH supports channels 63-32, while ERQL covers channels 31-00. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ registers. The SERQ and CERQ registers are provided so that the request enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the ERQ{H,L} registers.

Both the eDMA request input signal and enable request flag must be asserted before a channel's hardware service request is accepted. The state of the eDMA enable request flag does not affect a channel service request made explicitly through software or a linked channel request.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERQ31	ERQ30	ERQ29	ERQ28	ERQ27	ERQ26	ERQ25	ERQ24	ERQ23	ERQ22	ERQ21	ERQ20	ERQ19	ERQ18	ERQ17	ERQ16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERQ15	ERQ14	ERQ13	ERQ12	ERQ11	ERQ10	ERQ9	ERQ8	ERQ7	ERQ6	ERQ5	ERQ4	ERQ3	ERQ2	ERQ1	ERQ0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_ERQL field descriptions

Field	Description
0 ERQ31	Enable DMA Request 31 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
1 ERQ30	Enable DMA Request 30 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
2 ERQ29	Enable DMA Request 29 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
3 ERQ28	Enable DMA Request 28

Table continues on the next page...

DMA_ERQL field descriptions (continued)

Field	Description
	0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
4 ERQ27	Enable DMA Request 27 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
5 ERQ26	Enable DMA Request 26 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
6 ERQ25	Enable DMA Request 25 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
7 ERQ24	Enable DMA Request 24 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
8 ERQ23	Enable DMA Request 23 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
9 ERQ22	Enable DMA Request 22 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
10 ERQ21	Enable DMA Request 21 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
11 ERQ20	Enable DMA Request 20 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
12 ERQ19	Enable DMA Request 19 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
13 ERQ18	Enable DMA Request 18 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
14 ERQ17	Enable DMA Request 17 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
15 ERQ16	Enable DMA Request 16

Table continues on the next page...

DMA_ERQL field descriptions (continued)

Field	Description
	0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
16 ERQ15	Enable DMA Request 15 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
17 ERQ14	Enable DMA Request 14 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
18 ERQ13	Enable DMA Request 13 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
19 ERQ12	Enable DMA Request 12 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
20 ERQ11	Enable DMA Request 11 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
21 ERQ10	Enable DMA Request 10 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
22 ERQ9	Enable DMA Request 9 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
23 ERQ8	Enable DMA Request 8 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
24 ERQ7	Enable DMA Request 7 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
25 ERQ6	Enable DMA Request 6 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
26 ERQ5	Enable DMA Request 5 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
27 ERQ4	Enable DMA Request 4

Table continues on the next page...

DMA_ERQL field descriptions (continued)

Field	Description
	0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
28 ERQ3	Enable DMA Request 3 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
29 ERQ2	Enable DMA Request 2 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
30 ERQ1	Enable DMA Request 1 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
31 ERQ0	Enable DMA Request 0 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

18.5.5 Enable Error Interrupt Register High (DMA_EEIH)

The EEI{H,L} registers provide a bit map to enable the error interrupt signal for each of the 64 channels. EEIH supports channels 63-32, while EEIL covers channels 31-00. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI registers. The SEEI and CEEI registers are provided so that the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEI{H,L} registers.

Both the eDMA error indicator and error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EEI63	EEI62	EEI61	EEI60	EEI59	EEI58	EEI57	EEI56	EEI55	EEI54	EEI53	EEI52	EEI51	EEI50	EEI49	EEI48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EEI47	EEI46	EEI45	EEI44	EEI43	EEI42	EEI41	EEI40	EEI39	EEI38	EEI37	EEI36	EEI35	EEI34	EEI33	EEI32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_EEIH field descriptions

Field	Description
0 EEI63	Enable Error Interrupt 63 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
1 EEI62	Enable Error Interrupt 62 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
2 EEI61	Enable Error Interrupt 61 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
3 EEI60	Enable Error Interrupt 60 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
4 EEI59	Enable Error Interrupt 59 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
5 EEI58	Enable Error Interrupt 58 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
6 EEI57	Enable Error Interrupt 57 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
7 EEI56	Enable Error Interrupt 56 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
8 EEI55	Enable Error Interrupt 55 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
9 EEI54	Enable Error Interrupt 54 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table continues on the next page...

DMA_EEIH field descriptions (continued)

Field	Description
10 EEI53	Enable Error Interrupt 53 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
11 EEI52	Enable Error Interrupt 52 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
12 EEI51	Enable Error Interrupt 51 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
13 EEI50	Enable Error Interrupt 50 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
14 EEI49	Enable Error Interrupt 49 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
15 EEI48	Enable Error Interrupt 48 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
16 EEI47	Enable Error Interrupt 47 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
17 EEI46	Enable Error Interrupt 46 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
18 EEI45	Enable Error Interrupt 45 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
19 EEI44	Enable Error Interrupt 44 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
20 EEI43	Enable Error Interrupt 43 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
21 EEI42	Enable Error Interrupt 42 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table continues on the next page...

DMA_EEIH field descriptions (continued)

Field	Description
22 EEI41	Enable Error Interrupt 41 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
23 EEI40	Enable Error Interrupt 40 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
24 EEI39	Enable Error Interrupt 39 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
25 EEI38	Enable ErrorInterrupt 38 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
26 EEI37	Enable Error Interrupt 37 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
27 EEI36	Enable Error Interrupt 36 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
28 EEI35	Enable Error Interrupt 35 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
29 EEI34	Enable Error Interrupt 34 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
30 EEI33	Enable Error Interrupt 33 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
31 EEI32	Enable Error Interrupt 32 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

18.5.6 Enable Error Interrupt Register Low (DMA_EEIL)

The EEI{H,L} registers provide a bit map to enable the error interrupt signal for each of the 64 channels. EEIH supports channels 63-32, while EEIL covers channels 31-00. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI registers. The SEEI and CEEI registers are provided so that the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEI{H,L} registers.

Both the eDMA error indicator and error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EEI31	EEI30	EEI29	EEI28	EEI27	EEI26	EEI25	EEI24	EEI23	EEI22	EEI21	EEI20	EEI19	EEI18	EEI17	EEI16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EEI15	EEI14	EEI13	EEI12	EEI11	EEI10	EEI9	EEI8	EEI7	EEI6	EEI5	EEI4	EEI3	EEI2	EEI1	EEI0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_EEIL field descriptions

Field	Description
0 EEI31	Enable Error Interrupt 31 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
1 EEI30	Enable Error Interrupt 30 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
2 EEI29	Enable Error Interrupt 29 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
3 EEI28	Enable Error Interrupt 28

Table continues on the next page...

DMA_EEIL field descriptions (continued)

Field	Description
	0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
4 EEI27	Enable Error Interrupt 27 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
5 EEI26	Enable Error Interrupt 26 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
6 EEI25	Enable Error Interrupt 25 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
7 EEI24	Enable Error Interrupt 24 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
8 EEI23	Enable Error Interrupt 23 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
9 EEI22	Enable Error Interrupt 22 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
10 EEI21	Enable Error Interrupt 21 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
11 EEI20	Enable Error Interrupt 20 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
12 EEI19	Enable Error Interrupt 19 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
13 EEI18	Enable Error Interrupt 18 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
14 EEI17	Enable Error Interrupt 17 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
15 EEI16	Enable Error Interrupt 16

Table continues on the next page...

DMA_EEIL field descriptions (continued)

Field	Description
	0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
16 EEI15	Enable Error Interrupt 15 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
17 EEI14	Enable Error Interrupt 14 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
18 EEI13	Enable Error Interrupt 13 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
19 EEI12	Enable Error Interrupt 12 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
20 EEI11	Enable Error Interrupt 11 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
21 EEI10	Enable Error Interrupt 10 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
22 EEI9	Enable Error Interrupt 9 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
23 EEI8	Enable Error Interrupt 8 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
24 EEI7	Enable Error Interrupt 7 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
25 EEI6	Enable ErrorInterrupt 6 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
26 EEI5	Enable Error Interrupt 5 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
27 EEI4	Enable Error Interrupt 4

Table continues on the next page...

DMA_EEIL field descriptions (continued)

Field	Description
	0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
28 EEI3	Enable Error Interrupt 3 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
29 EEI2	Enable Error Interrupt 2 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
30 EEI1	Enable Error Interrupt 1 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
31 EEI0	Enable Error Interrupt 0 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

18.5.7 Set Enable Request Register (DMA_SERQ)

The SERQ register provides a simple memory-mapped mechanism to set a given bit in the ERQ{H,L} registers to enable the eDMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ{H,L} register to be set. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global set function, forcing the entire contents of ERQ{H,L} to be asserted. If bit 7 is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7
Read	0	SERQ						
Write	NOP							
Reset	0	0	0	0	0	0	0	0

DMA_SERQ field descriptions

Field	Description
0 NOP	No Op enable

Table continues on the next page...

DMA_SERQ field descriptions (continued)

Field	Description
	0 Normal operation 1 No operation, ignore the other bits in this register
1–7 SERQ	Set Enable Request. 0-63: Set the corresponding bit in ERQ{H,L} 64-127: Set all bits in ERQ{H,L}

18.5.8 Clear Enable Request Register (DMA_CERQ)

The CERQ register provides a simple memory-mapped mechanism to clear a given bit in the ERQ{H,L} registers to disable the eDMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ{H,L} register to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing the entire contents of the ERQ{H,L} to be zeroed, disabling all eDMA request inputs. If the NOP bit is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 19h offset = 19h

Bit	0	1	2	3	4	5	6	7
Read	0	0						
Write	NOP	CERQ						
Reset	0	0	0	0	0	0	0	0

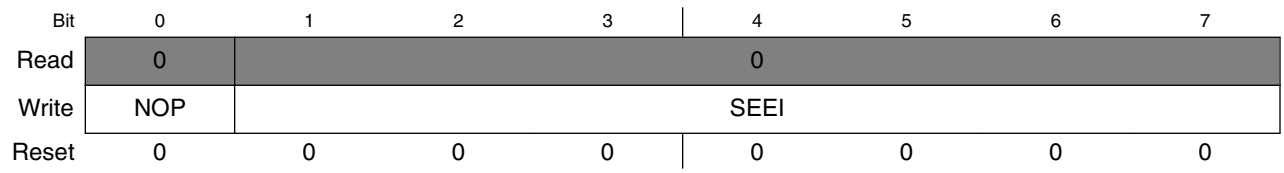
DMA_CERQ field descriptions

Field	Description
0 NOP	No Op enable. 0 Normal operation 1 No operation, ignore the other bits in this register
1–7 CERQ	Clear enable request. 0-63: Clear corresponding bit in ERQ{H,L} 64-127: Clear all bits in ERQ{H,L}

18.5.9 Set Enable Error Interrupt Register (DMA_SEEI)

The SEEI register provides a simple memory-mapped mechanism to set a given bit in the EEI{H,L} registers to enable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI{H,L} register to be set. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global set function, forcing the entire contents of EEI{H,L} to be asserted. If the NOP bit is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Ah offset = 1Ah



DMA_SEEI field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1–7 SEEI	Set enable error interrupt 0-63: Set the corresponding bit in EEI{H,L} 64-127: Set all bits in EEI{H,L}

18.5.10 Clear Enable Error Interrupt Register (DMA_CEEI)

The CEEI register provides a simple memory-mapped mechanism to clear a given bit in the EEI{H,L} registers to disable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI{H,L} register to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing the entire contents of the EEI{H,L} to be zeroed, disabling all eDMA request inputs. If the NOP bit is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Bh offset = 1Bh

Bit	0	1	2	3	4	5	6	7
Read	0	0						
Write	NOP	CEEI						
Reset	0	0	0	0	0	0	0	0

DMA_CEEI field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1–7 CEEI	Clear Enable Error Interrupt. 0-63: Clear corresponding bit in EEI{H,L} 64-127: Clear all bits in EEI{H,L}

18.5.11 Clear Interrupt Request Register (DMA_CINT)

The CINT register provides a simple memory-mapped mechanism to clear a given bit in the INT{H,L} registers to disable the interrupt request for a given channel. The given value on a register write causes the corresponding bit in the INT{H,L} register to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing the entire contents of the INT{H,L} to be zeroed, disabling all eDMA interrupt requests. If the NOP bit is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7
Read	0	0						
Write	NOP	CINT						
Reset	0	0	0	0	0	0	0	0

DMA_CINT field descriptions

Field	Description
0 NOP	No Op enable. 0 Normal operation 1 No operation, ignore the other bits in this register
1–7 CINT	Clear Interrupt Request. 0-63: Clear the corresponding bit in INT{H,L} 64-127: Clear all bits in INT{H,L}

18.5.12 Clear Error Register (DMA_CERR)

The CERR register provides a simple memory-mapped mechanism to clear a given bit in the ERR{H,L} registers to disable the error condition flag for a given channel. The given value on a register write causes the corresponding bit in the ERR{H,L} register to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing the entire contents of the ERR{H,L} to be zeroed, clearing all channel error indicators. If the NOP bit is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Dh offset = 1Dh

Bit	0	1	2	3	4	5	6	7
Read	0	0						
Write	NOP	CERR						
Reset	0	0	0	0	0	0	0	0

DMA_CERR field descriptions

Field	Description
0 NOP	No Op enable. 0 Normal operation 1 No operation, ignore the other bits in this register
1–7 CERR	Clear Error Indicator. 0-63: Clear corresponding bit in ERR{H,L} 64-127: Clear all bits in ERR{H,L}

18.5.13 Set START Bit Register (DMA_SSRT)

The SSRT register provides a simple memory-mapped mechanism to set the START bit in the TCD of the given channel. The data value on a register write causes the START bit in the corresponding Transfer Control Descriptor to be set. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global set function, forcing all START bits to be set. If the NOP bit is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Eh offset = 1Eh

Bit	0	1	2	3	4	5	6	7
Read	0	0						
Write	NOP	SSRT						
Reset	0	0	0	0	0	0	0	0

DMA_SSRT field descriptions

Field	Description
0 NOP	No Op enable. 0 Normal operation 1 No operation, ignore the other bits in this register
1–7 SSRT	Set START Bit 0-63: Sets the corresponding bit in TCDn_WORD_7[START] 64-127: Set all TCDn_WORD_7[START] bits

18.5.14 Clear DONE Status Bit Register (DMA_CDNE)

The CDNE register provides a simple memory-mapped mechanism to clear the DONE bit in the TCD of the given channel. The data value on a register write causes the DONE bit in the corresponding Transfer Control Descriptor to be cleared. A data value of 64 to 127 (regardless of the number of implemented channels) provides a global clear function, forcing all DONE bits to be cleared. If bit 7 is set, the command is ignored. This allows multiple byte registers to be written as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Fh offset = 1Fh

Bit	0	1	2	3	4	5	6	7
Read	0	0						
Write	NOP	CDNE						
Reset	0	0	0	0	0	0	0	0

DMA_CDNE field descriptions

Field	Description
0 NOP	No Op enable. 0 Normal operation 1 No operation, ignore the other bits in this register
1–7 CDNE	Clear DONE Status Bit. 0-63: Clears the corresponding bit in TCDn_WORD_7[DONE] 64-127: Clear all TCDn_WORD_7[DONE] bits

18.5.15 Interrupt Request Register High (DMA_INTH)

The INT{H,L} registers provide a bit map for the 64 channels, signaling the presence of an interrupt request for each channel. INTH supports channels 63-32, while INTL covers channels 31-00. The eDMA engine signals the occurrence of a programmed interrupt upon the completion of a data transfer as defined in the transfer control descriptor by setting the appropriate bit in this register. The outputs of this register are directly routed to the platform's interrupt controller. During the execution of the interrupt service routine associated with any given channel, it is software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the CINT register in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT register. On writes to the INT, a one in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no affect on the corresponding channel's current interrupt status. The CINT register is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INT{H,L} registers.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INT63	INT62	INT61	INT60	INT59	INT58	INT57	INT56	INT55	INT54	INT53	INT52	INT51	INT50	INT49	INT48
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INT47	INT46	INT45	INT44	INT43	INT42	INT41	INT40	INT39	INT38	INT37	INT36	INT35	INT34	INT33	INT32
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_INTH field descriptions

Field	Description
0 INT63	Interrupt Request 63 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
1 INT62	Interrupt Request 62 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
2 INT61	Interrupt Request 61 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
3 INT60	Interrupt Request 60 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
4 INT59	Interrupt Request 59

Table continues on the next page...

DMA_INTH field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
5 INT58	Interrupt Request 58 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
6 INT57	Interrupt Request 57 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
7 INT56	Interrupt Request 56 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
8 INT55	Interrupt Request 55 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
9 INT54	Interrupt Request 54 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
10 INT53	Interrupt Request 53 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
11 INT52	Interrupt Request 52 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
12 INT51	Interrupt Request 51 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
13 INT50	Interrupt Request 50 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
14 INT49	Interrupt Request 49 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
15 INT48	Interrupt Request 48 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
16 INT47	Interrupt Request 47

Table continues on the next page...

DMA_INTH field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
17 INT46	Interrupt Request 46 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
18 INT45	Interrupt Request 45 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
19 INT44	Interrupt Request 44 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
20 INT43	Interrupt Request 43 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
21 INT42	Interrupt Request 42 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
22 INT41	Interrupt Request 41 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
23 INT40	Interrupt Request 40 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
24 INT39	Interrupt Request 30 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
25 INT38	Interrupt Request 38 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
26 INT37	Interrupt Request 37 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
27 INT36	Interrupt Request 36 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
28 INT35	Interrupt Request 35

Table continues on the next page...

DMA_INTH field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
29 INT34	Interrupt Request 34 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
30 INT33	Interrupt Request 33 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
31 INT32	Interrupt Request 32 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active

18.5.16 Interrupt Request Register Low (DMA_INTL)

The INT{H,L} registers provide a bit map for the 64 channels, signaling the presence of an interrupt request for each channel. INTH supports channels 63-32, while INTL covers channels 31-00. The eDMA engine signals the occurrence of a programmed interrupt upon the completion of a data transfer as defined in the transfer control descriptor by setting the appropriate bit in this register. The outputs of this register are directly routed to the platform's interrupt controller. During the execution of the interrupt service routine associated with any given channel, it is software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the CINT register in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT register. On writes to the INT, a one in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no affect on the corresponding channel's current interrupt status. The CINT register is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INT{H,L} registers.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INT31	INT30	INT29	INT28	INT27	INT26	INT25	INT24	INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_INTL field descriptions

Field	Description
0 INT31	Interrupt Request 31 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
1 INT30	Interrupt Request 30 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
2 INT29	Interrupt Request 29 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
3 INT28	Interrupt Request 28 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
4 INT27	Interrupt Request 27 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
5 INT26	Interrupt Request 26 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
6 INT25	Interrupt Request 25

Table continues on the next page...

DMA_INTL field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
7 INT24	Interrupt Request 24 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
8 INT23	Interrupt Request 23 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
9 INT22	Interrupt Request 22 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
10 INT21	Interrupt Request 21 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
11 INT20	Interrupt Request 20 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
12 INT19	Interrupt Request 19 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
13 INT18	Interrupt Request 18 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
14 INT17	Interrupt Request 17 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
15 INT16	Interrupt Request 16 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
16 INT15	Interrupt Request 15 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
17 INT14	Interrupt Request 14 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
18 INT13	Interrupt Request 13

Table continues on the next page...

DMA_INTL field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
19 INT12	Interrupt Request 12 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
20 INT11	Interrupt Request 11 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
21 INT10	Interrupt Request 10 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
22 INT9	Interrupt Request 9 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
23 INT8	Interrupt Request 8 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
24 INT7	Interrupt Request 7 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
25 INT6	Interrupt Request 6 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
26 INT5	Interrupt Request 5 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
27 INT4	Interrupt Request 4 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
28 INT3	Interrupt Request 3 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
29 INT2	Interrupt Request 2 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
30 INT1	Interrupt Request 1

Table continues on the next page...

DMA_INTL field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
31 INT0	Interrupt Request 0 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active

18.5.17 Enable Error Interrupt Register High (DMA_ERRH)

The ERR{H,L} registers provide a bit map for the 64 channels, signaling the presence of an error for each channel. ERRH supports channels 63-32, while ERRL covers channels 31-00. The eDMA engine signals the occurrence of an error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the EEI register, then logically summed across groups of 16, 32, and 64 channels to form several group error interrupt requests which is then routed to the platform's interrupt controller. During the execution of the interrupt service routine associated with any eDMA errors, it is software's responsibility to clear the appropriate bit, negating the error interrupt request. Typically, a write to the CERR register in the interrupt service routine is used for this purpose. Recall the normal eDMA channel completion indicators, setting the transfer control descriptor done flag, and the possible assertion of an interrupt request are not affected when an error is detected.

The contents of this register can also be polled and a non-zero value indicates the presence of a channel error, regardless of the state of the EEI register. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR register. On writes to the ERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no affect on the corresponding channel's current error status. The CERR register is provided so the error indicator for a single channel can easily be cleared.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR63	ERR62	ERR61	ERR60	ERR59	ERR58	ERR57	ERR56	ERR55	ERR54	ERR53	ERR52	ERR51	ERR50	ERR49	ERR48
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR47	ERR46	ERR45	ERR44	ERR43	ERR42	ERR41	ERR40	ERR39	ERR38	ERR37	ERR36	ERR35	ERR34	ERR33	ERR32
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_ERRH field descriptions

Field	Description
0 ERR63	Error In Channel 63 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
1 ERR62	Error In Channel 62 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
2 ERR61	Error In Channel 61 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
3 ERR60	Error In Channel 60 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
4 ERR59	Error In Channel 59 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
5 ERR58	Error In Channel 58 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
6 ERR57	Error In Channel 57 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
7 ERR56	Error In Channel 56 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
8 ERR55	Error In Channel 55 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
9 ERR54	Error In Channel 54

Table continues on the next page...

DMA_ERRH field descriptions (continued)

Field	Description
	0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
10 ERR53	Error In Channel 53 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
11 ERR52	Error In Channel 52 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
12 ERR51	Error In Channel 51 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
13 ERR50	Error In Channel 50 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
14 ERR49	Error In Channel 49 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
15 ERR48	Error In Channel 48 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
16 ERR47	Error In Channel 47 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
17 ERR46	Error In Channel 46 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
18 ERR45	Error In Channel 45 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
19 ERR44	Error In Channel 44 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
20 ERR43	Error In Channel 43 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
21 ERR42	Error In Channel 42

Table continues on the next page...

DMA_ERRH field descriptions (continued)

Field	Description
	0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
22 ERR41	Error In Channel 41 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
23 ERR40	Error In Channel 40 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
24 ERR39	Error In Channel 39 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
25 ERR38	Error In Channel 38 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
26 ERR37	Error In Channel 37 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
27 ERR36	Error In Channel 36 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
28 ERR35	Error In Channel 35 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
29 ERR34	Error In Channel 34 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
30 ERR33	Error In Channel 33 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
31 ERR32	Error In Channel 32 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred

18.5.18 Enable Error Interrupt RegisterLow (DMA_ERRL)

The ERR{H,L} registers provide a bit map for the 64 channels, signaling the presence of an error for each channel. ERRH supports channels 63-32, while ERRL covers channels 31-00. The eDMA engine signals the occurrence of an error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the EEI register, then logically summed across groups of 16, 32, and 64 channels to form several group error interrupt requests which is then routed to the platform's interrupt controller. During the execution of the interrupt service routine associated with any eDMA errors, it is software's responsibility to clear the appropriate bit, negating the error interrupt request. Typically, a write to the CERR register in the interrupt service routine is used for this purpose. Recall the normal eDMA channel completion indicators, setting the transfer control descriptor done flag and the possible assertion of an interrupt request, are not affected when an error is detected.

The contents of this register can also be polled and a non-zero value indicates the presence of a channel error, regardless of the state of the EEI register. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR register. On writes to the ERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no affect on the corresponding channel's current error status. The CERR register is provided so the error indicator for a single channel can easily be cleared.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR31	ERR30	ERR29	ERR28	ERR27	ERR26	ERR25	ERR24	ERR23	ERR22	ERR21	ERR20	ERR19	ERR18	ERR17	ERR16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR15	ERR14	ERR13	ERR12	ERR11	ERR10	ERR9	ERR8	ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_ERRL field descriptions

Field	Description
0 ERR31	Error In Channel 31 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
1 ERR30	Error In Channel 30 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
2 ERR29	Error In Channel 29 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
3 ERR28	Error In Channel 28 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
4 ERR27	Error In Channel 27 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
5 ERR26	Error In Channel 26 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
6 ERR25	Error In Channel 25 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
7 ERR24	Error In Channel 24 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
8 ERR23	Error In Channel 23 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
9 ERR22	Error In Channel 22 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
10 ERR21	Error In Channel 21 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
11 ERR20	Error In Channel 20 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred

Table continues on the next page...

DMA_ERRL field descriptions (continued)

Field	Description
12 ERR19	Error In Channel 19 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
13 ERR18	Error In Channel 18 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
14 ERR17	Error In Channel 17 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
15 ERR16	Error In Channel 16 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
16 ERR15	Error In Channel 15 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
17 ERR14	Error In Channel 14 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
18 ERR13	Error In Channel 13 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
19 ERR12	Error In Channel 12 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
20 ERR11	Error In Channel 11 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
21 ERR10	Error In Channel 10 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
22 ERR9	Error In Channel 9 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
23 ERR8	Error In Channel 8 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred

Table continues on the next page...

DMA_ERRL field descriptions (continued)

Field	Description
24 ERR7	Error In Channel 7 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
25 ERR6	Error In Channel 6 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
26 ERR5	Error In Channel 5 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
27 ERR4	Error In Channel 4 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
28 ERR3	Error In Channel 3 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
29 ERR2	Error In Channel 2 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
30 ERR1	Error In Channel 1 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred
31 ERR0	Error In Channel 0 0 An error in the corresponding channel has not occurred 1 An error in the corresponding channel has occurred

18.5.19 Hardware Request Status Register High (DMA_HRSH)

The HRS{H,L} registers provide a bit map for the 64 channels to show the current hardware request status for each channel. HRSH supports channels 63-32, while HRSL covers channels 31-00. Hardware request status reflects the current state of the registered and qualified (via the ERQ field) ipd_req lines as seen by the eDMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	HRS63	HRS62	HRS61	HRS60	HRS59	HRS58	HRS57	HRS56	HRS55	HRS54	HRS53	HRS52	HRS51	HRS50	HRS49	HRS48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	HRS47	HRS46	HRS45	HRS44	HRS43	HRS42	HRS41	HRS40	HRS39	HRS38	HRS37	HRS36	HRS35	HRS34	HRS33	HRS32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_HRSH field descriptions

Field	Description
0 HRS63	Hardware Request Status Channel 63 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
1 HRS62	Hardware Request Status Channel 62 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
2 HRS61	Hardware Request Status Channel 61 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
3 HRS60	Hardware Request Status Channel 60 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
4 HRS59	Hardware Request Status Channel 59

Table continues on the next page...

DMA_HRSH field descriptions (continued)

Field	Description
	0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
5 HRS58	Hardware Request Status Channel 58 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
6 HRS57	Hardware Request Status Channel 57 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
7 HRS56	Hardware Request Status Channel 56 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
8 HRS55	Hardware Request Status Channel 55 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
9 HRS54	Hardware Request Status Channel 54 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
10 HRS53	Hardware Request Status Channel 53 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
11 HRS52	Hardware Request Status Channel 52 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
12 HRS51	Hardware Request Status Channel 51 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
13 HRS50	Hardware Request Status Channel 50 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
14 HRS49	Hardware Request Status Channel 49 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
15 HRS48	Hardware Request Status Channel 48 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
16 HRS47	Hardware Request Status Channel 47

Table continues on the next page...

DMA_HRSH field descriptions (continued)

Field	Description
	0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
17 HRS46	Hardware Request Status Channel 46 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
18 HRS45	Hardware Request Status Channel 45 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
19 HRS44	Hardware Request Status Channel 44 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
20 HRS43	Hardware Request Status Channel 43 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
21 HRS42	Hardware Request Status Channel 42 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
22 HRS41	Hardware Request Status Channel 41 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
23 HRS40	Hardware Request Status Channel 40 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
24 HRS39	Hardware Request Status Channel 39 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
25 HRS38	Hardware Request Status Channel 38 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
26 HRS37	Hardware Request Status Channel 37 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
27 HRS36	Hardware Request Status Channel 36 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
28 HRS35	Hardware Request Status Channel 35

Table continues on the next page...

DMA_HRSH field descriptions (continued)

Field	Description
	0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
29 HRS34	Hardware Request Status Channel 34 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
30 HRS33	Hardware Request Status Channel 33 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
31 HRS32	Hardware Request Status Channel 32 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present

18.5.20 Hardware Request Status Register Low (DMA_HRSL)

The HRS{H,L} registers provide a bit map for the 64 channels to show the current hardware request status for each channel. HRSH supports channels 63-32, while HRSL covers channels 31-00. Hardware request status reflects the current state of the registered and qualified (via the ERQ field) ipd_req lines as seen by the eDMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	HRS31	HRS30	HRS29	HRS28	HRS27	HRS26	HRS25	HRS24	HRS23	HRS22	HRS21	HRS20	HRS19	HRS18	HRS17	HRS16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	HRS15	HRS14	HRS13	HRS12	HRS11	HRS10	HRS9	HRS8	HRS7	HRS6	HRS5	HRS4	HRS3	HRS2	HRS1	HRS0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_HRSL field descriptions

Field	Description
0 HRS31	Hardware Request Status Channel 31 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
1 HRS30	Hardware Request Status Channel 30 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
2 HRS29	Hardware Request Status Channel 29 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
3 HRS28	Hardware Request Status Channel 28 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
4 HRS27	Hardware Request Status Channel 27 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
5 HRS26	Hardware Request Status Channel 26 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
6 HRS25	Hardware Request Status Channel 25 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
7 HRS24	Hardware Request Status Channel 24 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
8 HRS23	Hardware Request Status Channel 23 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
9 HRS22	Hardware Request Status Channel 22 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
10 HRS21	Hardware Request Status Channel 21 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
11 HRS20	Hardware Request Status Channel 20 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present

Table continues on the next page...

DMA_HRSL field descriptions (continued)

Field	Description
12 HRS19	Hardware Request Status Channel 19 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
13 HRS18	Hardware Request Status Channel 18 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
14 HRS17	Hardware Request Status Channel 17 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
15 HRS16	Hardware Request Status Channel 16 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
16 HRS15	Hardware Request Status Channel 15 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
17 HRS14	Hardware Request Status Channel 14 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
18 HRS13	Hardware Request Status Channel 13 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
19 HRS12	Hardware Request Status Channel 12 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
20 HRS11	Hardware Request Status Channel 11 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
21 HRS10	Hardware Request Status Channel 10 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
22 HRS9	Hardware Request Status Channel 9 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
23 HRS8	Hardware Request Status Channel 8 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present

Table continues on the next page...

DMA_HRSL field descriptions (continued)

Field	Description
24 HRS7	Hardware Request Status Channel 7 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
25 HRS6	Hardware Request Status Channel 6 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
26 HRS5	Hardware Request Status Channel 5 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
27 HRS4	Hardware Request Status Channel 4 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
28 HRS3	Hardware Request Status Channel 3 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
29 HRS2	Hardware Request Status Channel 2 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
30 HRS1	Hardware Request Status Channel 1 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present
31 HRS0	Hardware Request Status Channel 0 0 A hardware service request for the corresponding channel is not present 1 A hardware service request for the corresponding channel is present

18.5.21 Channel n Priority Register (DMA_DCHPRI*n*)

When the fixed-priority channel arbitration mode is enabled (CR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel within a group. The channel priorities are evaluated by numeric value, that is, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, etc. Software must program the channel priorities with unique values, otherwise a configuration error will be reported. The range of the priority value is limited to the values of 0–15. When read, the GRPPRI bits of the DCHPRIn register reflect the current priority level of the group of channels in which the corresponding channel resides. GRPPRI bits are not affected by writes to the DCHPRIn registers. The group priority is assigned in the CR. See the Control Register (DMA_CR) definition.

Channel preemption is enabled on a per channel basis by setting the ECP bit in the DCHPRIn register. Channel preemption allows the executing channel's data transfers to be temporarily suspended in favor of starting a higher priority channel. After the preempting channel has completed all of its minor loop data transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel will be suspended and the higher priority channel will be serviced. Nested preemption (attempting to preempt a preempting channel) is not supported. After a preempting channel begins execution, it cannot be preempted. Preemption is only available when fixed arbitration is selected for both group and channel arbitration modes.

A channel's ability to preempt another channel can be disabled by setting the DPA bit in the DCHPRIn register. When a channel's preempt ability is disabled, that channel cannot suspend a lower priority channel's data transfer; regardless of the lower priority channel's ECP setting. This allows for a pool of low priority, large data moving channels to be defined. These low priority channels can be configured to not preempt each other, thus preventing a low priority channel from consuming the preempt slot normally available a true, high priority channel.

Address: 0h base + 100h offset + (1d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7
Read	ECP	DPA	GRPPRI		CHPRI			
Write								
Reset	0	0	*	*	*	*	*	*

* Notes:
- CHPRI field: After reset, the default value of CHPRI bits will be equal to channel number.
- GRPPRI field: See bit field description

DMA_DCHPRI*n* field descriptions

Field	Description
0 ECP	Enable Channel Preemption. 0 Channel n cannot be suspended by a higher priority channel's service request 1 Channel n can be temporarily suspended by the service request of a higher priority channel
1 DPA	Disable Preempt Ability. 0 Channel n can suspend a lower priority channel 1 Channel n cannot suspend any channel, regardless of channel priority
2–3 GRPPRI	Channel n Current Group Priority Group priority assigned to this channel group when fixed-priority arbitration is enabled. These two bits are read only; writes are ignored.

Table continues on the next page...

DMA_DCHPRI*n* field descriptions (continued)

Field	Description
4–7 CHPRI	Channel n Arbitration Priority Channel priority when fixed-priority arbitration is enabled.

18.5.22 Channel n Master ID Register (DMA_DCHMID*n*)

The eDMA master ID replication registers allow the eDMA to use the same protection level and AHB system bus ID of the master programming the eDMA's TCD. When enabled, the eDMA will use the master ID and protection level stored in the DCHMID register instead of the eDMA's default values. When a master (a core for example) programs a TCD, its master ID and protection level are captured when TCD_WORD_7 control attributes are written. Although the scatter/gather operation can change the contents of TCD_WORD_7, that operation does not affect the DCHMIDn registers.

Address: 0h base + 140h offset + (1d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7
Read	EMI	PAL	0		MID			
Write								
Reset	0	0	0	0	0	0	0	0

DMA_DCHMID*n* field descriptions

Field	Description
0 EMI	Enable Master ID replication. 0 Master ID replication is disabled 1 Master ID replication is enabled
1 PAL	Privileged Access Level. The protection level captured in this register reflects the level used when writing the channel's control attributes; TCD_WORD_7, bits 24:31. 0 User protection level for DMA transfers 1 Privileged protection level for DMA transfers
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 MID	Master ID. eDMA's master ID when channel n is active and master ID replication is enabled. **NOTE:** The master ID captured in this register reflects the ID used when writing the channel's control attributes; TCD_WORD_7, bits 24-31

18.5.23 TCD Source Address (DMA_TCD*n*_WORD_0)

Each channel requires a 32-byte transfer control descriptor for defining the desired data movement operation. The TCD structure is discussed in detail in Features. The channel descriptors are stored in the local memory in sequential order: channel 0, channel 1, ... channel [n-1]. The definitions of the TCD are presented as eight 32-bit values. The following table is a 32-bit view of the basic TCD structure.

Table 18-154. TCDn 32-bit memory structure

eDMA Offset	TCDn Field	
0x1000 + (32 x n) + 0x00	Source Address (saddr)	
0x1000 + (32 x n) + 0x04	Transfer Attributes (SMOD, SSIZE, DMOD, DSIZE)	Signed Source Address Offset (SOFF)
0x1000 + (32 x n) + 0x08	Signed Minor Loop Offset (SMLOE, DMLOE, MLOF)	Inner “Minor” Byte Count (NBYTES)
0x1000 + (32 x n) + 0x0C	Last Source Address Adjustment (SLAST)	
0x1000 + (32 x n) + 0x10	Destination Address (DADDR)	
0x1000 + (32 x n) + 0x14	Current “Major” Iteration Count (CITER)	Signed Destination Address Offset (DOFF)
0x1000 + (32 x n) + 0x18	Last Destination Address Adjustment/Scatter Gather Address (DLAST_SGA)	
0x1000 + (32 x n) + 0x1C	Beginning “Major” Iteration Count (BITER)	Channel Control/Status BWC, MAJOR.LINKCH, DONE, ACTIVE, MAJOR.E.LINK,E_SG, D_REQ, INT_HALF, INT_MAJ, START

Address: 0h base + 1000h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SADDR																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

DMA_TCD*n*_WORD_0 field descriptions

Field	Description
0–31 SADDR	Source Address. Memory address pointing to the source data.

18.5.24 TCD Transfer Attributes (DMA_TCD*n*_WORD_1)

Address: 0h base + 1004h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SMOD					SSIZE			DMOD					DSIZE			SOFF															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

DMA_TCD*n*_WORD_1 field descriptions

Field	Description
0–4 SMOD	Source Address Modulo. 0 Source address modulo feature is disabled ≠0 The value defines a specific address bit which is selected to be either the value after saddr + soff calculation is performed or the original register value. This feature provides the ability to easily implement a circular data queue. For data queues requiring power-of-2 “size” bytes, the queue should be based at a 0-modulo-size address and the smod field set to the appropriate value to freeze the upper address bits. The bit select is defined as ((1 less than smod[4:0]) - 1) where a resulting 1 in a bit location selects the next state address for the corresponding address bit location and a 0 selects the original register value for the corresponding address bit location. For this application, the soff is typically set to the transfer size to implement post-increment addressing with the smod function constraining the addresses to a 0-modulo-size range.
5–7 SSIZE	Source data transfer size. 000 8-bit 001 16-bit 010 32-bit 011 Reserved 100 16-byte 101 32-byte 110 Reserved 111 Reserved
8–12 DMOD	Destination Address Modulo. See the SMOD definition.
13–15 DSIZE	Destination data transfer size See the SSIZE definition
16–31 SOFF	Source address signed offset. Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.

18.5.25 TCD Number of bytes to be transferred (DMA_TCD*n*_WORD_2_NBYTES)

When CR[EMLM] = 0, TCD_Word_2_NBYTES is defined as the number of bytes to be transferred in each service request of the channel.

Address: 0h base + 1008h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NBYTES																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

DMA_TCD*n*_WORD_2_NBYTES field descriptions

Field	Description
0–31 NBYTES	Inner "Minor" Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel is activated, the contents of the appropriate TCD is loaded into the eDMA engine, and the appropriate reads and writes performed until the complete byte transfer count has been transferred. This is an indivisible operation and cannot be stalled or halted. After the minor count is exhausted, the current values of the saddr and daddr are written back into the local memory, the major iteration count is decremented and restored to the local memory. If the major iteration count is completed, additional processing is performed. **NOTE:** The NBYTES value 0x0000_0000 is interpreted as 0x1_0000_0000, thus specifying a 4 GB transfer.

18.5.26 TCD Signed Minor Loop Offset (Minor Loop and Offset Enabled) (DMA_TCD*n*_WORD_2_MLOOP)

When CR[EMLM] = 1, TCD WORD_2_MLOOP is defined as four fields: a source minor loop offset enable, a destination minor loop offset enable, a minor loop offset field and a NBYTES field.

Address: 0h base + 1008h offset + (32d × i), where i=0d to 63d



* Notes:
- x = Undefined at reset.

DMA_TCD*n*_WORD_2_MLOOP field descriptions

Field	Description
0 SMLOE	Source minor loop offset enable. This flag selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the SADDR. 1 The minor loop offset is applied to the SADDR.
1 DMLOE	Destination minor loop offset enable. This flag selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0 The minor loop offset is not applied to the DADDR. 1 The minor loop offset is applied to the DADDR.
2–21 MLOFF	Inner “minor” byte transfer count or Minor loop offset. If both SMLOE and DMLOE are cleared, this field is part of the byte transfer count. If either SMLOE or DMLOE are set, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the minor loop is completed.

Table continues on the next page...

DMA_TCD*n*_WORD_2_MLOOP field descriptions (continued)

Field	Description
22–31 NBYTES	Inner "Minor" Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel is activated, the contents of the appropriate TCD is loaded into the eDMA engine, and the appropriate reads and writes performed until the complete byte transfer count has been transferred. This is an indivisible operation and cannot be stalled or halted. Once the minor count is exhausted, the current values of the saddr and daddr are written back into the local memory, the major iteration count is decremented and restored to the local memory. If the major iteration count is completed, additional processing is performed. **NOTE:** This field is extended to 30 bits when both SMLOE and DMLOE are cleared (disabled).

18.5.27 TCD Last Source Address Adjustment (DMA_TCD*n*_WORD_3)

Address: 0h base + 100Ch offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SLAST																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

DMA_TCD*n*_WORD_3 field descriptions

Field	Description
0–31 SLAST	Last Source Address Adjustment Adjustment value added to the source address at the completion of the outer major iteration count. This value can be applied to “restore” the source address to the initial value, or adjust the address to reference the next data structure.

18.5.28 TCD Destination Address (DMA_TCD*n*_WORD_4)

Address: 0h base + 1010h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DADDR																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

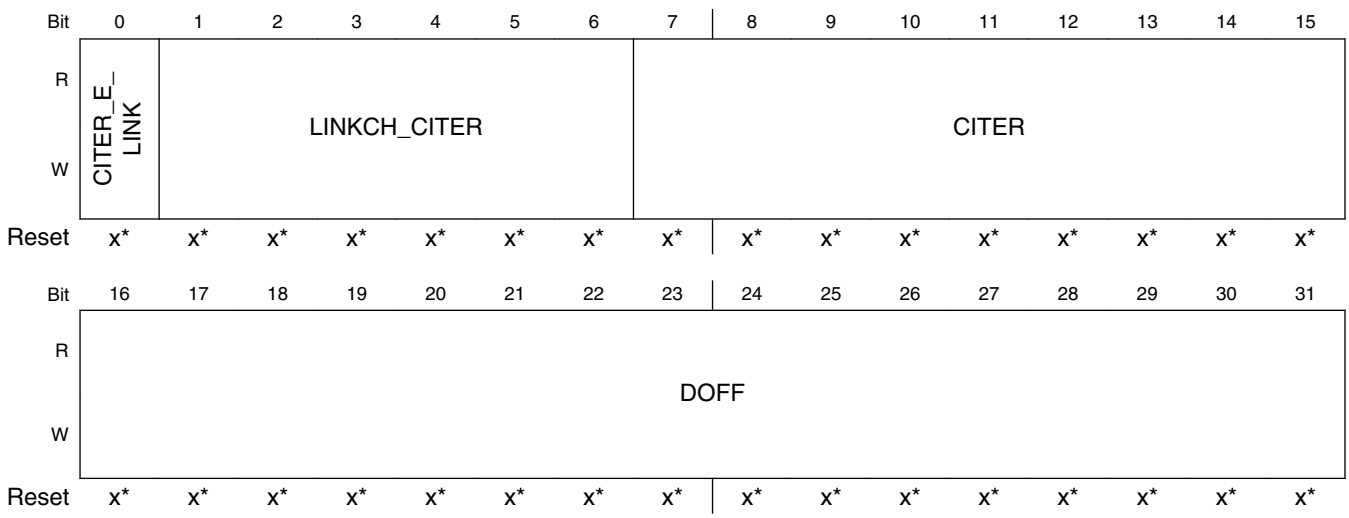
DMA_TCD*n*_WORD_4 field descriptions

Field	Description
0–31 DADDR	Destination Address. Memory address pointing to the destination data.

18.5.29 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_WORD_5)

If TCDn_WORD_5[CITER_E_LINK] is set, the TCDn_WORD_5[CITER] field is defined as follows.

Address: 0h base + 1014h offset + (32d × i), where i=0d to 63d



* Notes:
- x = Undefined at reset.

DMA_TCD*n*_WORD_5 field descriptions

Field	Description
0 CITER_E_LINK	Enable channel-to-channel linking on minor-loop complete. As the channel completes the inner minor loop, this flag enables the linking to another channel, defined by CITER-LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_WORD_7[START] bit of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the "major" loop is exhausted, this link mechanism is suppressed in favor of the TCDn_WORD_7[MAJOR_E_LINK] channel linking. **NOTE:** This bit must be equal to the TCDn_WORD_7[BITER_E_LINK] bit; otherwise, a configuration error is reported.

Table continues on the next page...

DMA_TCD*n*_WORD_5 field descriptions (continued)

Field	Description
	0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
1–6 LINKCH_CITER	Current “major” iteration count or Link channel number. If TCDn_WORD_5[CITER_E_LINK] = 0 Then No channel-to-channel linking (or chaining) is performed after the inner "minor" loop is exhausted. LINKCH-CITER are used to form a 15-bit CITER field. Else After the "minor" loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by CITER_LINKCH by setting that channel’s TCDn_WORD_7[START] bit. **NOTE:** The value contained in CITER_E_LINK must not exceed the number of implemented channels.
7–15 CITER	Current Major Iteration Count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations, optionally generating an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field. **NOTE:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. **NOTE:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.
16–31 DOFF	Destination address signed offset. Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed.

18.5.30 TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD*n*_WORD_6)

Address: 0h base + 1018h offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DLAST_SGA																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

DMA_TCD*n*_WORD_6 field descriptions

Field	Description
0–31 DLAST_SGA	Last destination address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather).

DMA_TCD*n*_WORD_6 field descriptions (continued)

Field	Description

	If (TCDn_WORD_7[E_SG] = 0)
	Then
	Adjustment value added to the destination address at the completion of the outer major iteration count. This value can be applied to “restore” the destination address to the initial value, or adjust the address to reference the next data structure.
	Else
	This address points to the beginning of a 0-modulo-32 region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo-32, else a configuration error is reported.

18.5.31 TCD Control and Status (DMA_TCD*n*_WORD_7)

Address: 0h base + 101Ch offset + (32d × i), where i=0d to 63d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	BITER_E_LINK	BITER_BITER_LINKCH							BITER							
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	BWC		MAJOR_LINKCH						DONE	ACTIVE	MAJOR_E_LINK	E_SG	DREQ	INT_HALF	INT_MAJOR	START
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

DMA_TCD*n*_WORD_7 field descriptions

Field	Description
0 BITER_E_LINK	Enable channel-to-channel linking on minor loop complete. This is the initial value copied into the citer.e_link field when the major loop is completed. The CITER_E_LINK field controls channel linking during channel execution. **NOTE:** This bit must be equal to the CITER_E_LINK bit otherwise a configuration error will be reported. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.

Table continues on the next page...

DMA_TCD*n*_WORD_7 field descriptions (continued)

Field	Description
1–6 BITER_BITER_ LINKCH	"Beginning major iteration count" or "Beginning Link channel number". This is the initial value copied into the citer field or citer.linkch field when the major loop is completed. The CITER fields controls the iteration count and linking during channel execution. If BITER_E_LINK = 0 Then No channel-to-channel linking (or chaining) is performed after the inner "minor" loop is exhausted. BITER-BITER_LINKCH are used to form a 15-bit BITER field. Else After the "minor" loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by BITER_LINKCH by setting that channel's TCD_WORD_7[START] bit. The value contained in BITER_LINKCH must not exceed the number of implemented channels.
7–15 BITER	Beginning "major" iteration count. This is the initial value copied into the CITER field or CITER.LINKCH field when the major loop is completed. The CITER fields control the iteration count and linking during channel execution. This 9- or 15-bit count represents the beginning major loop count for the channel. As the major iteration count is exhausted, the contents of the entire 16-bit BITER entry is reloaded into the 16-bit CITER entry. When the BITER field is initially loaded by software, it must be set to the same value as that contained in the CITER field. If the channel is configured to execute a single service request, the initial values of biter and citer should be 0x0001.
16–17 BWC	Bandwidth Control. This 2-bit field provides a mechanism to effectively throttle the amount of bus bandwidth consumed by the eDMA. In general, as the eDMA processes the inner minor loop, it continuously generates read/write, read/write, ...sequences until the minor count is exhausted. This field forces the eDMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the platform's cross-bar arbitration switch. To minimize start-up latency, bandwidth control stalls are suppressed for the first two AHB bus cycles and after the last write of each minor loop. 00 No eDMA engine stalls 01 Reserved 10 eDMA engine stalls for 4 cycles after each read/write 11 eDMA engine stalls for 8 cycles after each read/write

18–23
MAJOR_LINKCH
	Link Channel Number.
	If (MAJOR_E_LINK = 0,)
	Then
	No channel-to-channel linking (or chaining) is performed after the outer "major" loop counter is exhausted.
	Else
	After the "major" loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by MAJOR_LINKCH[5:0] by setting that channel's TCD_START bit.
	NOTE: The value contained in MAJOR_LINKCH[5:0] must not exceed the number of implemented channels.

24 DONE	Channel Done.

Table continues on the next page...

DMA_TCD*n*_WORD_7 field descriptions (continued)

Field	Description
	This flag indicates the eDMA has completed the major loop. The eDMA engine sets it as the CITER count reaches zero. The software or the hardware clears it when the channel is activated. **NOTE:** This bit must be cleared to write the MAJOR_E_LINK or ESG bits.
25 ACTIVE	Channel Active. This flag signals the channel is currently in execution. It is set when channel service begins, and is cleared by the eDMA as the minor loop completes or when any error condition is detected.
26 MAJOR_E_LINK	Enable channel-to-channel linking on major loop complete. As the channel completes the major loop, this flag enables the linking to another channel, defined by MAJOR_LINKCH. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. **NOTE:** To support the dynamic linking coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
27 E_SG	Enable Scatter/Cather Processing. As the channel completes the outer major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses DLAST_SGA as a memory pointer to a 0-modulo-32 address containing a 32-byte data structure which is loaded as the transfer control descriptor into the local memory. **NOTE:** To support the dynamic scatter/gather coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set. 0 The current channel's TCD is normal format. 1 The current channel's TCD specifies a scatter gather format. The DLAST_SGA field provides a memory pointer to the next TCD to be loaded into this channel after the outer major loop completes its execution.
28 DREQ	Disable Request. If this flag is set, the eDMA hardware automatically clears the corresponding ERQ bit when the current major iteration count reaches zero. 0 The channel's ERQ bit is not affected 1 The channel's ERQ bit is cleared when the major loop is complete
29 INT_HALF	Enable an interrupt when major counter is half complete. If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT register when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (CITER == (BITER >> 1)). This halfway point interrupt request is provided to support double-buffered schemes or other types of data movement where the processor needs an early indication of the transfer's progress. **NOTE:** If BITER = 1, do not use INT_HALT; use INT_MAJOR instead. 0 The half-point interrupt is disabled 1 The half-point interrupt is enabled
30 INT_MAJOR	Enable an interrupt when major iteration count completes.

Table continues on the next page...

DMA_TCD*n*_WORD_7 field descriptions (continued)

Field	Description
	If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT register when the current major iteration count reaches zero. 0 The end-of-major loop interrupt is disabled 1 The end-of-major loop interrupt is enabled
31 START	Channel Start. If this flag is set, the channel is requesting service. The eDMA hardware automatically clears this flag after the channel begins execution. 0 The channel is not explicitly started 1 The channel is explicitly started via a software initiated service request

18.6 Functional description

This section provides an overview of the microarchitecture and functional operation of the eDMA module.

18.6.1 eDMA microarchitecture

The eDMA module is partitioned into two major modules:

- the eDMA engine and
- the transfer-control descriptor local memory.

The eDMA engine is further partitioned into four submodules that are shown in the following table.

Table 18-803. eDMA engine submodules

Submodule	Function
Address path	This block implements registered versions of two channel transfer control descriptors, channel x and channel y, and manages all master bus-address calculations. All the channels provide the same functionality. This structure allows data transfers associated with one channel to be preempted after the completion of a read/write sequence if a higher priority channel activation is asserted while the first channel is active. After a channel is activated, it runs until the minor loop is completed, unless preempted by a higher priority channel. This provides a mechanism (enabled by DCHPRI*n*[ECP]) where a large data move operation can be preempted to minimize the time another channel is blocked from execution. When any channel is selected to execute, the contents of its TCD are read from local memory and loaded into the address path channel x registers for a normal start and into channel y registers for a preemption start. After the minor loop completes execution, the address path hardware writes the new values for the TCD*n*_WORD_x[SADDR, DADDR, CITER] back to local memory. If the major iteration count is exhausted, additional processing is performed, including the final address pointer updates, reloading the TCD*n*_WORD_5[CITER] field, and a possible fetch of the next TCD*n* from memory as part of a scatter/gather operation.
Data path	This block implements the bus master read/write datapath. It includes 32 bytes of register storage (matching the maximum transfer size) and the necessary multiplex logic to support any required data alignment. The AMBA-AHB read data bus is the primary input, and the AHB write data bus is the primary output. The address and data path modules directly support the 2-stage pipelined AMBA-AHB bus. The address path module represents the 1st stage of the bus pipeline (address phase), while the data path module implements the 2nd stage of the pipeline (data phase).
Program model/channel arbitration	This block implements the first section of the eDMA programming model as well as the channel arbitration logic. The programming model registers are connected to the slave bus (not shown). The ipd_req[n] inputs and dma_ipi_int[n] outputs are also connected to this module (via the control logic).
Control	This block provides all the control functions for the eDMA engine. For data transfers where the source and destination sizes are equal, the eDMA engine performs a series of source read/ destination write operations until the number of bytes specified in the minor loop byte count has moved. For descriptors where the sizes are not equal, multiple accesses of the smaller size data are required for each reference of the larger size. As an example, if the source size references 16-bit data and the destination is 32-bit data, two reads are performed, then one 32-bit write.

The transfer-control descriptor local memory is further partitioned as shown in the following table:

Table 18-804. Transfer control descriptor memory

Submodule	Description
Memory controller	This logic implements the required dual-ported controller, managing accesses from the eDMA engine as well as references from the slave bus. As noted earlier, in the event of simultaneous accesses, the eDMA engine is given priority and the peripheral slave bus transaction is stalled. The hooks to a BIST controller for the local TCD memory are included in this module.
Memory array	TCD storage is implemented using a single-port, synchronous RAM array.

18.6.2 eDMA basic data flow

The basic flow of a data transfer can be partitioned into three segments. As shown in the following figure, the first segment involves the channel service request. In the diagram, this example uses the assertion of the ipd_req[n] signal to request service for channel n. Channel service request via software and the TCDn_WORD_7[START] bit follows the same basic flow as an ipd_req. The ipd_req[n] input signal is registered internally and then routed to through the eDMA engine, first through the control module, then into the programming model/channel arbitration (pmodel_charb) module. In the next cycle, the channel arbitration is performed, either using the fixed-priority or round-robin algorithm. After the arbitration is complete, the activated channel number is sent through the address path (addr_path) and converted into the required address to access the TCD local memory. Next, the TCD memory is accessed and the required descriptor read from the local memory and loaded into the dma_engine.addr_path.channel_{x,y} registers. The TCD memory is organized 64 bits in width to minimize the time needed to fetch the activated channel's descriptor and load it into the dma_engine.addr_path.channel_{x,y} registers.

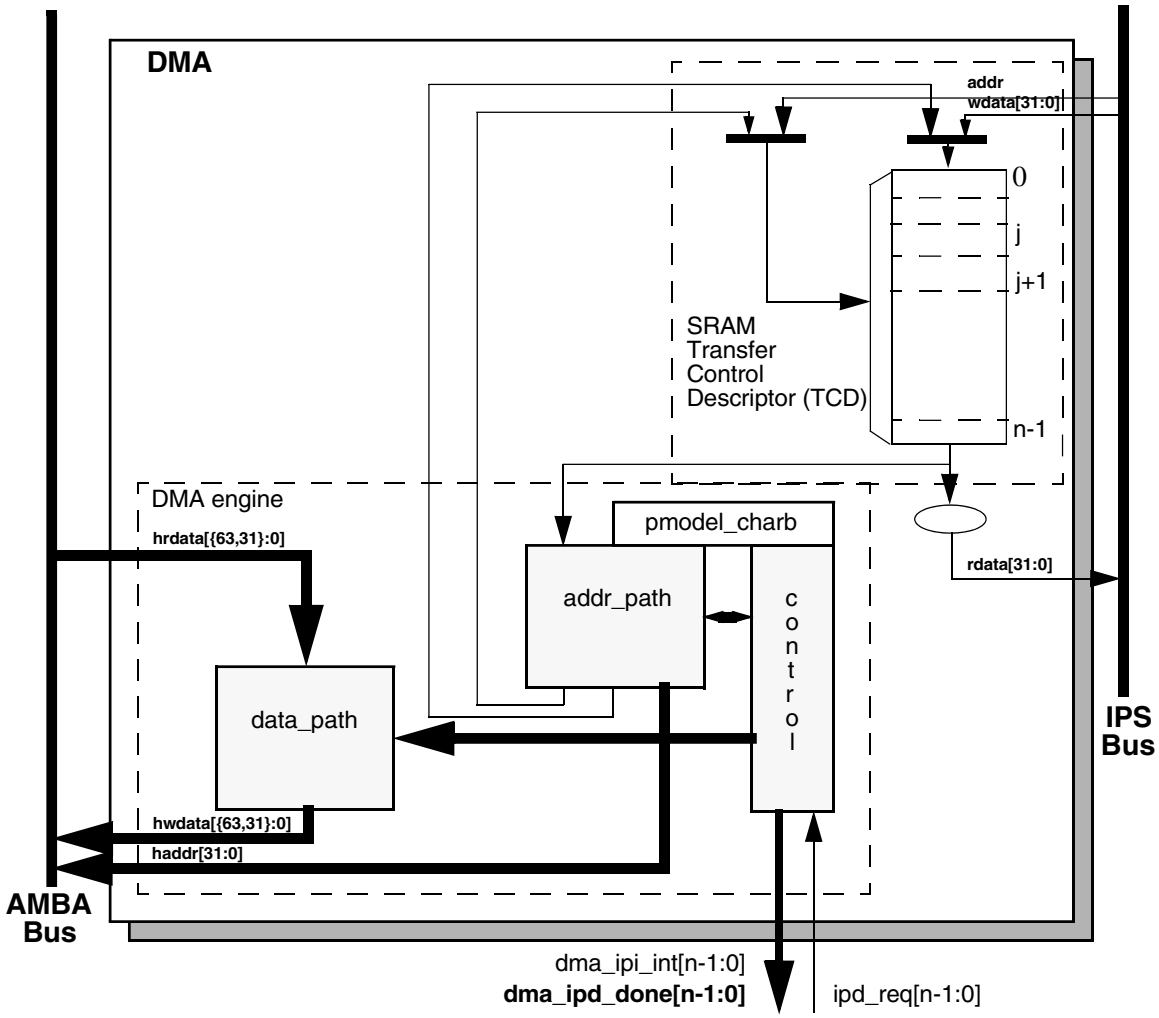


Figure 18-738. eDMA operation, part 1

In the second part of the basic data flow, as shown in the following figure, the modules associated with the data transfer (addr_path, data_path and control) sequence through the required source reads and destination writes to perform the actual data movement. The source reads are initiated and the fetched data is temporarily stored in the data_path module until it is gated onto the AMBA-AHB bus during the destination write. This source read/destination write processing continues until the inner minor byte count has been transferred. The dma_ipd_done[n] signal is asserted at the end of the minor byte count transfer.

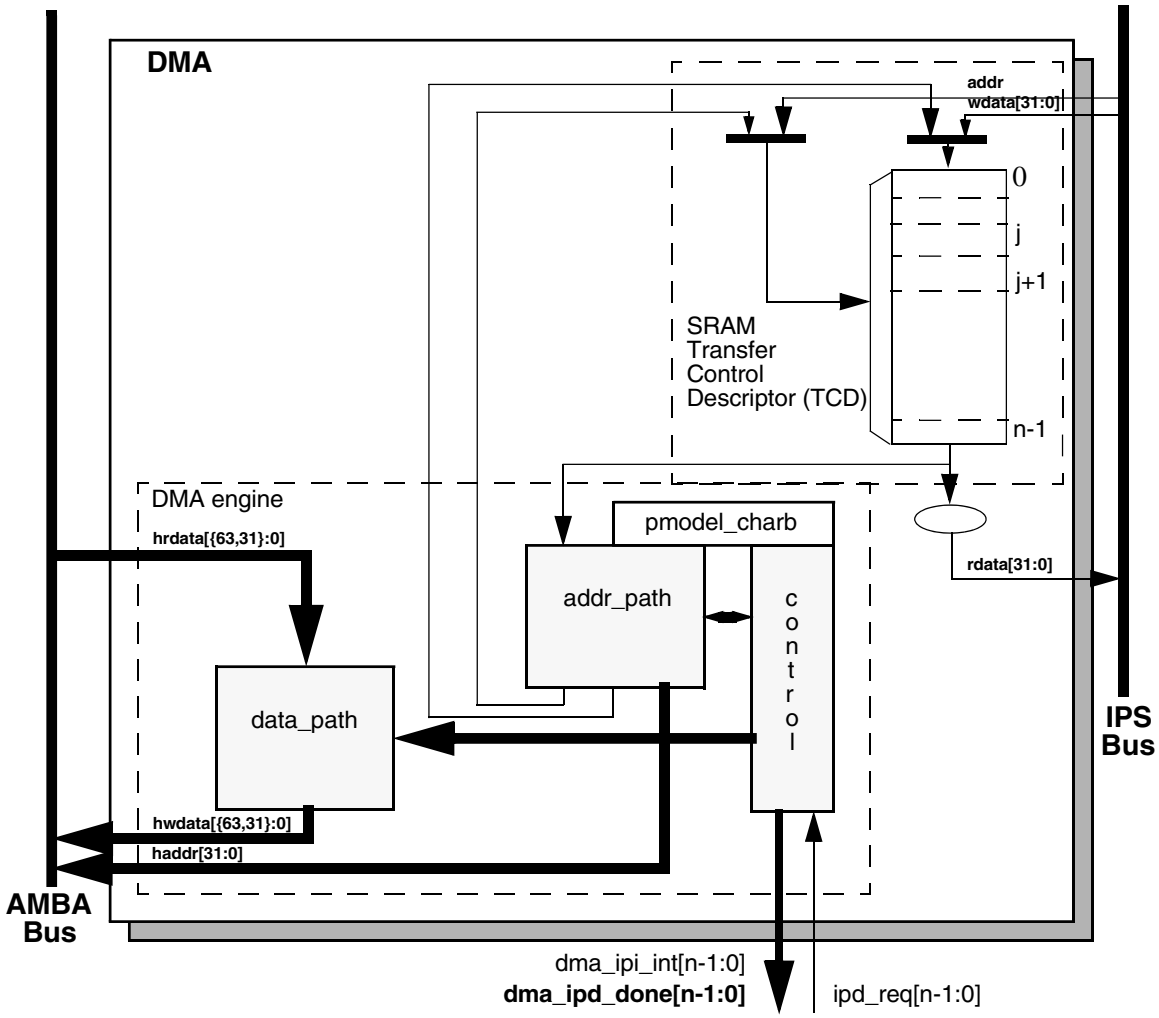


Figure 18-739. eDMA operation, part 2

Once the inner minor byte count has been moved, the final phase of the basic data flow is performed. In this segment, the addr_path logic performs the required updates to certain fields in the channel's TCD, for example, SADDT, DADDR, CITER. If the outer major iteration count is exhausted, then there are additional operations which are performed. These include the final address adjustments and reloading of the BITER field into the CITER. Additionally, assertion of an optional interrupt request occurs at this time, as does a possible fetch of a new TCD from memory using the scatter/gather address pointer included in the descriptor. The updates to the TCD memory and the assertion of an interrupt request are shown in the following figure.

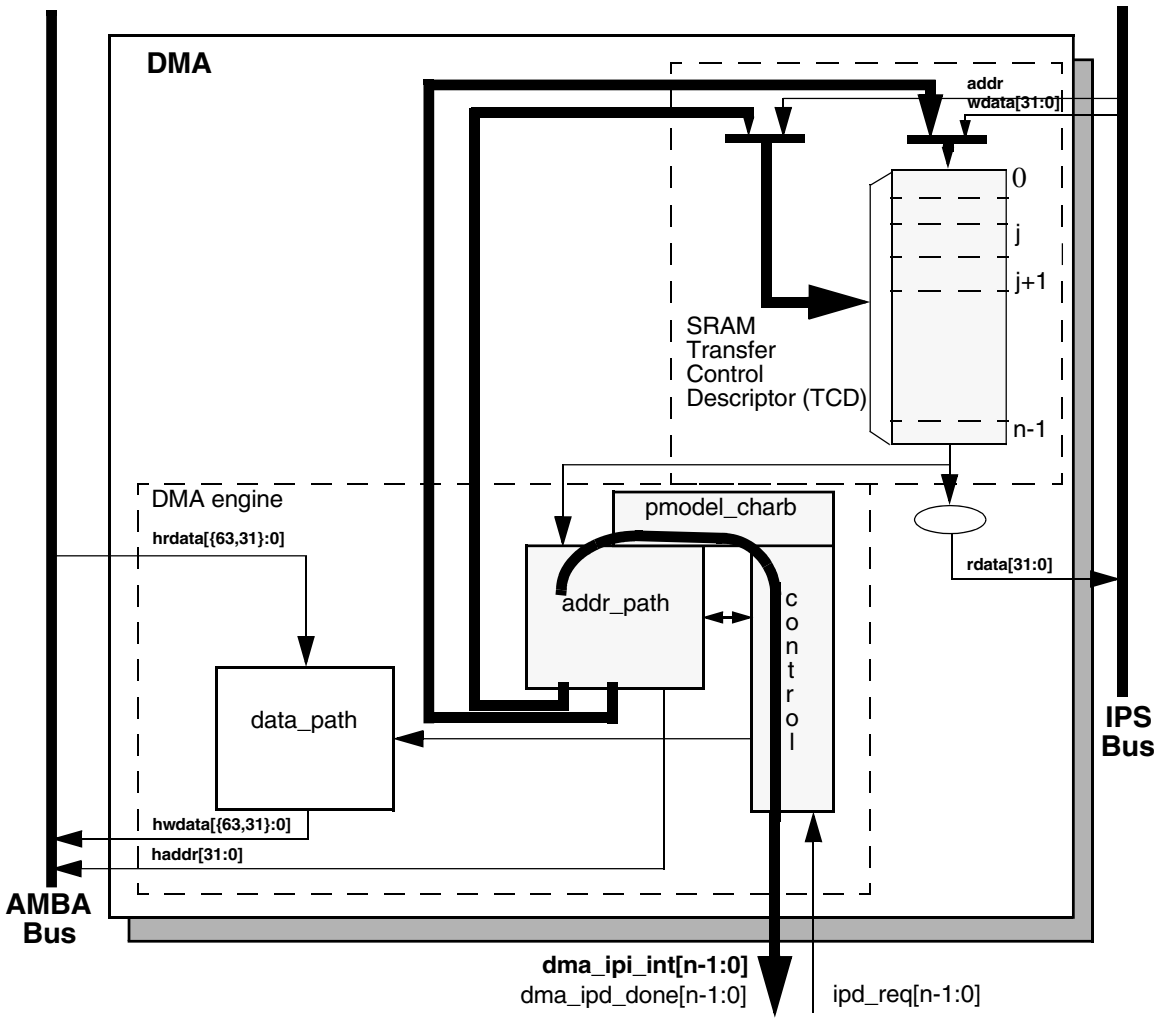


Figure 18-740. eDMA operation, part 3

18.6.3 eDMA performance

This section addresses the performance of the eDMA module, focusing on two separate metrics:

- In the traditional data movement context, performance is best expressed as the peak data transfer rates achieved using the eDMA. In most implementations, this transfer rate is limited by the speed of the source and destination address spaces.
- In a second context where device-paced movement of single data values to/from peripherals is dominant, a measure of the requests that can be serviced in a fixed time is a more relevant metric. In this environment, the speed of the source and destination

address spaces remains important. However, the microarchitecture of the eDMA also factors significantly into the resulting metric.

18.6.3.1 Peak transfer rates

The peak transfer rates for several different sources and destination transfers are shown in the following tables. These tables assume:

- Internal SRAM can be accessed with zero wait-states when viewed from the system bus data phase
- All internal peripheral bus reads require two wait-states, and internal peripheral bus writes three wait-states, when viewed from the system bus data phase
- All internal peripheral bus accesses are 32 bits in size

This table compares peak transfer rates based on different possible system speeds. Specific chips/devices may not support all system speeds listed.

Table 18-805. eDMA peak transfer rates (MB/s)

System Speed, Width	Internal SRAM-to-Internal SRAM	32b internal peripheral bus-to-Internal SRAM	Internal SRAM-to-32b internal peripheral bus
66.7 MHz, 32b	133.3	66.7	53.3
83.3 MHz, 32b	166.7	83.3	66.7
100.0 MHz, 32b	200.0	100.0	80.0
133.3 MHz, 32b	266.7	133.3	106.7
150.0 MHz, 32b	300.0	150.0	120.0

Internal-SRAM-to-internal-SRAM transfers occur at the core's datapath width. For all transfers involving the internal peripheral bus, 32-bit transfer sizes are used. In all cases, the transfer rate includes the time to read the source plus the time to write the destination.

18.6.3.2 Peak request rates

The second performance metric is a measure of the number of DMA requests that can be serviced in a given amount of time. For this metric, assume that the peripheral request causes the channel to move a single internal peripheral bus-mapped operand to/from internal SRAM. The same timing assumptions used in the previous example apply to this calculation. In particular, this metric also reflects the time required to activate the channel.

The eDMA design supports the following hardware service request sequence. Note that the exact timing from Cycle 7 is a function of the response times for the channel's read and write accesses. In the case of an internal peripheral bus read and internal SRAM write, the combined data phase time is 4 cycles. For an SRAM read and internal peripheral bus write, it is 5 cycles.

Table 18-806. Hardware service request process

Cycle		Description
With internal peripheral bus read and internal SRAM write	With SRAM read and internal peripheral bus write	
1		eDMA peripheral request is asserted.
2		The eDMA peripheral request is registered locally in the eDMA module and qualified. TCD*n*_WORD_7[START] bit initiated requests start at this point with the registering of the user write to TCD*n*_WORD_7.
3		Channel arbitration begins.
4		Channel arbitration completes. The transfer control descriptor local memory read is initiated.
5–6		The first two parts of the activated channel's TCD is read from the local memory. The memory width to the eDMA engine is 64 bits, so the entire descriptor can be accessed in four cycles
7		The first system bus read cycle is initiated, as the third part of the channel's TCD is read from the local memory. Depending on the state of the crossbar switch, arbitration at the system bus may insert an additional cycle of delay here.
8–11	8–12	The last part of the TCD is read in. This cycle represents the first data phase for the read, and the address phase for the destination write.
12	13	This cycle represents the data phase of the last destination write.
13	14	The eDMA engine completes the execution of the inner minor loop and prepares to write back the required TCD*n* fields into the local memory. The TCD*n*_WORD_7 is read and checked for channel linking or scatter/gather requests.
14	15	The appropriate fields in the first part of the TCD*n* are written back into the local memory.
15	16	The fields in the second part of the TCD*n* are written back into the local memory. This cycle coincides with the next channel arbitration cycle start.
16	17	The next channel to be activated performs the read of the first part of its TCD from the local memory. This is equivalent to Cycle 4 for the first channel's service request.

Assuming zero wait states on the system bus, DMA requests can be processed every 9 cycles. Assuming an average of the access times associated with internal peripheral bus-to-SRAM (4 cycles) and SRAM-to-internal peripheral bus (5 cycles), DMA requests can

be processed every 11.5 cycles (4 + (4+5)/2 + 3). This is the time from Cycle 4 to Cycle x +5. The resulting peak request rate, as a function of the system frequency, is shown in the following table.

Table 18-807. eDMA peak request rate (Millions of requests per second)

System frequency (MHz)	Request rate with zero wait states	Request rate with wait states
66.6	7.4	5.8
83.3	9.2	7.2
100.0	11.1	8.7
133.3	14.8	11.6
150.0	16.6	13.0

A general formula to compute the peak request rate with overlapping requests is:

PEAKreq = freq / [entry + (1 + read_ws) + (1 + write_ws) + exit], where terms are defined in the following table.

Table 18-808. Peak request formula operands

Operand	Description
PEAKreq	Peak request rate
freq	System frequency
entry	Channel startup (4 cycles)
read_ws	Wait states seen during the system bus read data phase
write_ws	Wait states seen during the system bus write data phase
exit	Channel shutdown (3 cycles)

18.6.3.3 eDMA performance example

Consider a system with the following characteristics:

- Internal SRAM can be accessed with one wait-state when viewed from the system bus data phase
- All internal peripheral bus reads require two wait-states, and internal peripheral bus writes three wait-states viewed from the system bus data phase
- System operates at 150 MHz

For an SRAM to internal peripheral bus transfer,

PEAKreq = 150 MHz / [4 + (1 + 1) + (1 + 3) + 3] cycles = 11.5 Mreq/sec

For an internal peripheral bus to SRAM transfer,

PEAKreq = 150 MHz / [4 + (1 + 2) + (1 + 1) + 3] cycles = 12.5 Mreq/sec

Assuming an even distribution of the two transfer types, the average peak request rate would be:

PEAKreq = (11.5 Mreq/sec + 12.5 Mreq/sec) / 2 = 12.0 Mreq/sec

The minimum number of cycles to perform a single read/write, zero wait states on the system bus, from a cold start where no channel is executing and eDMA is idle are:

- 11 cycles for a software, that is, a TCDn_WORD_7[START] bit, request
- 12 cycles for a hardware, that is, an eDMA peripheral request signal, request

Two cycles account for the arbitration pipeline and one extra cycle on the hardware request resulting from the internal registering of the eDMA peripheral request signals. For the peak request rate calculations above, the arbitration and request registering is absorbed in or overlaps the previous executing channel.

Note

When channel linking or scatter/gather is enabled, a two cycle delay is imposed on the next channel selection and startup. This allows the link channel or the scatter/gather channel to be eligible and considered in the arbitration pool for next channel selection.

18.7 Initialization/application information

18.7.1 eDMA initialization

A typical initialization of the eDMA is:

1. Write the CR register if a configuration other than the default is desired.
2. Write the channel priority levels into the DCHPRIn registers if a configuration other than the default is desired.
3. Enable error interrupts in the EEI registers if so desired.
4. Write the 32-byte TCD for each channel that may request service.
5. Enable any hardware service requests via the ERQ register.
6. Request channel service by either software (setting the TCDn_WORD_7[START] bit) or by hardware (slave device asserting its ipd_req signal).

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels written into the programmer's model. The eDMA engine will read the entire TCD for the selected channel into its internal address path module. As the TCD is being read, the first transfer is initiated on the AHB bus unless a configuration error is detected. Transfers from the source (as defined by the source address, TCDn_WORD_0[SADDR] to the destination (as defined by the destination address, TCDn_WORD_4[DADDR] continue until the specified number of bytes (TCDn_WORD_2[NBYTES] have been transferred. When the transfer is complete, the eDMA engine's local TCDn_WORD_0[SADDR], TCDn_WORD_4[DADDR], and TCDn_WORD_5[CITER] are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post processing is executed, that is, interrupts, major loop channel linking, and scatter/gather operations, if enabled.

18.7.2 eDMA programming errors

The eDMA performs various tests on the Transfer Control Descriptor to verify consistency in the descriptor data. Most programming errors are reported on a per channel basis with the exception of two errors; Group Priority Error and Channel Priority Error, GPE and CPE in the ES register respectively.

For all error types other than Group or Channel Priority Errors, the channel number causing the error is recorded in the ES register. If the error source is not removed before the next activation of the problem channel, the error will be detected and recorded again.

The sequence listed below is correct. For item 2, the dma_ipd_ack{done} lines will assert only if the selected channel is requesting service via the ipd_req signal. I think the typical application will enable error interrupts for all channels. So the user will get an error interrupt, but the channel number for the ERR register and the error interrupt request line may be wrong because they reflect the selected channel.

Channel priority errors are identified within a group after that group has been selected as the active group. For example:

1. The eDMA is configured for fixed group and fixed channel arbitration modes.
2. Group3 is the highest priority and all channels are unique in that group.
3. Group2 is the next highest priority and has two channels with the same priority level.
4. If Group3 has any service requests, those requests will be executed.
5. Once all of the Group3 requests have completed, Group2 will be the next active group.
6. If Group2 has a service request, then an undefined channel in Group2 will be selected and a channel priority error will occur.

7. This will repeat until the all of Group2 requests have been removed or a higher priority Group3 request comes in.

A group priority error is global and any request in any group will cause a group priority error.

In general, if priority levels are not unique, the highest (channel/group) priority that has an active request will be selected, but the lowest numbered (channel/group) with that priority will be selected by arbitration and executed by the eDMA engine. The hardware service request handshake signals, error interrupts and error reporting will be associated with the selected channel.

18.7.3 eDMA arbitration mode considerations

18.7.3.1 Fixed group arbitration, fixed channel arbitration

In this mode, the channel service request from the highest priority channel in the highest priority group will be selected to execute. If the eDMA is so programmed that the channels within one group use “fixed” priorities, and that group is assigned the highest “fixed” priority of all groups, it is possible for that group to take all the bandwidth of the eDMA controller — that is, no other groups will be serviced if there is always at least one eDMA request pending on a channel in the highest priority group when the controller arbitrates the next eDMA request.

The advantage of this scenario is that latency can be small for channels that need to be serviced quickly.

Preemption is available in this scenario only.

18.7.3.2 Round-robin group arbitration, fixed channel arbitration

The occurrence of one or more eDMA requests from one or more groups, the channel with the highest priority from a specific group will be serviced first. Groups are serviced starting with the highest group number with a service request and rotating through to the lowest group number containing a service request.

Once the channel request is serviced, the group round-robin algorithm will select the highest pending request from the next group in the round-robin sequence. Servicing continues round robin, always servicing the highest priority channel in the next group in the sequence, or just skipping a group if it has no pending requests.

If a channel requests service at a rate that equals or exceeds the round-robin service rate, then that channel will always be serviced before lower priority channels in the same group, and thus the lower priority channels will never be serviced.

The advantage of this scenario is that a single group cannot use all the eDMA bandwidth.

The highest priority channel selection latency is potentially greater than fixed/fixed arbitration.

Excessive request rates on high priority channels can prevent the servicing of lower priority channels in the same group.

18.7.3.3 Round-robin group arbitration, round-robin channel arbitration

Groups will be serviced as described in Round-robin group arbitration, fixed channel arbitration, but this time channels will be serviced in channel number order. Only one channel is serviced from each requesting group for each round-robin pass through the groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to channel priority levels.

Because channels are serviced in round-robin manner, any channel that generates eDMA requests faster than a combination of the group round robin service rate and the channel service rate for its group will not prevent the servicing of other channels in its group. Any eDMA requests that are not serviced are simply lost, but at least one channel will be serviced.

This scenario ensures that all channels will be guaranteed service at some point, regardless of the request rates. However, the potential latency could be quite high.

All channels are treated equally. Priority levels are not used in round-robin/round-robin mode.

18.7.3.4 Fixed group arbitration, round-robin channel arbitration

The highest priority group with a request will be serviced. Lower priority groups will be serviced if no pending requests exist in the higher priority groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels assigned within the group.

This scenario could cause the same bandwidth consumption problem as indicated in Fixed group arbitration, fixed channel arbitration, but all the channels in the highest priority group will be serviced.

Service latency will be short on the highest priority group, but can become much longer as the group priority decreases.

18.7.4 Transfer

18.7.4.1 Single request

To perform a simply transfer of ‘n’ bytes of data with one activation, set the major loop to one (TCDn_WORD_5[CITER] = TCDn_WORD_7[BITER] = 1). The data transfer will begin after the channel service request is acknowledged and the channel is selected to execute. Once the transfer is complete, the TCDn_WORD_7[DONE] bit will be set and an interrupt will be generated if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The eDMA is programmed for one iteration of the major loop transferring 16 bytes per iteration. The source memory has a byte wide memory port located at 0x1000. The destination memory has a word wide port located at 0x2000. The address offsets are programmed in increments to match the size of the transfer; one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

- TCDn_WORD_5[CITER] = TCDn_WORD_7[BITER] = 1
- TCDn_WORD_2[NBYTES] = 16
- TCDn_WORD_0[SADR] = 0x1000
- TCDn_WORD_1[SOFF] = 1
- TCDn_WORD_1[SSIZE] = 0
- TCDn_WORD_3[SLAST] = -16
- TCDn_WORD_4[DADDR] = 0x2000
- TCDn_WORD_5[DOFF] = 4
- TCDn_WORD_1[DSIZE] = 2
- TCDn_WORD_6[DLAST_SGA] = -16
- TCDn_WORD_7[INT_MAJOR] = 1
- TCDn_WORD_7[START] = 1 (TCDn_WORD_7 should be written last after all other fields have been initialized)
- All other TCD fields = 0

The eDMA is programmed for one iteration of the major loop transferring 16 bytes per iteration. The source memory has a byte wide memory port located at 0x1000. The destination memory has a word wide port located at 0x2000. The address offsets are programmed in increments to match the size of the transfer; one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

The TCD values above generate the following sequence of events:

1. Slave bus write to the TCDn_WORD_7[START] bit requests channel service
2. The channel is selected by arbitration for servicing
3. eDMA engine writes:
   - TCDn_WORD_7[DONE] = 0,
   - TCDn_WORD_7[START] = 0,
   - TCDn_WORD_7[ACTIVE] = 1
4. eDMA engine reads: channel TCD data from local memory to internal register file
5. The source to destination transfers are executed as follows:
   a. read_byte(0x1000), read_byte(0x1001), read_byte(0x1002), read_byte(0x1003)
   b. write_word(0x2000) -> first iteration of the minor loop
   c. read_byte(0x1004), read_byte(0x1005), read_byte(0x1006), read_byte(0x1007)
   d. write_word(0x2004) -> second iteration of the minor loop
   e. read_byte(0x1008), read_byte(0x1009), read_byte(0x100A), read_byte(0x100B)
   f. write_word(0x2008) -> third iteration of the minor loop
   g. read_byte(0x100C), read_byte(0x100D), read_byte(0x100E), read_byte(0x100F)
   h. write_word(0x200C) -> last iteration of the minor loop -> major loop complete
6. eDMA engine writes:
   - TCDn_WORD_0[SADR] = 0x1000,
   - TCDn_WORD_4[DADDR] = 0x2000,
   - TCDn_WORD_5[CITER] = 1 (TCDn_WORD_7[BITER])
7. eDMA engine writes:
   - TCDn_WORD_7[ACTIVE] = 0,
   - TCDn_WORD_7[DONE] = 1,
   - INT[n] = 1
8. The channel retires

The eDMA goes idle or services next channel.

18.7.4.2 Multiple requests

The next example is the same as the previous example, with the exception of transferring 32 bytes via two hardware requests. The only fields that change are the major loop iteration count and the final address offsets. The eDMA is programmed for two iterations

of the major loop transferring 16 bytes per iteration. After the channel's hardware requests is enabled in the ERQ register, channel service requests are initiated by the slave device.

- TCDn_WORD_5[CITER] = TCD.biter = 2
- TCDn_WORD_3[SLAST] = -32
- TCDn_WORD_6[DLAST_SGA] = -32

This would generate the following sequence of events:

1. First hardware (ipd_req) request for channel service
2. The channel is selected by arbitration for servicing
3. eDMA engine writes:
   - TCDn_WORD_7[DONE] = 0,
   - TCDn_WORD_7[START] = 0,
   - TCDn_WORD_7[ACTIVE] = 1
4. eDMA engine reads: channel TCD data from local memory to internal register file
5. The source to destination transfers are executed as follows:
   a. read_byte(0x1000), read_byte(0x1001), read_byte(0x1002), read_byte(0x1003)
   b. write_word(0x2000) ≥ first iteration of the minor loop
   c. read_byte(0x1004), read_byte(0x1005), read_byte(0x1006), read_byte(0x1007)
   d. write_word(0x2004) ≥ second iteration of the minor loop
   e. read_byte(0x1008), read_byte(0x1009), read_byte(0x100A), read_byte(0x100B)
   f. write_word(0x2008) ≥ third iteration of the minor loop
   g. read_byte(0x100C), read_byte(0x100D), read_byte(0x100E), read_byte(0x100F)
   h. write_word(0x200C) ≥ last iteration of the minor loop
6. eDMA engine writes:
   - TCDn_WORD_0[SADDR] = 0x1010,
   - TCDn_WORD_4[DADDR] = 0x2010,
   - TCDn_WORD_5[CITER] = 1
7. eDMA engine writes: TCDn_WORD_7[ACTIVE] = 0
8. The channel retires ≥ one iteration of the major loop

   The eDMA goes idle or services the next channel.

9. Second hardware (ipd_req) requests channel service
10. The channel is selected by arbitration for servicing
11. eDMA engine writes:
    - TCDn_WORD_7[DONE] = 0,
    - TCDn_WORD_7[START] = 0,
    - TCDn_WORD_7[ACTIVE] = 1
12. eDMA engine reads: channel TCD data from local memory to internal register file
13. The source to destination transfers are executed as follows:
    a. read_byte(0x1010), read_byte(0x1011), read_byte(0x1012), read_byte(0x1013)
    b. write_word(0x2010) ≥ first iteration of the minor loop

c. read_byte(0x1014), read_byte(0x1015), read_byte(0x1016), read_byte(0x1017)
d. write_word(0x2014) ≥ second iteration of the minor loop
e. read_byte(0x1018), read_byte(0x1019), read_byte(0x101A), read_byte(0x101B)
f. write_word(0x2018) ≥ third iteration of the minor loop
g. read_byte(0x101C), read_byte(0x101D), read_byte(0x101E), read_byte(0x101F)
h. write_word(0x201C) ≥ last iteration of the minor loop ≥ major loop complete

14. eDMA engine writes:
    - TCDn_WORD_0[SADDR] = 0x1000,
    - TCDn_WORD_4[DADDR] = 0x2000,
    - TCDn_WORD_5[CITER] = 2 (TCDn_WORD_5TCDn_WORD_7[BITER])
15. eDMA engine writes:
    - TCDn_WORD_7[ACTIVE] = 0,
    - TCDn_WORD_7[DONE] = 1,
    - INT[n] = 1
16. The channel retires ≥ major loop complete.

    The eDMA goes idle or services the next channel.

18.7.5 TCD status

18.7.5.1 Minor loop complete

There are two methods to test for minor loop completion when using software initiated service requests. The first method is to read the TCD.citer field and test for a change. Another method may be extracted from the sequence shown below. The second method is to test the TCD.start bit AND the TCD.active bit. The minor loop complete condition is indicated by both bits reading zero after the TCD.start was written to a one. Polling the TCD.active bit may be inconclusive because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute the following sequence for a software activated channel:

1. TCD.start = 1, TCD.active = 0, TCD.done = 0 (channel service request via software)
2. TCD.start = 0, TCD.active = 1, TCD.done = 0 (channel is executing)
3. Either TCD.start = 0, TCD.active = 0, TCD.done = 0 (channel has completed the minor loop and is idle) or TCD.start = 0, TCD.active = 0, TCD.done = 1 (channel has completed the major loop and is idle)

The best method to test for minor loop completion when using hardware initiated service requests is to read the TCD.citer field and test for a change. The hardware request and acknowledge handshakes signals are not visible in the programmer's model.

The TCD status bits execute the following sequence for a hardware activated channel:

1. ipd_req asserts (channel service request via hardware)
2. TCD.start = 0, TCD.active = 1, TCD.done = 0 (channel is executing)
3. Either TCD.start = 0, TCD.active = 0, TCD.done = 0 (channel has completed the minor loop and is idle) or TCD.start = 0, TCD.active = 0, TCD.done = 1 (channel has completed the major loop and is idle)

For both activation types, the major loop complete status is explicitly indicated via the TCD.done bit.

The TCD.start bit is cleared automatically when the channel begins execution regardless of how the channel was activated.

18.7.5.2 Active channel TCD reads

The eDMA will read back the 'true' TCD.saddr, TCD.daddr, and TCD.nbytes values if read while a channel is executing. The 'true' values of the saddr, daddr, and nbytes are the values the eDMA engine is currently using in its internal register file and not the values in the TCD local memory for that channel. The addresses (saddr and daddr) and nbytes (decrements to zero as the transfer progresses) can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

18.7.5.3 Preemption status

Preemption is only available when fixed arbitration is selected for both group and channel arbitration modes. A preempt-able situation is one in which a preempt-enabled channel is running and a higher priority request becomes active. When the eDMA engine is not operating in fixed group, fixed channel arbitration mode, the determination of the relative priority of the actively running and the outstanding requests become undefined. Channel and/or group priorities are treated as equal (constantly rotating) when round-robin arbitration mode is selected.

The TCD.active bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting channel executes one iteration of the major loop. Two TCD.active bits set at the same time in the overall TCD map indicates a higher priority channel is actively preempting a lower priority channel.

The worst case latency when switching to a preempt channel is the summation of:

- arbitration latency (2 cycles)
- bandwidth control stalls (if enabled)
- the time to execute two read/write sequences (including AHB bus holds; a system dependency driven by the slave devices or the crossbar).

18.7.6 Channel linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD.start bit of another channel (or itself) thus initiating a service request for that channel. This operation is automatically performed by the eDMA engine at the conclusion of the major or minor loop when properly enabled.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD.citer.e_link field are used to determine whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, with the initial fields of:

- TCD.citer.e_link = 1
- TCD.citer.linkch = 0xC
- TCD.citer value = 0x4
- TCD.major.e_link = 1
- TCD.major.linkch = 0x7

will execute as:

1. minor loop done -> set channel 12 TCD.start bit
2. minor loop done -> set channel 12 TCD.start bit
3. minor loop done -> set channel 12 TCD.start bit
4. minor loop done, major loop done -> set channel 7 TCD.start bit

When minor loop linking is enabled (TCD.citer.e_link = 1), the TCD.citer field uses a nine bit vector to form the current iteration count.

When minor loop linking is disabled (TCD.citer.e_link = 0), the TCD.citer field uses a 15 bit vector to form the current iteration count. The bits associated with the TCD.citer.linkch field are concatenated onto the citer value to increase the range of the citer.

NOTE

The TCD.citer.e_link bit and the TCD.biter.e_link bit must equal or a configuration error will be reported. The citer and biter vector widths must be equal to calculate the major loop, half-way done interrupt point.

18.7.7 Dynamic programming

18.7.7.1 Dynamic priority changing

The following two options are recommended for dynamically changing channel priority levels:

1. Switch to round-robin channel arbitration mode, change the channel priorities, then switch back to fixed arbitration mode.
2. Disable all the channels within a group, then change the channel priorities within that group only, then enable the appropriate channels.

The following two options are available for dynamically changing group priority levels:

1. Switch to round-robin group arbitration mode, change the group priorities, then switch back to fixed arbitration mode.
2. Disable all channels, change the group priorities, then enable the appropriate channels.

18.7.7.2 Dynamic channel linking

Dynamic channel linking is the process of setting the TCD.major.e_link bit during channel execution. This bit is read from the TCD local memory at the end of channel execution, thus allowing the user to enable the feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic channel link by enabling the TCD.major.e_link bit at the same time the eDMA engine is retiring the channel. The TCD.major.e_link would be set in the programmer's model, but it would be unclear whether the actual link was made before the channel retired.

The coherency model in the following table is recommended when executing a dynamic channel link request.

Step #	Action
1	Write 1b to the TCD.major.e_link bit.
2	Read back the TCD.major.e_link bit.
3	Test the TCD.major.e_link request status: • If TCD.major.e_link = 1b, the dynamic link attempt was successful. • If TCD.major.e_link = 0b, the attempted dynamic link did not succeed (the channel was already retiring).

For this request, the TCD local memory controller forces the TCD.major.e_link bit to zero on any writes to a channel's TCD.word7 after that channel's TCD.done bit is set, indicating the major loop is complete.

NOTE

The user must clear the TCD.done bit before writing the TCD.major.e_link bit. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.

18.7.7.3 Dynamic scatter/gather

Dynamic scatter/gather is the process of setting the TCD.e_sg bit during channel execution. This bit is read from the TCD local memory at the end of channel execution, thus allowing the user to enable the feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic scatter/gather operation by enabling the TCD.e_sg bit at the same time the eDMA engine is retiring the channel. The TCD.e_sg would be set in the programmer's model, but it would be unclear whether the actual scatter/gather request was honored before the channel retired.

Two methods for this coherency model are shown in the following subsections. Method 1 has the advantage of reading the major.linkch field and the e_sg bit with a single read. For both dynamic channel linking and scatter/gather requests, the TCD local memory controller forces the TCD.major.e_link and TCD.e_sg bits to zero on any writes to a channel's TCD.word7 if that channel's TCD.done bit is set indicating the major loop is complete.

NOTE

The user must clear the TCD.done bit before writing the TCD.major.e_link or TCD.e_sg bits. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.

18.7.7.3.1 Method 1 (channel not using major loop channel linking)

For a channel not using major loop channel linking, the coherency model in the table below may be used for a dynamic scatter/gather request.

When the TCD.major.e_link bit is zero, the TCD.major.linkch field is not used by the eDMA. In this case, the TCD.major.linkch bits may be used for other purposes. This method uses the TCD.major.linkch field as a TCD identification (ID).

Step	Action
1	When the descriptors are built, write a unique TCD ID in the TCD.major.linkch field for each TCD associated with a channel using dynamic scatter/gather.
2	Write 1b to theTCD.d_req bit. **NOTE:** Should a dynamic scatter/gather attempt fail, setting the d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offest value.
3	Write theTCD.dlast_sga field with the scatter/gather address.
4	Write 1b to the TCD.e_sg bit.
5	Read back the 16 bit TCD control/status field.
6	Test the TCD.e_sg request status and TCD.major.linkch value: • If e_sg = 1b, the dynamic link attempt was successful. • If e_sg = 0b and the major.linkch (ID) did not change, the attempted dynamic link did not succeed (the channel was already retiring). • If e_sg = 0b and the major.linkch (ID) changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

18.7.7.3.2 Method 2 (channel using major loop linking)

For a channel using major loop channel linking, the coherency model in the following table may be used for a dynamic scatter/gather request. This method uses the TCD.dlast_sga field as a TCD identification (ID).

Table 18-811. Coherency model for Method 2

Step	Action
1	Write 1b to theTCD.d_req bit. **NOTE:** Should a dynamic scatter/gather attempt fail, setting the d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offest value.
2	Write theTCD.dlast_sga field with the scatter/gather address.
3	Write 1b to the TCD.e_sg bit.
4	Read back the TCD.e_sg bit.
5	Test the TCD.e_sg request status: • If e_sg = 1b, the dynamic link attempt was successful. • If e_sg = 0b, read the 32 bit TCD dlast_sga field. • If e_sg = 0b and the dlast_sga did not change, the attempted dynamic link did not succeed (the channel was already retiring). • If e_sg = 0b and the dlast_sga changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

18.7.7.4 Hardware request release timing

This section provides a timing diagram for deasserting the ipd_req hardware request signal. The following figure shows two read write sequences with shadowing that indicates the release of the ipd_req hardware request signal.

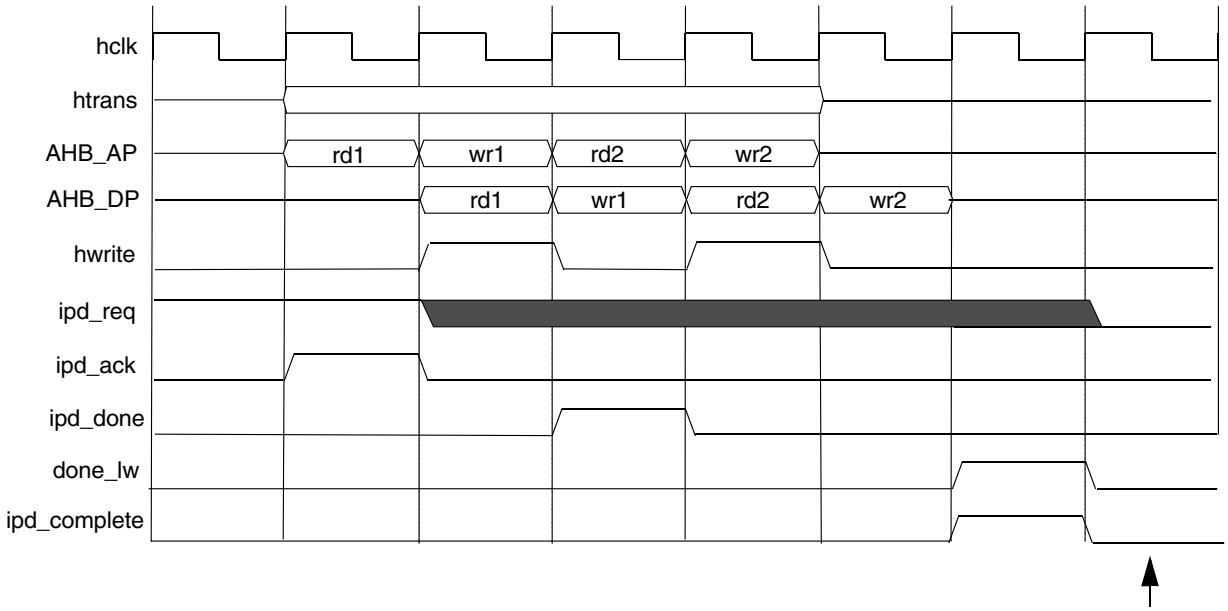


Figure 18-741. ipd_req hardware handshake

Chapter 19
Direct Memory Access Multiplexer (DMACHMUX)

19.1 DMACHMUX chip-specific module information

The peripheral DMA requests are serviced by a DMA controller. A static multiplexing scheme allows the support of more peripheral requests than available DMA channels.

19.1.1 DMA channel assignment

This device has four multiplexers. The following table lists which DMA channels apply to each multiplex.

Table 19-1. DMA multiplexer and channel list

DMAMUX instance	DMA channel
0	0-15
1	16-31
2	32-47
3	48-63

19.1.2 DMA channel multiplexing block diagram

The connections of the DMA channel multiplexors to the eDMA are shown in the following figure. One DMAMUX is instantiated for every 16 DMA channels.

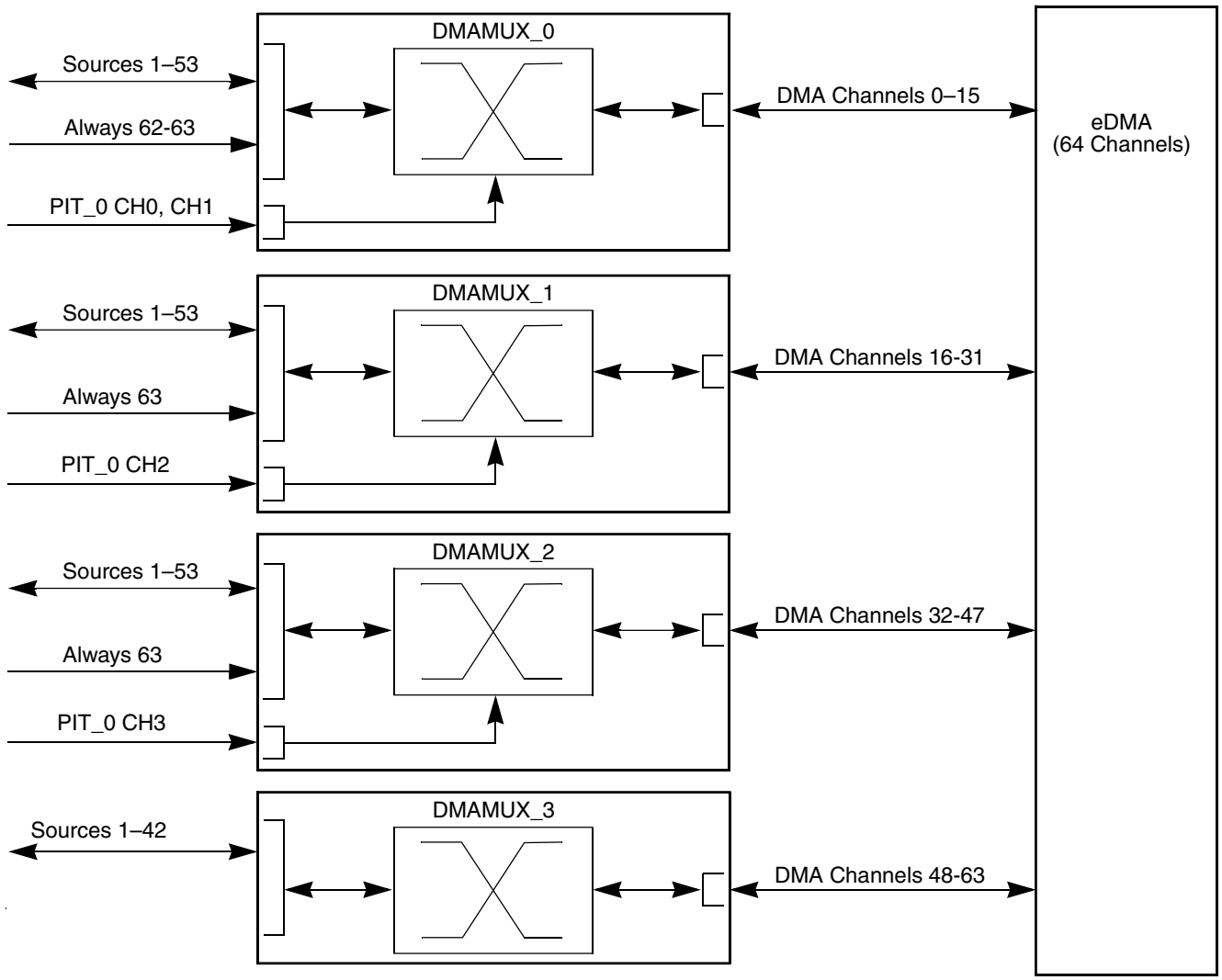


Figure 19-1. DMA channel multiplexing

19.1.3 DMA request mapping

The mapping of the peripheral DMA requests to the DMA_CH_MUX input sources is shown in the following table.

Table 19-2. DMA request mapping

Source	DMAMUX_0	DMAMUX_1	DMAMUX_2	DMAMUX_3
0	Reserved2	Reserved2	Reserved2	Reserved2
1	SAR_ADC_0	SAR_ADC_2	BCTU_0	SAR_ADC_1
2	BCTU_2	BCTU_1	ADC_SD_2 EOC	SAR_ADC_3
3	ADC_SD_0 EOC	BCTU_3	DECFIL_1 DRAIN_BUF	ADC_SD_1 EOC
4	DECFIL_0 DRAIN_BUF	DSPI_2 CMD	DECFIL_1 FILL_BUF	DSPI_1 CMD

Table continues on the next page...

Table 19-2. DMA request mapping (continued)

Source	DMAMUX_0	DMAMUX_1	DMAMUX_2	DMAMUX_3
5	DECFIL_0 FILL_BUF	DSPI_2 RX	DSPI_4 CMD	DSPI_1 RX
6	DSPI_0 CMD	DSPI_2 TX	DSPI_4 RX	DSPI_1 TX
7	DSPI_0 RX	DSPI_M0 CMD	DSPI_4 Tx	DSPI_3 CMD
8	DSPI_0 TX	DSPI_M0 RX	LINFlex_2 RX	DSPI_3 RX
9	LINFlex_0 RX	DSPI_M0 TX	LINFlex_2 TX	DSPI_3 TX
10	LINFlex_0 TX	LINFlex_M0 RX	SENT_0 RX FAST	DSPI_M1 CMD
11	SIUL2 REQ0	LINFlex_M0 TX	SENT_0 RX SLOW	DSPI_M1 RX
12	SIUL2 REQ1	SIUL2 REQ2	SIUL2 REQ4	DSPI_M1 TX
13	SIUL2 REQ6	SIUL2 REQ3	SIUL2 REQ5	LINFlex_1 RX
14	Reserved	SIUL2 REQ7	FlexCAN_0	LINFlex_1 TX
15	eMIOS_0 CH8	FlexCAN_2	eMIOS_0 CH18	LINFlex_3 RX
16	eMIOS_0 CH9	eMIOS_0 CH13	eMIOS_0 CH19	LINFlex_3 TX
17	eMIOS_0 CH10	eMIOS_0 CH14	eMIOS_0 CH20	LINFlex_M1 RX
18	eMIOS_0 CH11	eMIOS_0 CH15	eMIOS_0 CH21	LINFlex_M1 TX
19	eMIOS_0 CH12	eMIOS_0 CH16	eMIOS_0 CH22	SENT_1 RX FAST
20	eTPU2_A CH0	eMIOS_0 CH17	eMIOS_0 CH23	SENT_1 RX SLOW
21	eTPU2_A CH1	eTPU2_A CH5	eTPU2_A CH10	SIPI CH0
22	eTPU2_A CH2	eTPU2_A CH6	eTPU2_A CH11	SIPI CH1
23	eTPU2_A CH3	eTPU2_A CH7	eTPU2_A CH12	SIPI CH2
24	eTPU2_A CH4	eTPU2_A CH8	eTPU2_A CH13	SIPI CH3
25	eTPU2_A CH15	eTPU2_A CH9	eTPU2_A CH14	FlexCAN_1
26	eTPU2_A CH16	eTPU2_A CH20	eTPU2_A CH25	FlexCAN_3
27	eTPU2_A CH17	eTPU2_A CH21	eTPU2_A CH26	eMIOS_1 CH8
28	eTPU2_A CH18	eTPU2_A CH22	eTPU2_A CH27	eMIOS_1 CH9
29	eTPU2_A CH19	eTPU2_A CH23	eTPU2_A CH28	eMIOS_1 CH10
30	eTPU2_A CH30	eTPU2_A CH24	eTPU2_A CH29	eMIOS_1 CH11
31	eTPU2_A CH31	eTPU2_B CH3	eTPU2_B CH8	eMIOS_1 CH12
32	eTPU2_B CH0	eTPU2_B CH4	eTPU2_B CH9	eMIOS_1 CH13
33	eTPU2_B CH1	eTPU2_B CH5	eTPU2_B CH10	eMIOS_1 CH14
34	eTPU2_B CH2	eTPU2_B CH6	eTPU2_B CH11	eMIOS_1 CH15
35	eTPU2_B CH13	eTPU2_B CH7	eTPU2_B CH12	eMIOS_1 CH16
36	eTPU2_B CH14	eTPU2_B CH18	eTPU2_B CH23	eMIOS_1 CH17
37	eTPU2_B CH15	eTPU2_B CH19	eTPU2_B CH24	eMIOS_1 CH18
38	eTPU2_B CH16	eTPU2_B CH20	eTPU2_B CH25	eMIOS_1 CH19
39	eTPU2_B CH17	eTPU2_B CH21	eTPU2_B CH26	eMIOS_1 CH20
40	eTPU2_B CH28	eTPU2_B CH22	eTPU2_B CH27	eMIOS_1 CH21
41	eTPU2_B CH29	BCTU_0	SAR_ADC_0	eMIOS_1 CH22
42	eTPU2_B CH30	ADC_SD_0 EOC	BCTU_1	eMIOS_1 CH23
43	eTPU2_B CH31	LINFlex_0 RX	DSPI_0 CMD	Reserved

Table continues on the next page...

Table 19-2. DMA request mapping (continued)

Source	DMAMUX_0	DMAMUX_1	DMAMUX_2	DMAMUX_3
44	SAR_ADC_2	LINFlex_0 TX	DSPI_0 RX	Reserved
45	DSPI_2 CMD	LINFlex_2 RX	DSPI_0 TX	Reserved
46	DSPI_2 RX	LINFlex_2 TX	SIUL2 REQ0	Reserved
47	DSPI_2 TX	SENT_0 RX FAST	eTPU2_A CH0	Reserved
48	SIUL2 REQ4	SENT_0 RX SLOW	eTPU2_A CH1	Reserved
49	FlexCAN_0	eTPU2_B CH0	eTPU2_A CH2	Reserved
50	Reaction_Module RCH0	eTPU2_B CH1	Reaction_Module RCH2	Reserved
51	Reaction_Module RCH3	eTPU2_B CH2	Reaction_Module RCH5	Reserved
52	Reaction_Module RCH6	Reaction_Module RCH1	Reaction_Module RCH8	Reserved
53	Reaction_Module RCH7	Reaction_Module RCH4	Reaction_Module RCH9	Reserved
54	Reserved	Reserved	Reserved	Reserved
55	Reserved	Reserved	Reserved	Reserved
56	Reserved	Reserved	Reserved	Reserved
57	Reserved	Reserved	Reserved	Reserved
58	Reserved	Reserved	Reserved	Reserved
59	Reserved	Reserved	Reserved	Reserved
60	Reserved	Reserved	Reserved	Reserved
61	Reserved	Reserved	Reserved	Reserved
62	Always on	Reserved	Reserved	Reserved
63	Always on	Always on	Always on	Reserved

NOTE

Multiple CHCONFIG registers, whether within the same DMAMUX instance or among multiple DMAMUX instances, must not select the same source value.

The mapping of the PIT periodic triggers to the DMA channel sources is shown in the following table.

Table 19-3. PIT_0 periodic trigger to DMA channel mapping

PIT_0 channel	PIT_0 periodic trigger	DMA Channel
0	DMAMUX_0 Trigger 0	0
1	DMAMUX_0 Trigger 1	1
2	DMAMUX_1 Trigger 0	16
3	DMAMUX_2 Trigger 0	32

The DMAMUX_CHCFGn[TRIG] field is only available in the following DMAMUX instances.

Table 19-4. DMAMUX instance with TRIG field

DMAMUX instance	DMAMUX_CHCFGn register with TRIG field
DMAMUX_0	DMAMUX_CHCFG0 and DMAMUX_CHCFG1
DMAMUX_1	DMAMUX_CHCFG0
DMAMUX_2	DMAMUX_CHCFG0

19.1.4 DMAMUX Memory map

Each of the four instances of DMAMUX (0–3) is allocated 512 bytes of memory space. The DMAMUX address offsets are shown in the following table. Offsets are from the DMAMUX base address.

Table 19-5. DMAMUX memory map

Offset	Register	Access
0x0000	DMAMUX_0 CHCONFIG0-CHCONFIG15	R/W
0x0200	DMAMUX_1 CHCONFIG0-CHCONFIG15	R/W
0x0400	DMAMUX_2 CHCONFIG0-CHCONFIG15	R/W
0x0600	DMAMUX_3 CHCONFIG0-CHCONFIG15	R/W

Each channel must be configured separately via settings in a dedicated configuration register. There is one configuration register per channel for each DMAMUX.

19.2 Introduction

19.2.1 Overview

The Direct Memory Access Multiplexer (DMAMUX) routes DMA sources, called slots, to any of the 16 DMA channels. This process is illustrated in the following figure.

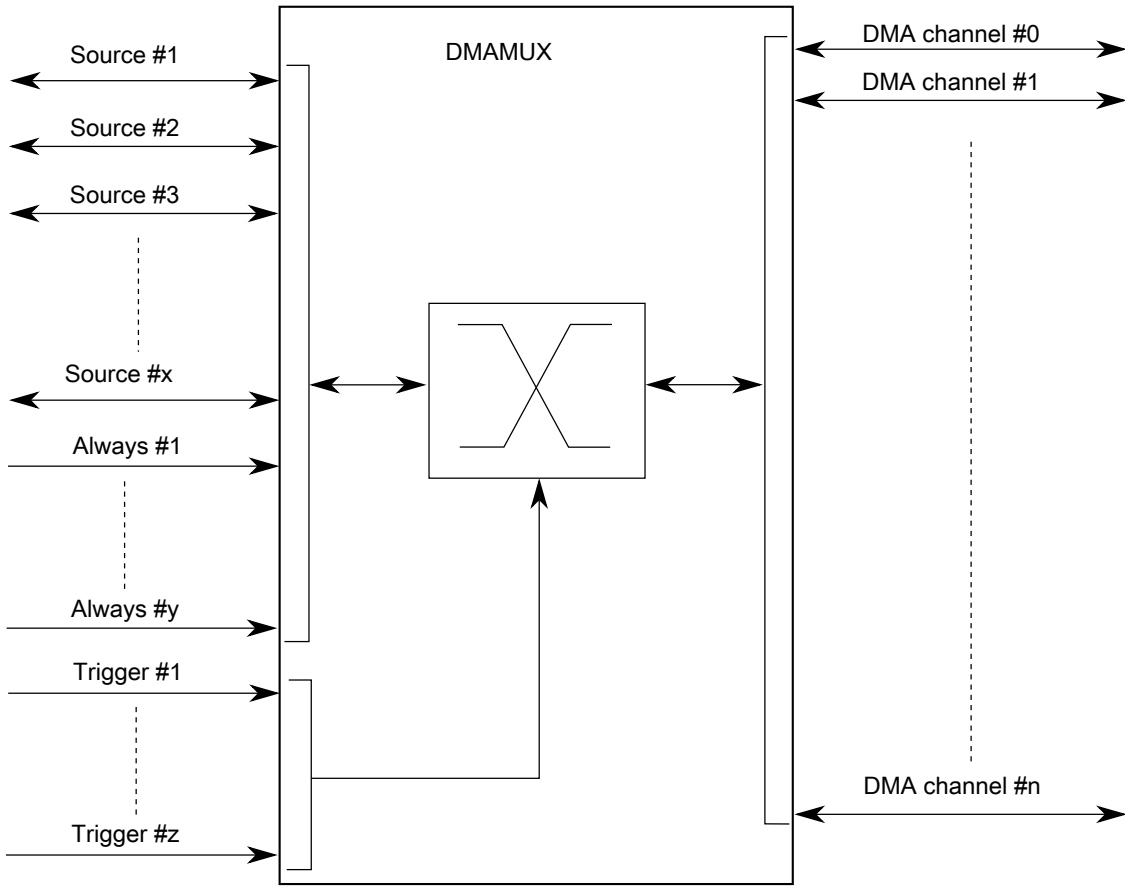


Figure 19-2. DMAMUX block diagram

19.2.2 Features

The DMAMUX module provides these features:

- Up to 61 peripheral slots and up to two always-on slots can be routed to 16 channels.
- 16 independently selectable DMA channel routers.
  - The first four channels additionally provide a trigger functionality.
- Each channel router can be assigned to one of the possible peripheral DMA slots or to one of the always-on slots.

19.2.3 Modes of operation

The following operating modes are available:

- Disabled mode

In this mode, the DMA channel is disabled. Because disabling and enabling of DMA channels is done primarily via the DMA configuration registers, this mode is used mainly as the reset state for a DMA channel in the DMA channel MUX. It may also be used to temporarily suspend a DMA channel while reconfiguration of the system takes place, for example, changing the period of a DMA trigger.

- Normal mode

  In this mode, a DMA source is routed directly to the specified DMA channel. The operation of the DMAMUX in this mode is completely transparent to the system.

- Periodic Trigger mode

  In this mode, a DMA source may only request a DMA transfer, such as when a transmit buffer becomes empty or a receive buffer becomes full, periodically.

  Configuration of the period is done in the registers of the periodic interrupt timer (PIT). This mode is available only for channels 0–3.

19.3 External signal description

The DMAMUX has no external pins.

19.4 Memory map/register definition

This section provides a detailed description of all memory-mapped registers in the DMAMUX.

DMAMUX memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Channel Configuration register (DMAMUX_CHCFG0)	8	R/W	00h	19.4.1/614
1	Channel Configuration register (DMAMUX_CHCFG1)	8	R/W	00h	19.4.1/614
2	Channel Configuration register (DMAMUX_CHCFG2)	8	R/W	00h	19.4.1/614
3	Channel Configuration register (DMAMUX_CHCFG3)	8	R/W	00h	19.4.1/614
4	Channel Configuration register (DMAMUX_CHCFG4)	8	R/W	00h	19.4.1/614
5	Channel Configuration register (DMAMUX_CHCFG5)	8	R/W	00h	19.4.1/614
6	Channel Configuration register (DMAMUX_CHCFG6)	8	R/W	00h	19.4.1/614
7	Channel Configuration register (DMAMUX_CHCFG7)	8	R/W	00h	19.4.1/614
8	Channel Configuration register (DMAMUX_CHCFG8)	8	R/W	00h	19.4.1/614

Table continues on the next page...

DMAMUX memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9	Channel Configuration register (DMAMUX_CHCFG9)	8	R/W	00h	19.4.1/614
A	Channel Configuration register (DMAMUX_CHCFG10)	8	R/W	00h	19.4.1/614
B	Channel Configuration register (DMAMUX_CHCFG11)	8	R/W	00h	19.4.1/614
C	Channel Configuration register (DMAMUX_CHCFG12)	8	R/W	00h	19.4.1/614
D	Channel Configuration register (DMAMUX_CHCFG13)	8	R/W	00h	19.4.1/614
E	Channel Configuration register (DMAMUX_CHCFG14)	8	R/W	00h	19.4.1/614
F	Channel Configuration register (DMAMUX_CHCFG15)	8	R/W	00h	19.4.1/614

19.4.1 Channel Configuration register (DMAMUX_CHCFG*n*)

Each of the DMA channels can be independently enabled/disabled and associated with one of the DMA slots (peripheral slots or always-on slots) in the system.

NOTE

Setting multiple CHCFG registers with the same source value will result in unpredictable behavior. This is true, even if a channel is disabled (ENBL==0).

Before changing the trigger or source settings, a DMA channel must be disabled via CHCFGn[ENBL].

Address: 0h base + 0h offset + (1d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7
Read Write	ENBL	TRIG	SOURCE					
Reset	0	0	0	0	0	0	0	0

DMAMUX_CHCFG*n* field descriptions

Field	Description
0 ENBL	DMA Channel Enable Enables the DMA channel. 0 DMA channel is disabled. This mode is primarily used during configuration of the DMAMux. The DMA has separate channel enables/disables, which should be used to disable or reconfigure a DMA channel. 1 DMA channel is enabled
1 TRIG	DMA Channel Trigger Enable Enables the periodic trigger capability for the triggered DMA channel.

Table continues on the next page...

DMAMUX_CHCFG*n* field descriptions (continued)

Field	Description
	0 Triggering is disabled. If triggering is disabled and ENBL is set, the DMA Channel will simply route the specified source to the DMA channel. (Normal mode) 1 Triggering is enabled. If triggering is enabled and ENBL is set, the DMAMUX is in Periodic Trigger mode.
2–7 SOURCE	DMA Channel Source (Slot) Specifies which DMA source, if any, is routed to a particular DMA channel. See the chip-specific DMAMUX information for details about the peripherals and their slot numbers.

19.5 Functional description

The primary purpose of the DMAMUX is to provide flexibility in the system's use of the available DMA channels.

As such, configuration of the DMAMUX is intended to be a static procedure done during execution of the system boot code. However, if the procedure outlined in Enabling and configuring sources is followed, the configuration of the DMAMUX may be changed during the normal operation of the system.

Functionally, the DMAMUX channels may be divided into two classes:

- Channels that implement the normal routing functionality plus periodic triggering capability
- Channels that implement only the normal routing functionality

19.5.1 DMA channels with periodic triggering capability

Besides the normal routing functionality, the first 4 channels of the DMAMUX provide a special periodic triggering capability that can be used to provide an automatic mechanism to transmit bytes, frames, or packets at fixed intervals without the need for processor intervention.

The trigger is generated by the periodic interrupt timer (PIT); as such, the configuration of the periodic triggering interval is done via configuration registers in the PIT. See the section on periodic interrupt timer for more information on this topic.

Note

Because of the dynamic nature of the system (due to DMA channel priorities, bus arbitration, interrupt service routine lengths, etc.), the number of clock cycles between a trigger and the actual DMA transfer cannot be guaranteed.

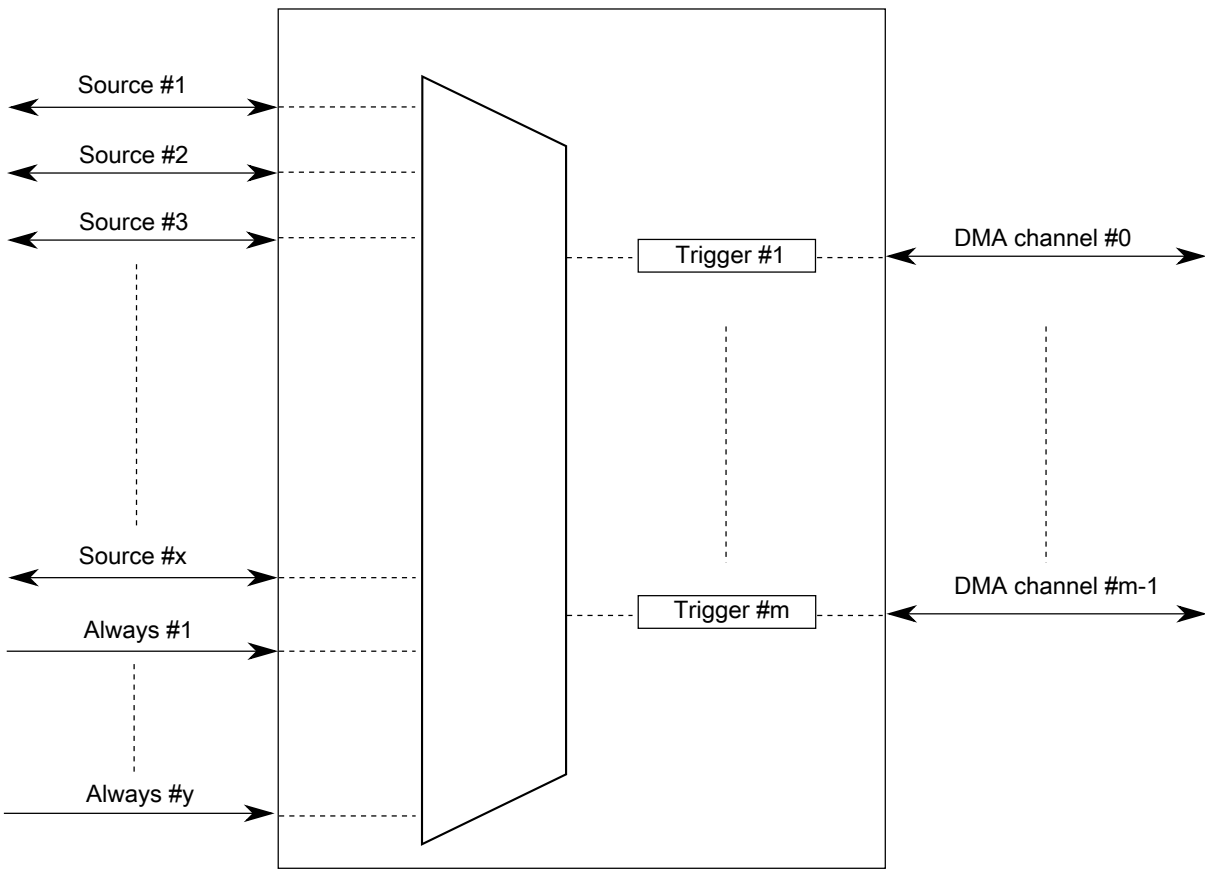


Figure 19-20. DMAMUX triggered channels

The DMA channel triggering capability allows the system to schedule regular DMA transfers, usually on the transmit side of certain peripherals, without the intervention of the processor. This trigger works by gating the request from the peripheral to the DMA until a trigger event has been seen. This is illustrated in the following figure.

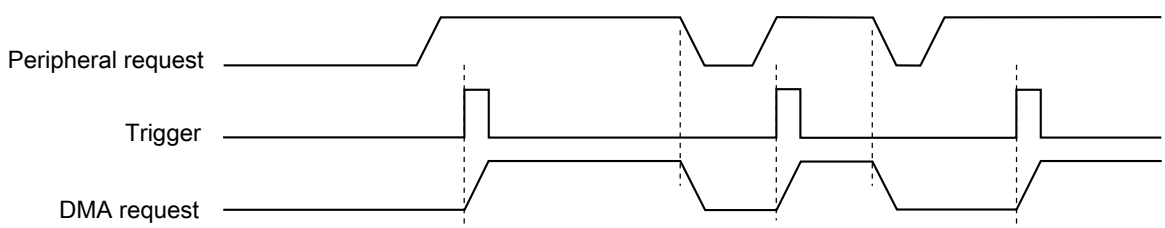


Figure 19-21. DMAMUX channel triggering: normal operation

After the DMA request has been serviced, the peripheral will negate its request, effectively resetting the gating mechanism until the peripheral reasserts its request and the next trigger event is seen. This means that if a trigger is seen, but the peripheral is not requesting a transfer, then that trigger will be ignored. This situation is illustrated in the following figure.

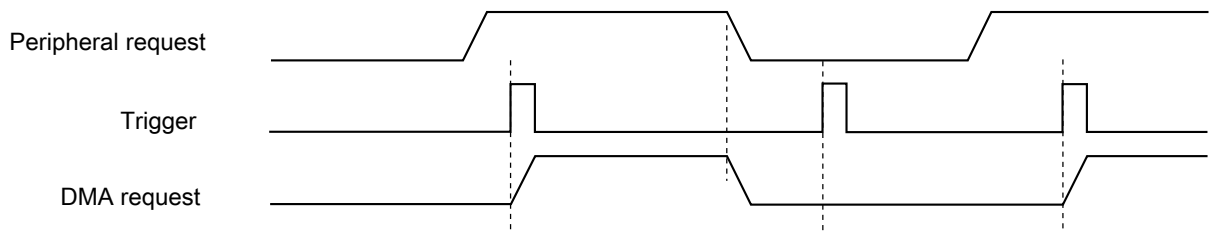


Figure 19-22. DMAMUX channel triggering: ignored trigger

This triggering capability may be used with any peripheral that supports DMA transfers, and is most useful for two types of situations:

- Periodically polling external devices on a particular bus

  As an example, the transmit side of an SPI is assigned to a DMA channel with a trigger, as described above. After it has been set up, the SPI will request DMA transfers, presumably from memory, as long as its transmit buffer is empty. By using a trigger on this channel, the SPI transfers can be automatically performed every 5 µs (as an example). On the receive side of the SPI, the SPI and DMA can be configured to transfer receive data into memory, effectively implementing a method to periodically read data from external devices and transfer the results into memory without processor intervention.

- Using the GPIO ports to drive or sample waveforms

  By configuring the DMA to transfer data to one or more GPIO ports, it is possible to create complex waveforms using tabular data stored in on-chip memory. Conversely, using the DMA to periodically transfer data from one or more GPIO ports, it is possible to sample complex waveforms and store the results in tabular form in on-chip memory.

A more detailed description of the capability of each trigger, including resolution, range of values, and so on, may be found in the periodic interrupt timer section.

19.5.2 DMA channels with no triggering capability

The other channels of the DMAMUX provide the normal routing functionality as described in Modes of operation.

19.5.3 Always-enabled DMA sources

In addition to the peripherals that can be used as DMA sources, there are two additional DMA sources that are always enabled. Unlike the peripheral DMA sources, where the peripheral controls the flow of data during DMA transfers, the sources that are always enabled provide no such "throttling" of the data transfers. These sources are most useful in the following cases:

- Performing DMA transfers to/from GPIO—Moving data from/to one or more GPIO pins, either unthrottled (that is, as fast as possible), or periodically (using the DMA triggering capability).
- Performing DMA transfers from memory to memory—Moving data from memory to memory, typically as fast as possible, sometimes with software activation.
- Performing DMA transfers from memory to the external bus, or vice-versa—Similar to memory to memory transfers, this is typically done as quickly as possible.
- Any DMA transfer that requires software activation—Any DMA transfer that should be explicitly started by software.

In cases where software should initiate the start of a DMA transfer, an always-enabled DMA source can be used to provide maximum flexibility. When activating a DMA channel via software, subsequent executions of the minor loop require that a new start event be sent. This can either be a new software activation, or a transfer request from the DMA channel MUX. The options for doing this are:

- Transfer all data in a single minor loop.

  By configuring the DMA to transfer all of the data in a single minor loop (that is, major loop counter = 1), no reactivation of the channel is necessary. The disadvantage to this option is the reduced granularity in determining the load that the DMA transfer will impose on the system. For this option, the DMA channel must be disabled in the DMA channel MUX.

- Use explicit software reactivation.

  In this option, the DMA is configured to transfer the data using both minor and major loops, but the processor is required to reactivate the channel by writing to the DMA registers *after every minor loop*. For this option, the DMA channel must be disabled in the DMA channel MUX.

- Use an always-enabled DMA source.

In this option, the DMA is configured to transfer the data using both minor and major loops, and the DMA channel MUX does the channel reactivation. For this option, the DMA channel should be enabled and pointing to an "always enabled" source. Note that the reactivation of the channel can be continuous (DMA triggering is disabled) or can use the DMA triggering capability. In this manner, it is possible to execute periodic transfers of packets of data from one source to another, without processor intervention.

19.6 Initialization/application information

This section provides instructions for initializing the DMA channel MUX.

19.6.1 Reset

The reset state of each individual bit is shown in Memory map/register definition. In summary, after reset, all channels are disabled and must be explicitly enabled before use.

19.6.2 Enabling and configuring sources

To enable a source with periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 4 DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Configure the corresponding timer.
5. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] and CHCFG[TRIG] fields are set.

NOTE

The following is an example. See the chip configuration details for the number of this device's DMA channels that have triggering capability.

To configure source #5 transmit for use with DMA channel 1, with periodic triggering capability:

1. Write 0x00 to CHCFG1.

2. Configure channel 1 in the DMA, including enabling the channel.
3. Configure a timer for the desired trigger interval.
4. Write 0xC5 to CHCFG1.

The following code example illustrates steps 1 and 4 above:

```
void DMAMUX_Init(uint8_t DMA_CH, uint8_t DMAMUX_SOURCE)
{
    DMAMUX_0.CHCFG[DMA_CH].B.SOURCE = DMAMUX_SOURCE;
    DMAMUX_0.CHCFG[DMA_CH].B.ENBL   = 1;
    DMAMUX_0.CHCFG[DMA_CH].B.TRIG   = 1;
}
```

To enable a source, without periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 4 DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that CHCFG[ENBL] is set while CHCFG[TRIG] is cleared.

NOTE

The following is an example. See the chip configuration details for the number of this device's DMA channels that have triggering capability.

To configure source #5 transmit for use with DMA channel 1, with no periodic triggering capability:

1. Write 0x00 to CHCFG1.
2. Configure channel 1 in the DMA, including enabling the channel.
3. Write 0x85 to CHCFG1.

The following code example illustrates steps 1 and 3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
```

```
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);

In File main.c:
#include "registers.h"
:
:
*CHCFG1 = 0x00;
*CHCFG1 = 0x85;
```

To disable a source:

A particular DMA source may be disabled by not writing the corresponding source value into any of the CHCFG registers. Additionally, some module-specific configuration may be necessary. See the appropriate section for more details.

To switch the source of a DMA channel:

1. Disable the DMA channel in the DMA and reconfigure the channel for the new source.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] bits of the DMA channel.
3. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] and CHCFG[TRIG] fields are set.

To switch DMA channel 8 from source #5 transmit to source #7 transmit:

1. In the DMA configuration registers, disable DMA channel 8 and reconfigure it to handle the transfers to peripheral slot 7. This example assumes channel 8 doesn't have triggering capability.
2. Write 0x00 to CHCFG8.
3. Write 0x87 to CHCFG8. (In this example, setting CHCFG[TRIG] would have no effect due to the assumption that channel 8 does not support the periodic triggering functionality.)

The following code example illustrates steps 2 and 3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);
```

```
In File main.c:
#include "registers.h"
:
:
*CHCFG8 = 0x00;
*CHCFG8 = 0x87;
```

Chapter 20
Wakeup Unit (WKPU)

20.1 Introduction

The Wakeup Unit (WKPU) supports one external source that can cause non-maskable interrupts to on-chip cores and wakeup events to the system. The following figure is a block diagram of the WKPU and its interfaces to other system components.

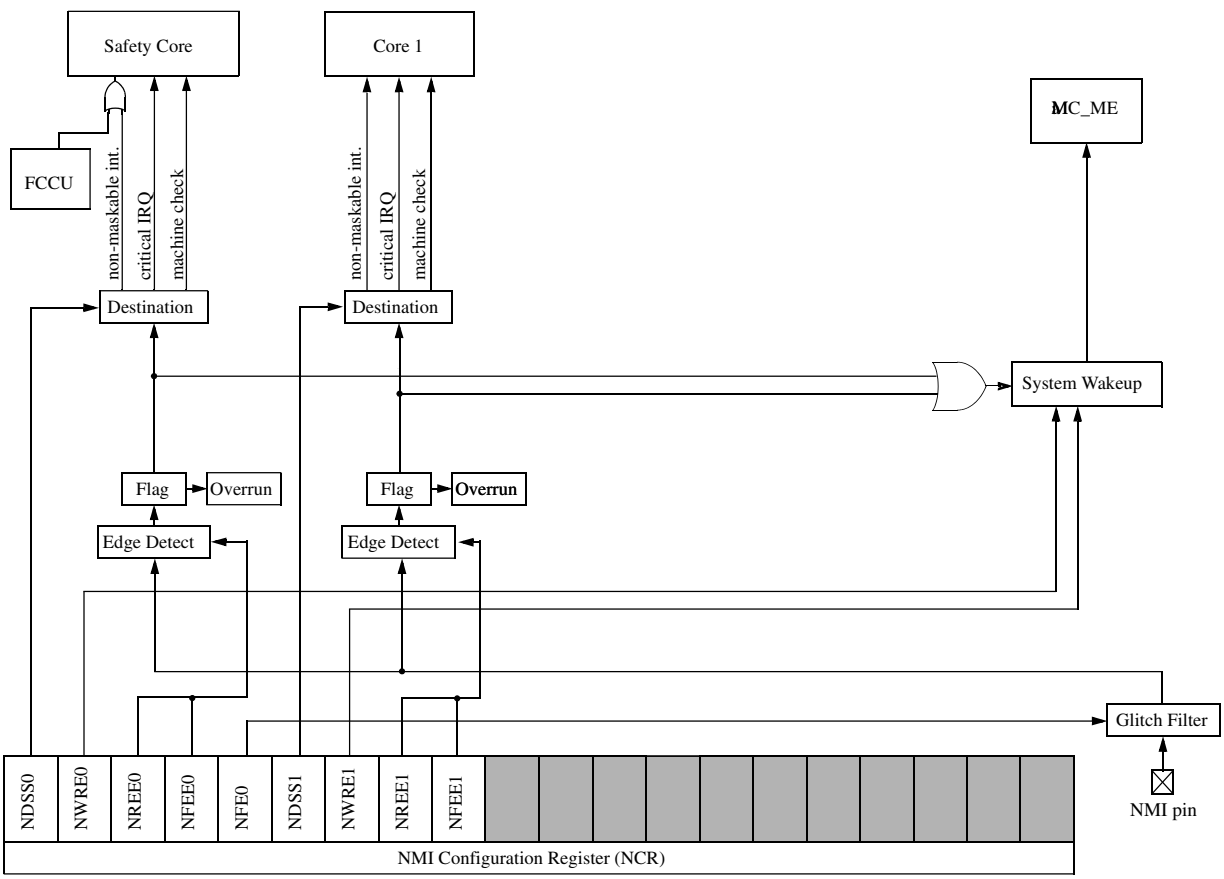


Figure 20-1. WKPU block diagram

20.1.1 Features

The WKPU supports these features:

- One external NMI source
- One analog glitch filter
- Independent interrupt destination for each core (as shown in figure above):
  - Non-maskable interrupt
  - Critical interrupt
  - Machine check request
- Active edge selection control for events to each core
- Configurable wakeup event to exit from low-power mode to the MC_ME
- Configurable wakeup event to exit from Stop mode to the MC_ME
- WKPU register configuration lock

20.2 External Signal Description

There is a single, active-edge selectable, NMI pin on the device that connects to the WKPU block. See this chip's signal description information for details on this pin.

20.3 WKPU memory map and registers

This section provides a detailed description of all registers accessible in the WKPU module.

NOTE

Reserved registers will read as 0, writes will have no effect. A transfer error will be issued when trying to access completely reserved register space.

WKPU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	NMI Status Flag Register (WKPU_NSR)	32	R/W	0000_0000h	20.3.1/625
8	NMI Configuration Register (WKPU_NCR)	32	R/W	0000_0000h	20.3.2/627

20.3.1 NMI Status Flag Register (WKPU_NSR)

This register holds the status flags of the NMI external source pin. This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register. The status flag is set whenever an NMI event is detected. The overrun flag is set whenever an NMI event is detected and the status flag is set (i.e., has not yet been cleared).

The status flags are set independently of the NDSS bits. Therefore, these flags can set but not generate a non-maskable interrupt or system wakeup if NWRE = 0 or if NDSS is set to “no request”.

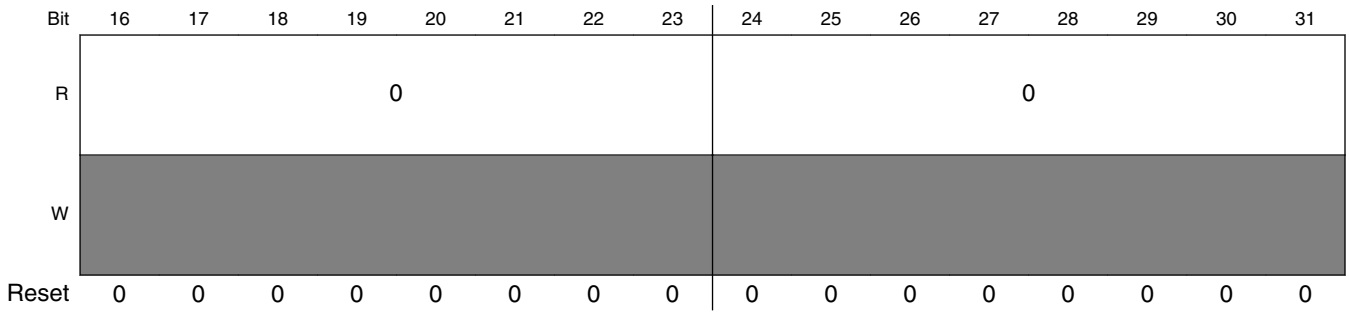
NOTE

The overrun flag is cleared by writing a ‘1’ to the appropriate overrun bit in the NSR. If the status bit is cleared and the overrun bit is still set, the pending interrupt will not be cleared.

Also see the notes in NMI Configuration Register (WKPU_NCR).

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NIF0	NOVF0	0						NIF1	NOVF1	0					
W	w1c	w1c							w1c	w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



WKPU_NSR field descriptions

Field	Description
0 NIF0	NMI Status Flag 0 (safety core) If enabled (i.e., NCR[NREE0] or NCR[NFEE0] is set), an NMI input event will set NIF0. NIF0 causes an interrupt request based on the NMI destination, as configured by NCR[NDSS0], and/or a system wakeup request based on NCR[NWRE0]. 0 No event has occurred on the pad 1 An event as defined by NCR[NREE0] and NCR[NFEE0] has occurred
1 NOVF0	NMI Overrun Status Flag 0 (safety core) This field is a copy of the current NIF0 value whenever a NMI event occurs, thereby indicating to the software that an NMI occurred while the last one was not yet serviced. NOVF0 causes an interrupt request based on the NMI destination, as configured by NCR[NDSS0], and/or a system wakeup request based on the NCR[NWRE0] bit configuration. 0 No overrun has occurred on NMI input 0 1 An overrun has occurred on NMI input 0
2–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 NIF1	NMI Status Flag 1 (Core 1) If enabled (i.e., NCR[NREE1] or NCR[NFEE1] is set), an NMI input event will set NIF1. NIF1 causes an interrupt request based on the NMI destination, as configured by NCR[NDSS1], and/or a system wakeup request based on NCR[NWRE1]. 0 No event has occurred on the pad 1 An event as defined by NCR[NREE1] and NCR[NFEE1] has occurred
9 NOVF1	NMI Overrun Status Flag 1 (Core 1) This field is a copy of the current NIF1 value whenever a NMI event occurs, thereby indicating to the software that an NMI occurred while the last one was not yet serviced. NOVF1 causes an interrupt request based on the NMI destination, as configured by the NCR[NDSS1] bits, and/or a system wakeup request based on the NCR[NWRE1] bit configuration. 0 No overrun has occurred on NMI input 1 1 An overrun has occurred on NMI input 1
10–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

WKPU_NSR field descriptions (continued)

Field	Description
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

20.3.2 NMI Configuration Register (WKPU_NCR)

This register holds the configuration bits for the non-maskable interrupt settings.

NOTE

Writing a '0' to both NREE*n* and NFEE*n* disables the NMI functionality completely (i.e. no system wakeup or interrupt will be generated on any pad activity).

NOTE

Writing to the NCR when NIF = 1 or NOVF = 1 causes an interrupt to the destination source/system wakeup to be generated independent of an NMI wakeup event.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NLOCK0	NDSS0		NWRE0	0	NREE0	NFEE0	NFE0	NLOCK1	NDSS1		NWRE1	0	NREE1	NFEE1	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

WKPU_NCR field descriptions

Field	Description
0 NLOCK0	NMI Configuration Lock Register 0 0 Writing 0 to this bit has no effect. 1 Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset.
1–2 NDSS0	NMI Destination Source Select 0 **NOTE:** If this field is configured for machine check exception, the following steps must be performed in order to receive the exception on the NMI Pad:

Table continues on the next page...

WKPU_NCR field descriptions (continued)

Field	Description
	1. Set MSR[ME], Machine Check Enable field of the core's Machine State Register, to 1. 2. Set HID0[EMCP] to 1. 00 A non-maskable interrupt will occur if NSR[NIF0] or NSR[NOVF0] is set. 01 A critical interrupt will occur if NSR[NIF0] or NSR[NOVF0] is set. 10 A machine check request will occur if NSR[NIF0] or NSR[NOVF0] is set. 11 No reaction
3 NWRE0	NMI Wakeup Request Enable 0 **NOTE:** If wakeup requests are disabled, the corresponding NDSS field must be set to 11 to disable wakeups from an NMI, critical interrupt, or machine check. 0 System wakeup requests from the corresponding NSR[NIF0] bit are disabled 1 A set NSR[NIF0] bit or set NSR[NOVF0] bit causes a system wakeup request
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 NREE0	NMI Rising-edge Events Enable 0 0 Rising-edge event is disabled 1 Rising-edge event is enabled
6 NFEE0	NMI Falling-edge Events Enable 0 0 Falling-edge event is disabled 1 Falling-edge event is enabled
7 NFE0	NMI Filter Enable Enable analog glitch filter on the NMI pad input. 0 Filter is disabled 1 Filter is enabled
8 NLOCK1	NMI Configuration Lock Register 1 0 Writing 0 to this bit has no effect. 1 Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset.
9–10 NDSS1	NMI Destination Source Select 1 00 A non-maskable interrupt will occur if NSR[NIF1] or NSR[NOVF1] is set. 01 A critical interrupt will occur if NSR[NIF1] or NSR[NOVF1] is set. 10 A machine check request will occur if NSR[NIF1] or NSR[NOVF1] is set. 11 No reaction
11 NWRE1	NMI Wakeup Request Enable 1 0 System wakeup requests from the corresponding NSR[NIF1] bit are disabled 1 A set NSR[NIF1] bit or set NSR[NOVF1] bit causes a system wakeup request
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 NREE1	NMI Rising-edge Events Enable 1

Table continues on the next page...

WKPU_NCR field descriptions (continued)

Field	Description
	0 Rising-edge event is disabled 1 Rising-edge event is enabled
14 NFEE1	NMI Falling-edge Events Enable 1 0 Falling-edge event is disabled 1 Falling-edge event is enabled
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

20.4 Functional Description

This section provides a functional description of the WKPU features.

20.4.1 Non-Maskable Interrupts

The WKPU allows the external NMI pin to assert the non-maskable interrupts of the cores on the device. These interrupts include non-maskable interrupts, critical interrupts, or machine check requests to any of the cores including the main core and safety core. The safety core's non-maskable interrupt line from the WKPU is routed through the FCCU block, where is it combined with the FCCU internal NMI interrupt event arising due to fault management logic.

The NMI destination interrupt is controlled by the user through the configuration of NCR[NDSS*n*].

The enabling of rising-edge, falling-edge, or either-edge reaction to the NMI pin is independent for each core, and is controlled by NCR[NFEE*n*] and NCR[NREE*n*].

The WKPU supports the capturing of a second event per NMI input before the interrupt is cleared, thus reducing the chance of losing an NMI event.

The NMI supports a status flag, NSR[NIF*n*], and an overrun flag, NSR[NOVF*n*].

20.4.2 System Wakeup

The WKPU allows the external NMI pin to assert a system wakeup from STOP to the MC_ME module on the chip. The system wakeup is enabled through the NWRE bits in the NCR register.

20.4.3 Glitch Filter

An analog glitch filter, between the NMI pin and the WKPU module, is enabled by NCR[NFE0]. See the chip's data sheet for the glitch filter's electrical specifications.

20.4.3.1 Edge Detect

Each NMI request can be enabled or disabled independently. This can be performed using the NCR register, which contains all configuration bits for a given NMI in a single byte. The user can configure interrupts with an active rising edge, an active falling edge, or either edge being active. Disabling both edge events results in no interrupt being propagated from the WKPU block.

The active NMI edge is controlled by the user through the configuration of the NREE and NFEE bits.

NOTE

After reset, all rising-/falling-edge enable bits set to 0, therefore the NMI functionality is disabled after reset and must be enabled explicitly by software.

20.5 Initialization Information

This section discusses initialization for the following features:

- Glitch Filter and Pad Configuration
- Non-Maskable Interrupts

20.5.1 Glitch Filter and Pad Configuration

Glitch filter control and pad configuration should be performed while the NMI is disabled in order to avoid erroneous triggering by glitches caused by the configuration process itself.

When enabling the glitch filter, do not enable the rising-/falling-edge events bits, i.e., the NREEand NFEE bits in the same register write.

20.5.2 Non-Maskable Interrupts

When an NMI interrupt pin is first enabled, it is possible to get a false interrupt flag. To prevent a false interrupt request during pin interrupt initialization the user must do the following:

1. Mask interrupts by clearing NSR[NIF*n*] and NSR[NOVF*n*].
2. Clear NCR[NFEE*n*] and NCR[NREE*n*], preferably by writing 0 to all bits in the register.
3. Configure the analog glitch filter using NCR[NFE0].
4. Write appropriate bit values to the NCR register to configure:
    - NMI source (NDSS*n*).
    - Wakeup request (NWRE*n*).
    - Glitch filter (NFE0)
    - Rising/falling edge events enable (NFEE*n* and NREE*n*).
    - NMI configuration lock (NLOCK*n*).

Chapter 21
Error Injection Module (EIM)

21.1 Error Injection Module (EIM) chip-specific module information

The EIM provides error injection for the DMA memory Error Correcting Code (ECC).

21.2 Introduction

The Error Injection module is mainly used for diagnostic purposes. It provides a method for diagnostic coverage of the peripheral memories. See the chip-specific EIM information to determine which peripheral memories are supported by this method.

21.2.1 Overview

The Error Injection Module (EIM) provides support for inducing single-bit and multi-bit inversions on read data when accessing peripheral RAMs. Injecting faults on memory accesses can be used to exercise the SEC-DED ECC function of the related system.

NOTE

The following diagram shows an example EIM implementation with a 64-bit read data bus and an 8-bit checkbit bus.

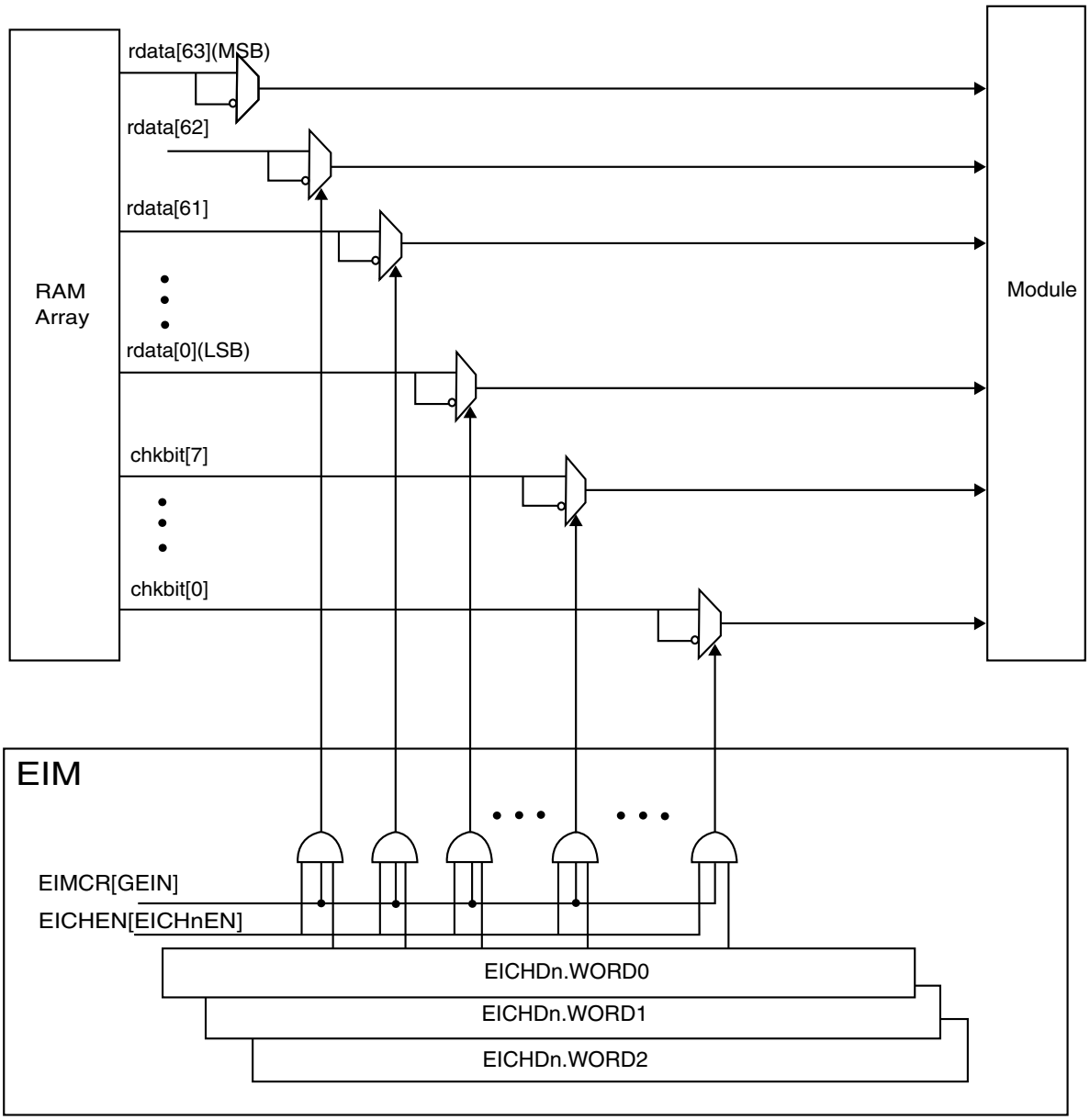


Figure 21-1. EIM functional block diagram (64-bit read data bus and 8-bit checkbit bus)

21.2.2 Features

The EIM includes these features:

- Supports 1 error injection channel
- Protection against accidental enable and reconfiguration error injection function via two-stage enable mechanism

21.3 Memory map and register definition

The EIM provides an IPS programming model mapped to an on-platform peripheral slot.

Programming model access

All system bus masters can access the programming model:

- Only in supervisor mode
- Using only 32-bit (word) accesses

Any of the following attempted references to the programming model generates an IPS error termination:

- In user mode
- Using non-32-bit access sizes
- To undefined (reserved) addresses

Attempted updates to the programming model while the EIM is in the midst of an operation will result in non-deterministic behavior.

Error injection channel descriptor: function and structure

Each error injection channel descriptor:

- Specifies a mask that defines which bits of the read data and checkbit bus from target RAM are inverted on a read access.
- Consists of a 128-bit (16-byte) structure, composed of four 32-bit words, in the EIM programming model.
    - Each descriptor consists of Error Injection Channel Descriptor Word0-1 (EICHD*n*_WORD0-1) and, depending on the implementation, Word2 (EICHD*n*_WORD2).
    - When Word2 is unused, the corresponding word in the 16-byte structure is unused.
    - The fourth word in the 16-byte structure is always unused.

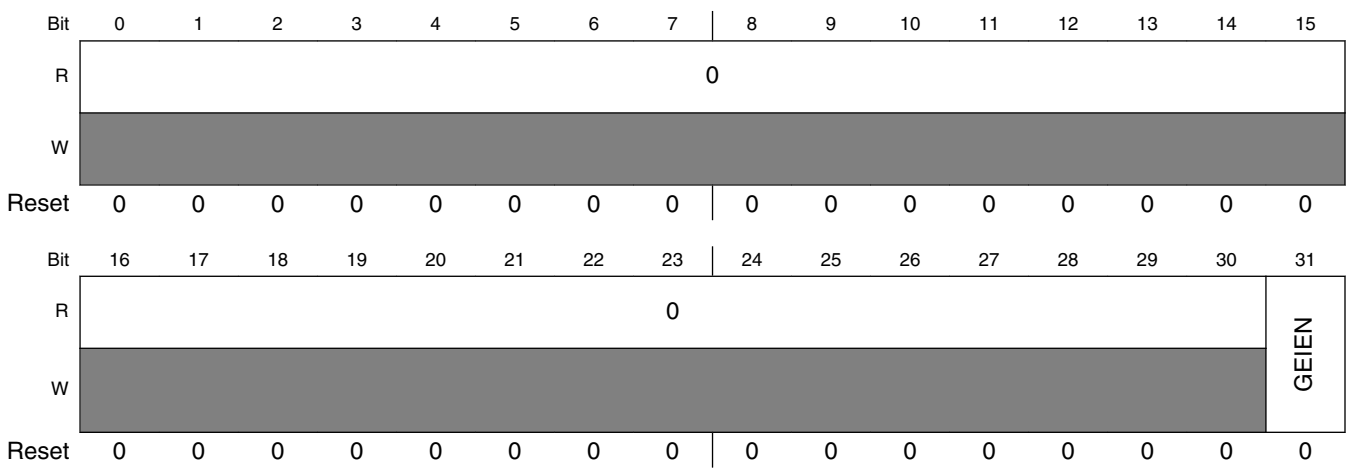
EIM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Error Injection Module Configuration Register (EIM_EIMCR)	32	R/W	0000_0000h	21.3.1/636
4	Error Injection Channel Enable register (EIM_EICHEN)	32	R/W	0000_0000h	21.3.2/637
100	Error Injection Channel Descriptor, Word0 (EIM_EICHD0_WORD0)	32	R/W	0000_0000h	21.3.3/638
104	Error Injection Channel Descriptor, Word1 (EIM_EICHD0_WORD1)	32	R/W	0000_0000h	21.3.4/639
108	Error Injection Channel Descriptor, Word2 (EIM_EICHD0_WORD2)	32	R/W	0000_0000h	21.3.5/639

21.3.1 Error Injection Module Configuration Register (EIM_EIMCR)

The EIM Configuration Register is used to globally enable/disable the error injection function.

Address: 0h base + 0h offset = 0h



EIM_EIMCR field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 GEIEN	Global Error Injection Enable This bit globally enables or disables the error injection function of the EIM. This field is initialized by hardware reset. 0 Disabled 1 Enabled

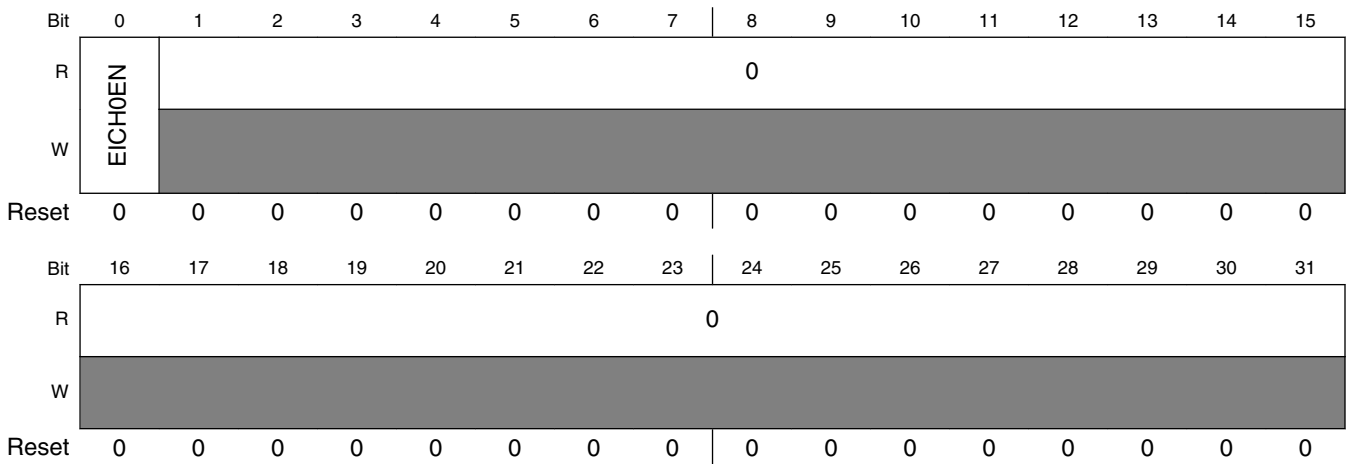
21.3.2 Error Injection Channel Enable register (EIM_EICHEN)

Each field of the Error Injection Channel Enable register (EICHEN) is used to enable or disable the corresponding error injection channel.

NOTE

To enable an error injection channel, the Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted.

Address: 0h base + 4h offset = 4h



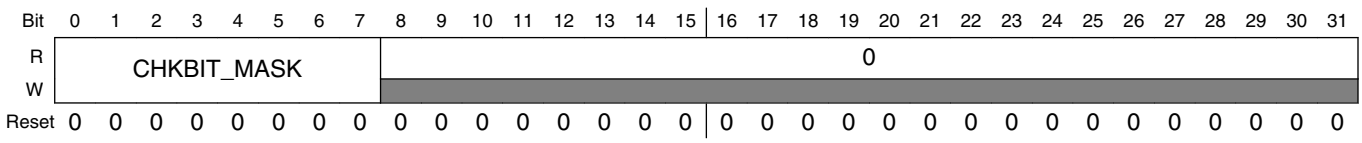
EIM_EICHEN field descriptions

Field	Description
0 EICH0EN	Error Injection Channel 0 Enable This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection. After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software. Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel. This field is cleared by power-on reset and unaffected by other types of system reset. 0 Error injection is disabled on Error Injection Channel 0 1 Error injection is enabled on Error Injection Channel 0
1–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

21.3.3 Error Injection Channel Descriptor, Word0 (EIM_EICHD*n*_WORD0)

The first word of the Error Injection Channel Descriptor defines a left-justified mask field (CHKBIT_MASK) whose width is up to 8 bits. Each bit of CHKBIT_MASK specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified on read accesses. Successful writes to this field clear the corresponding error injection channel valid bit, EICHEN[EICH*n*EN].

Address: 0h base + 100h offset + (256d × i), where i=0d to 0d



EIM_EICHD*n*_WORD0 field descriptions

Field	Description
0–7 CHKBIT_MASK	Checkbit Mask This field defines a bit-mapped mask that specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified. The width of the field for the channel is defined in the chip-specific EIM information. 0 The corresponding bit of the checkbit bus remains unmodified. 1 The corresponding bit of the checkbit bus is inverted.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

21.3.4 Error Injection Channel Descriptor, Word1 (EIM_EICHD*n*_WORD1)

The second word of the Error Injection Channel Descriptor defines the bits in the mask corresponding to bytes 0–3 of the read data bus. Each bit specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified on read accesses. Successful writes to this field clear the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

Address: 0h base + 104h offset + (256d × i), where i=0d to 0d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	B0_3DATA_MASK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EIM_EICHD*n*_WORD1 field descriptions

Field	Description
0–31 B0_3DATA_ MASK	Data Mask Bytes 0-3 This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified. For details about which bits in this field are implemented for the applicable channel and how the bits map to bytes 0-3 of the read data bus, see the chip-specific EIM information. 0 The corresponding bit of bytes 0-3 on the read data bus remains unmodified 1 The corresponding bit of bytes 0-3 on the read data bus is inverted.

21.3.5 Error Injection Channel Descriptor, Word2 (EIM_EICHD*n*_WORD2)

The third word of the Error Injection Channel Descriptor, when present, defines the bits in the mask corresponding to bytes 4–7 of the read data bus. Each bit specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified on read accesses. Successful writes to this field clear the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

Address: 0h base + 108h offset + (256d × i), where i=0d to 0d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	B4_7DATA_MASK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EIM_EICHD*n*_WORD2 field descriptions

Field	Description
0–31 B4_7DATA_ MASK	Data Mask Bytes 4-7 This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified. For details about which bits in this field are implemented for the applicable channel and how the bits map to bytes 4-7 of the read data bus, see the chip-specific EIM information. 0 The corresponding bit of bytes 4-7 on the read data bus remains unmodified. 1 The corresponding bit of bytes 4-7 on the read data bus is inverted.

21.4 Functional description

The EIM provides protection against accidental enabling and reconfiguration of the error injection function by enforcing a two-stage enablement mechanism. To properly enable the error injection mechanism for a channel, execute both of the following:

- Assert the EICHEN[EICH*n*EN] field, where *n* denotes the channel number.
- Assert EIMCR[GEIEN].

NOTE

When the use case for a channel requires writing any EICHD*n*_WORD register, write the EICHD*n*_WORD register before executing the two-stage enablement mechanism. A successful write to any EICHD*n*_WORD register clears the corresponding EICHEN[EICH*n*EN] field.

The EIM supports 1 error injection channel. Each channel:

- is assigned to a single memory array interface.
- intercepts the assigned memory read data bus and checkbit bus and injects errors by inverting the value transmitted for selected bits on each bus line.

On a memory read access, the applicable EICHD*n*_WORD registers define which bits of the read data and/or checkbit bus to invert.

Figure 21-1 depicts the interception and override of a 64-bit read data bus and an 8-bit checkbit data bus for an example memory array.

Chapter 22
Clocking

22.1 Introduction

This chapter describes the architecture for the system level clocks and includes the following information:

- System clock specifics
- Clock architecture
- Clock sources
- Clock monitoring
- Programmable clock dividers
- Clock control registers
- Progressive Clock Switching

NOTE

This chapter covers only clocks that are generated on the device. Protocol clocks that are provided by external devices are covered in the respective chapters where they are used.

The clock (system clock) for the computational shell (which includes the checker and two computational cores as well as all modules that run in the high speed domain), memories, and debug logic is independent from the peripheral clocks. This allows the system clock to be a frequency modulated clock to reduce electromagnetic emissions, while having a precise clock for the peripheral timer and communication functions. It also allows the system clock to be reduced during low computational activity periods to help reduce power consumption, while providing a fixed frequency to the communication interfaces and timers.

The MPC5746R boots from the internal 16 MHz RC oscillator (IRCOSC), and uses this as a backup clock in the event of a PLL or XOSC failure (if backup clock is enabled).

There are three possible ways to provide the source clock:

- External oscillator

- External crystal
- Internal 16 MHz RC oscillator.

The internal clocks are generated from one of these input sources. These output clocks of the PLLs, along with the XOSC and IRCOSC, can be selected to drive system peripherals depending on the configuration of the Auxiliary Clock Selectors (in the MC_CGM). The PHI1 output of PLL0 can also be used as the clock source for PLL1. A total of eleven clock selectors allows developers to select the PLL reference clocks, drive various system peripherals with an independent clock source, and select a clock source to drive the CLKOUT signal for off-chip use. One of the clock selectors is used exclusively for the system clock.

Each of the outputs of the module clock selectors has dividers which allows for even more clock frequency granularity with division factors up to 128 for a given group of peripherals. On selected outputs of the clock dividers are Clock Monitor Units (CMU) that are used to test the integrity of the clocks making sure that their frequencies stay within necessary operating limits. If any of the eleven CMUs detects an issue with the clock signal that is being monitored, an interrupt or system reset can be generated, depending on the CMUs configurations.

22.2 Clock generation

The top-level clock generation architecture is given in the following figure. All clock selectors and dividers in the figure are located and programmed in the Clock Generation Module (MC_CGM). All dividers are integer dividers (1, 2, 3, …, n) with the ranges shown in Clock generation .

NOTE

CMU_0…CMU_9, CMU_12 in Clock generation use IRCOSC for their reference clock. However, the user can select XOSC as the reference in CMU_0 for monitoring IRCOSC.

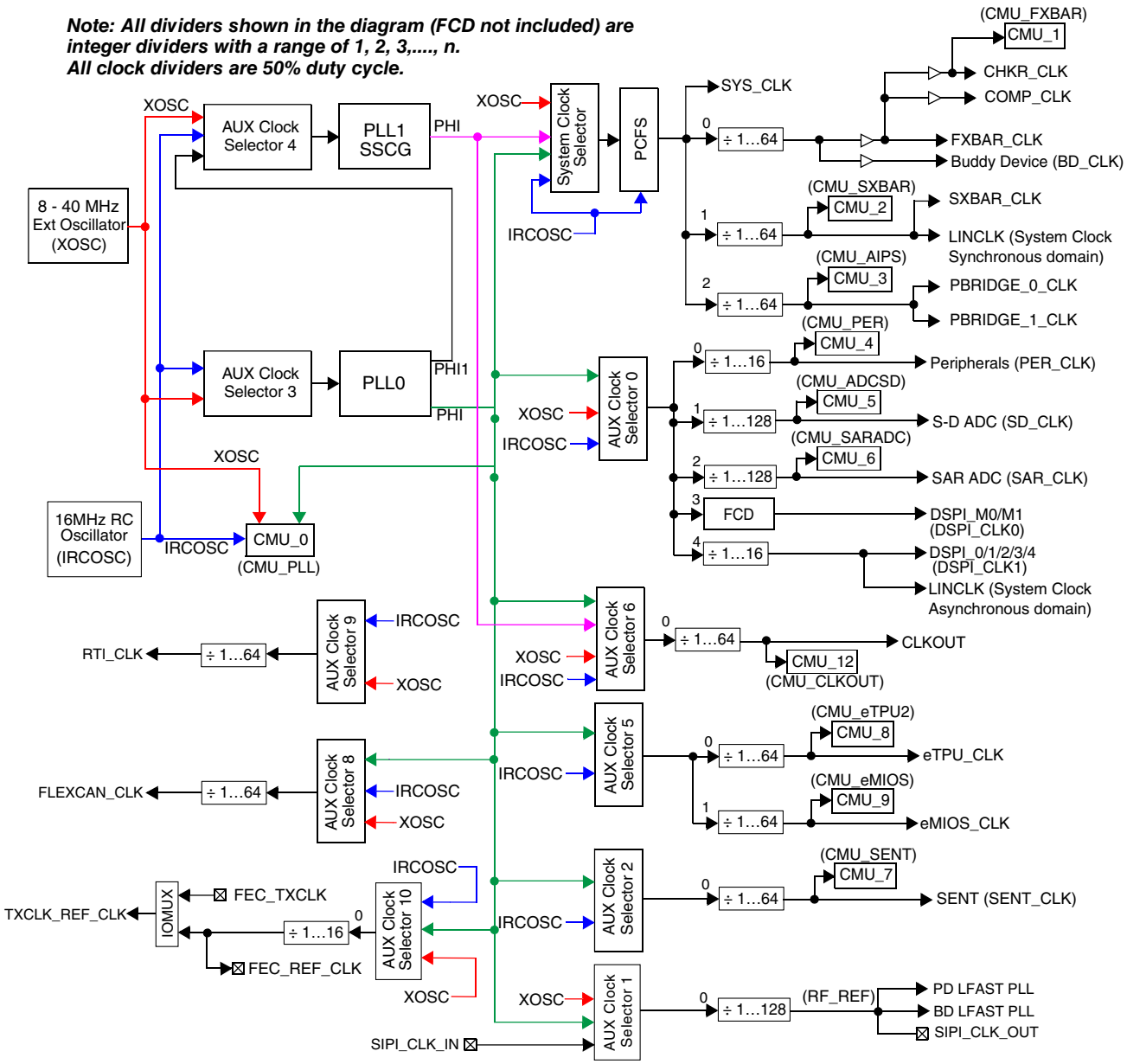


Figure 22-1. Clock generation

NOTE

Default clock source for all Auxillary Clock Selectors is the IRCOSC. User must configure MC_CGM_AC*n*_SC[SELCTL] before enabling the corresponding module functionality.

IRCOSC is the default clock for PLL1, but a PLL lock cannot be achieved using IRCOSC as the reference for PLL1.

NOTE

PBRIDGE_x_CLK may be referred to as "AIPS_CLK" elsewhere in this document.

NOTE

Do not bring out the ADC clock (XOSC + PLL0) or its component (XOSC and PLL0) on the CLKOUT pad when the ADC is in use using the same clock.

22.2.1 MC_CGM registers

Table 22-1 shows the relationship between the clocks in Clock generation and the MC_CGM registers.

NOTE

See Clock Generation Module (CGM) for specifics on register configurations.

Table 22-1. MC_CGM relationship to clocks

MC_CGM Registers	Register Description	Output Clock	CMU
MC_CGM_SC_DC0	System Clock Divider Configuration 0	Checker Core Computational Cores SYSCLK[1], (FXBAR, BD_CLK)	CMU_1 (CMU_FXBAR)
MC_CGM_SC_DC1	System Clock Divider Configuration 1	SXBAR_CLK	CMU_2 (CMU_SXBAR)
MC_CGM_SC_DC2	System Clock Divider Configuration 2	PBRIDGE_0_CLK PBRIDGE_1_CLK	CMU_3 (CMU_AIPS)
MC_CGM_AC0_DC0 MC_CGM_AC0_SC	Aux Clock 0 Divider Configuration 0 Aux Clock 0 Select Control	PER_CLK	CMU_4 (CMU_PER)
MC_CGM_AC0_DC1 MC_CGM_AC0_SC	Aux Clock 0 Divider Configuration 1 Aux Clock 0 Select Control	SD_CLK	CMU_5 (CMU_ADCSD)
MC_CGM_AC0_DC2 MC_CGM_AC0_SC	Aux Clock 0 Divider Configuration 2 Aux Clock 0 Select Control	SAR_CLK	CMU_6 (CMU_SARADC)
MC_CGM_AC0_DC3 MC_CGM_AC0_SC	Aux Clock 0 Divider Configuration 3[2] Aux Clock 0 Select Control	DSPI_CLK0	—
MC_CGM_AC0_DC4 MC_CGM_AC0_SC	Aux Clock 0 Divider Configuration 4 Aux Clock 0 Select Control	DSPI_CLK1 LINCLK	—
MC_CGM_AC1_DC0 MC_CGM_AC1_SC	Aux Clock 1 Divider Configuration 0 Aux Clock 1 Select Control	RF_REF (PD_LFAST_PLL, BD_LFAST_PLL, SIPI_CLK_OUT)	—
MC_CGM_AC2_DC0 MC_CGM_AC2_SC	Aux Clock 2 Divider Configuration 0 Aux Clock 2 Select Control	SENT_CLK	CMU_7 (CMU_SENT)

Table continues on the next page...

Table 22-1. MC_CGM relationship to clocks (continued)

MC_CGM Registers	Register Description	Output Clock	CMU
MC_CGM_AC3_DC0 MC_CGM_AC3_SC	Aux Clock 3 Divider Configuration 0 Aux Clock 3 Select Control	PLL0 Reference	—
MC_CGM_AC4_DC0 MC_CGM_AC4_SC	Aux Clock 4 Divider Configuration 0 Aux Clock 4 Select Control	PLL1 Reference	—
MC_CGM_AC5_DC0 MC_CGM_AC5_SC	Aux Clock 5 Divider Configuration 0 Aux Clock 5 Select Control	eTPU2_CLK	CMU_8 (CMU_eTPU2)
MC_CGM_AC5_DC1 MC_CGM_AC5_SC	Aux Clock 5 Divider Configuration 1 Aux Clock 5 Select Control	eMIOS_CLK	CMU_9 (CMU_eMIOS)
MC_CGM_AC6_DC0 MC_CGM_AC6_SC	Aux Clock 6 Divider Configuration 0 Aux Clock 6 Select Control	CLKOUT	CMU_12 (CMU_CLKOUT)
MC_CGM_AC8_DC MC_CGM_AC8_SC	Aux Clock 8 Divider Configuration 0 Aux Clock 8 Select Control	FLEXCAN_CLK	—
MC_CGM_AC9_DC MC_CGM_AC9_SC	Aux Clock 9 Divider Configuration 0 Aux Clock 9 Select Control	RTI_CLK	—
MC_CGM_AC10_DC MC_CGM_AC10_SC	Aux Clock 10 Divider Configuration 0 Aux Clock 10 Select Control	FEC_REF_CLK[3]	—

1. SYSCLK output is before the divider, so it is not directly monitored by CMU_1 (CMU_FXBAR).
2. MC_CGM_AC0_DC3 is a FCD.
3. FEC_REF_CLK is the output for the divider, but the IOMUX can select either FEC_TXCLK input signal or FEC_REF_CLK for TXCLK_REF_CLK generation.

22.3 System clock frequency limitations

The maximum frequency of operation for the device system level clocks is given in the following table.

NOTE

It is the responsibility of the user not to exceed these values.

Table 22-2. Maximum system level clock frequencies

System clock	Max frequency (MHz)
Cores (CHKR_CLK, COMP_CLK)	200
FXBAR, BD_CLK, System Ram Ctrl	200
SXBAR, System Ram Array, LINCLK (System Clock Synchronous domain)	100
PBRIDGE_0	50
PBRIDGE_0 (eTPU2 Slot)	200
PBRIDGE_1	50
eDMA	100
INTC (Interrupt Controller)	50
FLEXCAN_CLK	40

Table continues on the next page...

Table 22-2. Maximum system level clock frequencies (continued)

System clock	Max frequency (MHz)
DSPI_CLK0, DSPI_CLK1	80
PER_CLK	80
LINCLK (System Clock Asynchronous domain)	80
SAR_CLK	80
SD_CLK	16
SENT_CLK	80
FEC_REF_CLK, TXCLK, RXCLK	50, 25, 25
RF_REF (BD_LFAST_PLL, PD_LFAST_PLL)	26
RTI_CLK	50
eMIOS_CLK	100
CTU	100

In order to maintain synchronization between the different system clock branches (for example, SYS_CLK output from the System Clock Selector), there are limitations on the frequencies of those clocks. The system clocks must be binary multiples of each other with the frequency relationships described in the electrical specifications.

NOTE

The MC_CGM_SC_DCn registers must be configured with dividers that respect the clock ratios 1:2:4, for FXBAR_CLK, SXBAR_CLK, and PBRIDGE_n_CLK respectively.

Table 22-3. Examples of valid clock domain combinations (FXBAR_CLK / SXBAR_CLK / PBRIDGE_n_CLK)

FXBAR_CLK	SXBAR_CLK	PBRIDGE_n
200 MHz	100 MHz	50 MHz
160 MHz	80 MHz	40 MHz
100 MHz	50 MHz	25 MHz

22.3.1 JTAG frequencies

The following table shows the maximum frequencies of the JTAG interface.

Table 22-4. JTAG frequencies

Device	Configuration	e200z4 max. internal JTAG frequency	Buddy Device max. RWA internal frequency	Maximum external JTAG frequency	Notes
PD[1]	Unconfigured clock	—	—	See electrical characteristics for maximum usable external JTAG frequency	Assumes 16 MHz clock with +/- 1.5% tolerance, and JTAG usable at 1/2 CPU frequency
	Clock configured to 200 MHz	100 MHz			JTAG usable at 1/2 CPU frequency
ED[2]	BD only powered	—	22.4 MHz		BD clocked by internal 64 MHz RC clock (RCOSC64) with +/- 30% tolerance and BD RWA usable at 1/2 frequency.
	PD and BD powered, unconfigured clock	—	—		ED clocked by internal 8 MHz RC clock (IRC) with +/- 1.5% tolerance, and BD RWA usable at 1/2 frequency
	PD and BD powered, PD clock configured to 200 MHz	100 MHz	100 MHz		100 MHz JTAG achievable using LFAST mode debug, or s/w mode JTAGM. JTAG messages are generated internally by JTAGM and stay within BD and PD domain. The external tool interface is LFAST only

1. PD = Production Die
2. ED = Emulation and Debug devices = Production die + Buddy die

22.4 Default clock configuration

At power up, the IRCOSC is the default clock for the entire system. All system clock dividers are set to divide by two at power-on reset.

22.5 Clock sources

The following clock sources are described in this section:

- PLL0
- PLL1 (FMPLL)
- External Oscillator (XOSC)
- External Clock (EXTAL Bypass)
- Internal 16 MHz RC Oscillator (IRCOSC)

22.5.1 PLL

The PLL system in the MPC5746R is a Dual PLL that provides separate system and peripheral clocks. The PLLs are disabled after power on and must be enabled by software. The block diagram for the Dual PLL appears in Figure 22-2.

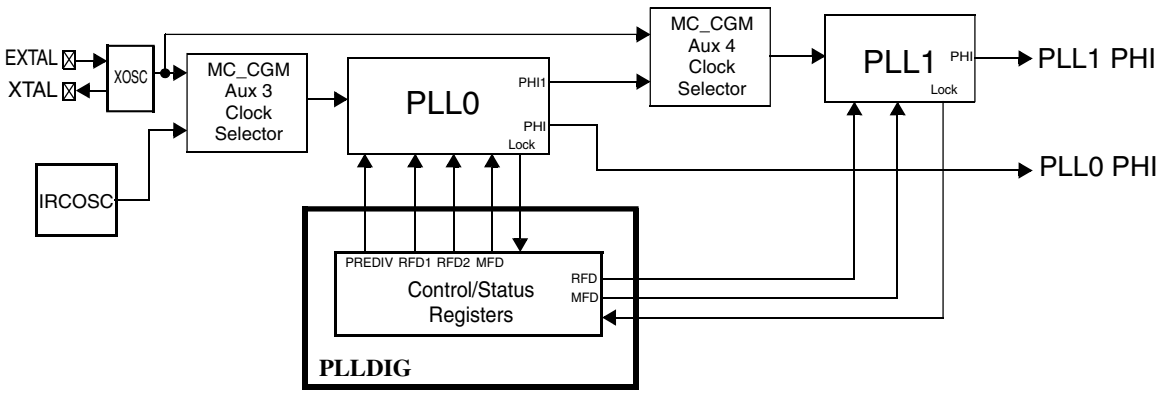


Figure 22-2. MPC5746R Dual PLL digital interface block diagram

22.5.1.1 PLL0 - base PLL (non-FM)

PLL0 is the primary PLL. This PLL is used to source a non-FM clock to the MPC5746R peripherals and also the reference clock to PLL1.

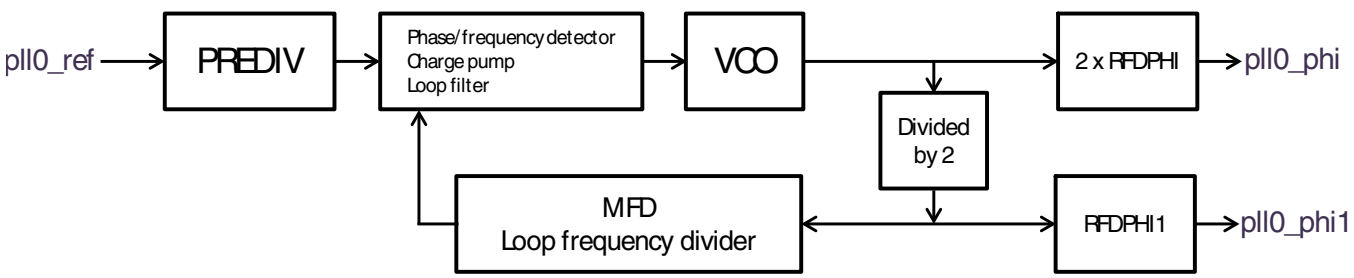


Figure 22-3. PLL0 Block Diagram

22.5.1.1.1 PLL0 input clocks

The possible input clock sources for PLL0 are the XOSC, IRCOSC, and EXTAL Bypass. The EXTAL Bypass input is the EXTAL pin. AUX Clock Selector 3 selects which input clock will be used as the source for PLL0. The selection between XOSC and EXTAL Bypass is made via the XOSC_CTL register of the XOSC module.

22.5.1.1.2 PLL0 output clocks

The output clocks from PLL0 are PHI and PHI1. The PHI output clock drives various peripheral clocks and the system clock when selected in the MC_CGM. The PHI1 output provides one of the input references for PLL1.

22.5.1.2 PLL1 - FMPLL

PLL1 is a Frequency Modulated PLL (FMPLL) that is typically used to drive the system clock. PHI is the output of PLL1 which drives the System Clock Selector of the MC_CGM.

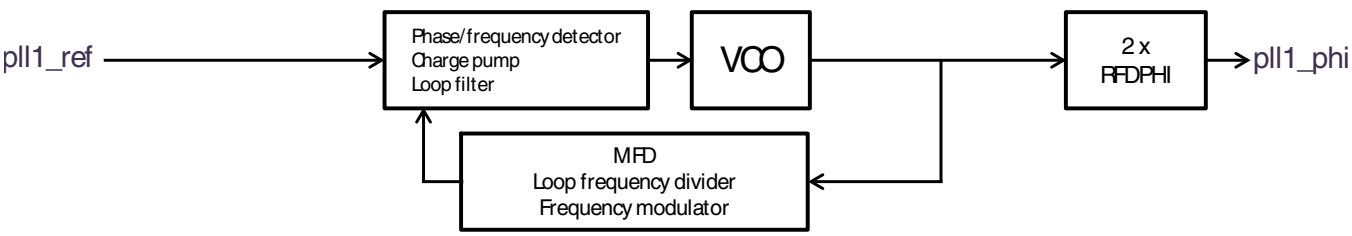


Figure 22-4. PLL1 Block Diagram

22.5.1.2.1 PLL1 input clocks

The possible input clock sources for PLL1 are XOSC, PLL0_PHI, and EXTAL Bypass. The EXTAL Bypass input is the EXTAL pin, which "bypasses" the XOSC output. AUX Clock Selector 4 selects which input clock is used as the source for PLL1. The selection between XOSC and EXTAL Bypass is made via the XOSC_CTL register of the XOSC module.

22.5.1.2.2 PLL1 output clocks

The output clock from PLL1 is the PHI clock, which can drive the system clock if the System Clock Selector of the MC_CGM is configured to do so. The PHI output clock contains a fractional divider that can be applied to the loop divide of the PLL to achieve good granularity in the PLL1 PHI output clock frequency.

22.5.1.2.3 PLL register interface

This device has a single digital interface for the dual PLLs. The digital interface provides register control of all features of PLL0 and PLL1. Details are provided in the "Dual PLL Digital Interface" chapter.

22.5.1.2.4 PLL register write protection

The PLL registers with write protection are defined in PLL_DIG protected registers.

22.5.1.3 PLL register reset values

All reset values for the PLLDIG registers that are device specific are shown in the following table. All other register reset values are shown in the Dual PLL Digital Interface chapter.

Table 22-5. Dual PLL Digital Interface register reset values

Offset	Register	Reset Value
0008h	PLL0DV – PLL0 Divider Register	0000_0000h
0028h	PLL1DV – PLL1 Divider Register	0000_0000h

22.5.1.4 PLL initialization information

Coming out of reset PLL0 and PLL1 are disabled per the DRUN mode configuration register, MC_ME_DRUN_MC.

1. Configure PLL0 and related modules.
   a. With PLL0 disabled, program the PLL0 clock source in MC_CGM_AC3_SC[SELCTL]. Default source is the 16 MHz IRCOSC. Write MC_CGM_AC3_SC[SELCTL] = 1 to select the XOSC as source.
   b. Program PLL0DV[PREDIV], PLL0DV[RFDPHI1], PLL0DV[MFD], and PLL0DV[RFDPHI].
   c. If desired, modify the XOSC_CTL[EOCV] to adjust the external oscillator stabilization count, used when the external oscillator is turned on (by the MC_ME).
   d. If necessary, modify the CMU frequency meter values (CMU_MDR and CMU_FDR) used to monitor the XOSC frequency. The XOSC frequency is compared to the IRCOSC source when the XOSC is turned on.
2. Turn on the XOSC and PLL0.
   a. Configure a mode configuration for turning on PLL0 and the XOSC. In a mode configuration register (for example, MC_ME_DRUN_MC) set MC_ME_<*mode*>_MC[XOSCON] = 1 and

MC_ME_<*mode*>_MC[PLL0ON] = 1. If desired, also set MC_ME_<*mode*>_MC[SYSCLK] = 2 for this new mode configuration to use PLL0 (PLL0 PHI output) as the sysclk.

b. Enter that mode by two writes to MC_ME_CTRL register to enter that mode. This is required even if entering the same mode.

3. Wait for the mode transition to complete.

a. Wait for the mode transition to complete by polling MC_ME_GS[S_MTRANS] or enabling an interrupt for flag MC_ME_IS[I_MTC]. A timer, even if it is the watchdog, should be used to make sure the mode transition completes. Mode transition will NOT complete until:

- The XOSC counter expires (if the new mode configuration changes MC_ME_<*mode*>_MC[XOSCON] = 1), and
- PLL is locked (if the new mode configuration changes MC_ME_<*mode*>_MC[PLL0ON] = 1)

b. Confirm the desired target mode was entered by checking the status of MC_ME_GS[S_CURRENT_MODE].

4. Configure PLL1 and related modules.

a. With PLL1 disabled, program PLL1 clock source in MC_CGM_AC4_SC[SELCTL]. Default is MC_CGM_AC4_SC[SELCTL] = 1 which selects XOSC.

b. Program PLL1DV[MFD] and PLL1DV[RFDPHI].

c. If required, program the PLL1FM register with the desired frequency modulation parameters and enable FM modulation, PLL1FM[MODEN] = 1. The PLL1FM register must not be written after PLL1 is enabled and operating in normal mode.

5. Turn on PLL1.

a. Configure a mode configuration for turning on PLL1, and keeping the XOSC and PLL0 on. In a mode configuration register (for example, MC_ME_DRUN_MC), initialize MC_ME_<*mode*>_MC[XOSCON] = 1, MC_ME_<*mode*>_MC[PLL0ON] = 1 and MC_ME_<*mode*>_MC[PLL1ON] = 1. If desired, also set MC_ME_<*mode*>_MC[SYSCLK] = 4 for this new mode configuration to use PLL1 as the sysclk.

b. Enter that mode by two writes to ME_CTRL register to enter that mode. This is required even if re-entering the same mode.

6. Wait for the mode transition to complete.
   a. Wait for the mode transition to complete by polling MC_ME_GS[S_TRANS] or enabling an interrupt for flag MC_ME_IS[I_MTC]. A timer, even if it is the watchdog, should be used to make sure the mode transition completes. Mode transition will NOT complete until:
      - XOSC counter expires (if the new mode configuration changes XOSCON to 1), and
      - PLL1 is locked (if the new mode configuration changes MC_ME_<*mode*>_MC[PLL1ON] = 1).
   b. Confirm the desired target mode was entered by checking the status of MC_ME_GS[S_CURRENT_MODE].

NOTE

See the "Mode Entry Module (MC_ME)" chapter in this *Reference Manual* for details.

22.5.2 External oscillator (XOSC)

The external oscillator (XOSC) is provided as a reference for the on-chip PLLs. It can also can be used as a clock source to the ADCs, timer, serial interfaces, RTI, LFAST, and PLL reference as well as a system clock source. It is available for observation on the CLKOUT pin, and is used as a reference to calibrate the IRCOSC frequency.

The external oscillator allows a crystal or external clock to be used as the reference clock for the microcontroller. The XOSC has the following features:

- Reference clock to PLL0 and PLL1
- Reference clock to CMU_0 (IRCOSC trimming)
- Option to drive the FlexCAN protocol clock from the XOSC through AUX Clock Selector 8
- Provides support for 8 MHz to 40 MHz crystal inputs

22.5.2.1 XOSC register write protection

The XOSC registers have write protection as shown in OSC_DIG protected registers.

22.5.2.2 XOSC reset value

The following table shows the default reset value for the XOSC registers.

Table 22-6. XOSC register reset values

Offset	Register	Reset Value
0x00	XOSC_CTL-XOSC Control Register	0030_0000h

22.5.3 Internal 16 MHz RC Oscillator (IRCOSC)

The MCU has a 16 MHz RC oscillator which is always enabled and can be used as the clock source for both PLLs. It is used as the clock source for the FCCU, SWT, and SIUL2 input filters, and it can also be used as a clock source for LINFlex, DSPI both ADCs and peripheral clock as well as the system clock. The IRCOSC is the default system clock after reset. The register interface is used for user trimming of the oscillator and dividing the output from 1 to 32.

22.5.3.1 IRCOSC register interface

This device has a dedicated digital interface for the IRCOSC. The digital interface provides a register for fine tuning the IRCOSC, 6-bit factory trim value frequency by the user. Other registers allow reading of the settings of IRCOSC temperature sensor, voltage regulator and capacitor trimming values that are set by the factory. The IRCOSC is always enabled when the device is powered.

22.5.3.2 IRCOSC reset value

The following table shows the default reset value for the IRCOSC registers.

Table 22-7. IRCOSC register reset values

Offset	Register	Reset Value
0004	NT-IRCOSC Native Trimming Register	0000_0000h

22.5.3.3 IRCOSC register write protection

The IRCOSC registers do not utilize write protection.

22.5.3.4 IRCOSC trimming

The IRCOSC requires trimming to meet accuracy requirements. During the initial phases of a reset sequence triggered by a power-on or destructive reset, the IRCOSC runs at its untrimmed frequency value. A trim value that was determined during factory test, and subsequently stored in flash memory, is loaded during reset via the SSCM to the IRCOSC_CTL[USER_TRIM] (see the "IRCOSC Digital Interface" chapter for details).

22.6 Peripheral Clocks

The following figure shows the clock distribution to the core and peripheral modules.

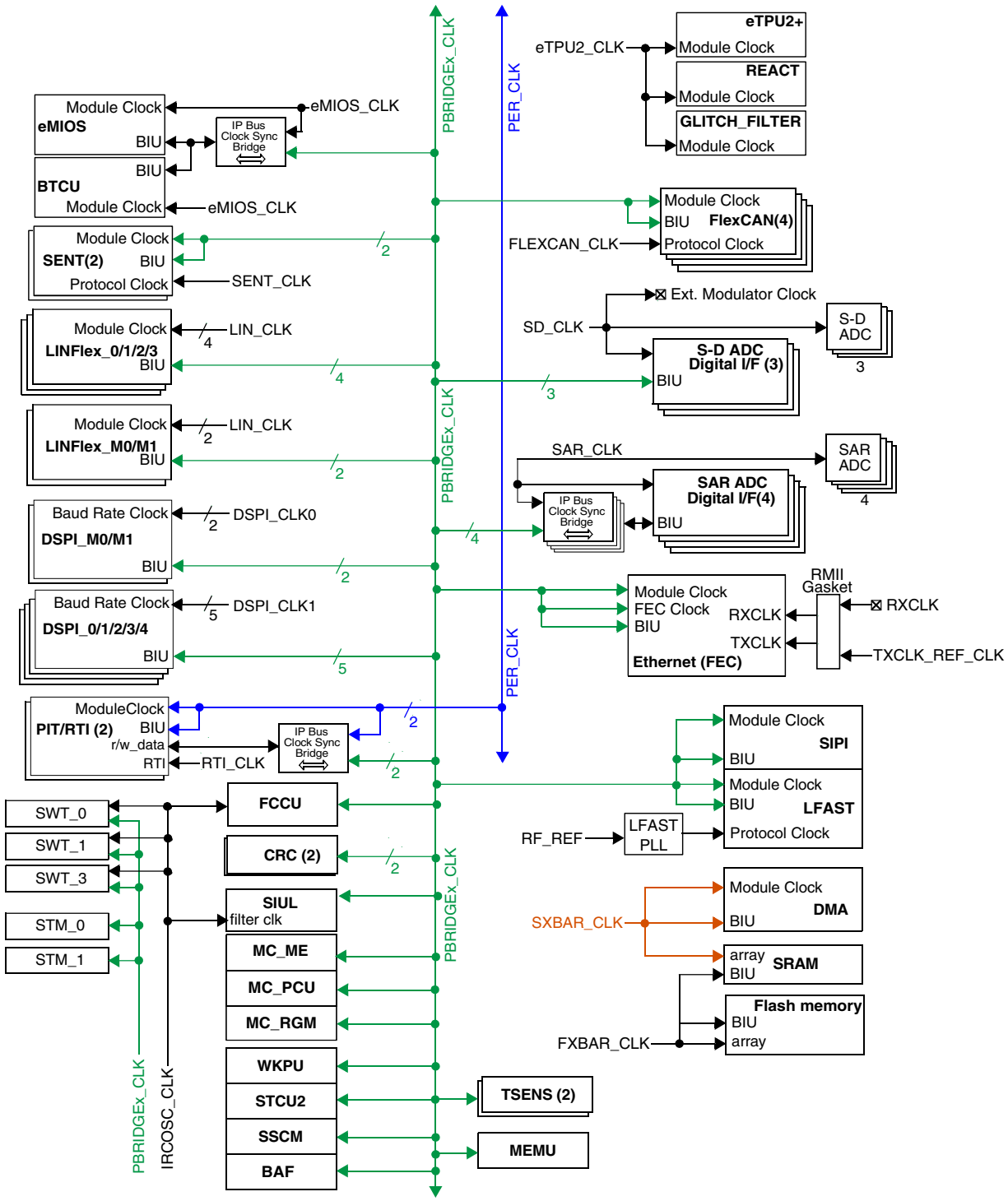


Figure 22-5. Clock distribution

22.6.1 LFAST clocking

The MPC5746R includes an LFAST module to support high speed device communications. A single LFAST PLL supports high speed operation of the LFAST module. For low speed LFAST operation, the reference clock is used directly by the LFAST module.

The source of the LFAST PLL may be the PLL0_PHI or the external oscillator (XOSC) or input from the external LFAST device via the SIPI_CLK_IN package signal. When this reference clock is generated internally, it can also be output on the SIPI_CLK_IN signal. The SIPI_CLK_IN signal connection appears in the following figure. Input/ Output direction of the SIPI_CLK_IN/OUT signal is controlled by the MSCR in the SIUL2.

On Emulation and Debug Devices (ED), the LFAST module on the Buddy Die (BD) is used and connected to the package debug pins instead of the Production Die (PD) debug LFAST module. This module has a separate dedicated LFAST PLL, although as with non EDs', the SIPI_CLK_OUT/IN pin only supports output of the reference clock to a connected LFAST capable tool.

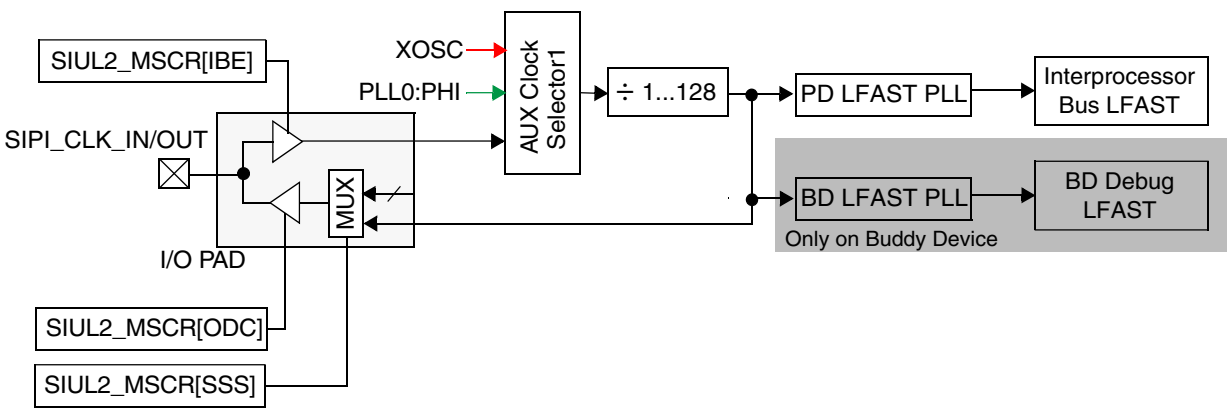


Figure 22-6. Device LFAST SysClk

22.6.2 Ethernet clocking

Both internal and external clocks sources for both MII and RMII Ethernet interface modes are supported. The MII clocks (TXCLK/RXCLK) are 25 MHz, and the RMII clock (REF_CLK) is 50 MHz. Figure 22-7 shows the connections for the Ethernet clocks.

The RXCLK pin is an input, used in MII mode only, and ignored by the RMII gasket in RMII mode.

The TXCLK pin is an input, used in MII mode only, but is shared with the REF_CLK input of the RMII gasket. Therefore, the Aux clock selector is required to select between TXCLK and REF_CLK for MII and RMII modes.

The REF_CLK can be generated internally or input to the device. For MII mode, the FEC_REF_CLK can be used for the 25 MHz reference for the external Ethernet PHY, which provides the TXCLK/RXCLK clocks back to the device. For RMII mode, FEC_REF_CLK can be generated internally and fed to both the PHY device and to the RMII gasket by enabling the REF_CLK pad input and output buffers. REF_CLK can also be provided externally as an input to the device.

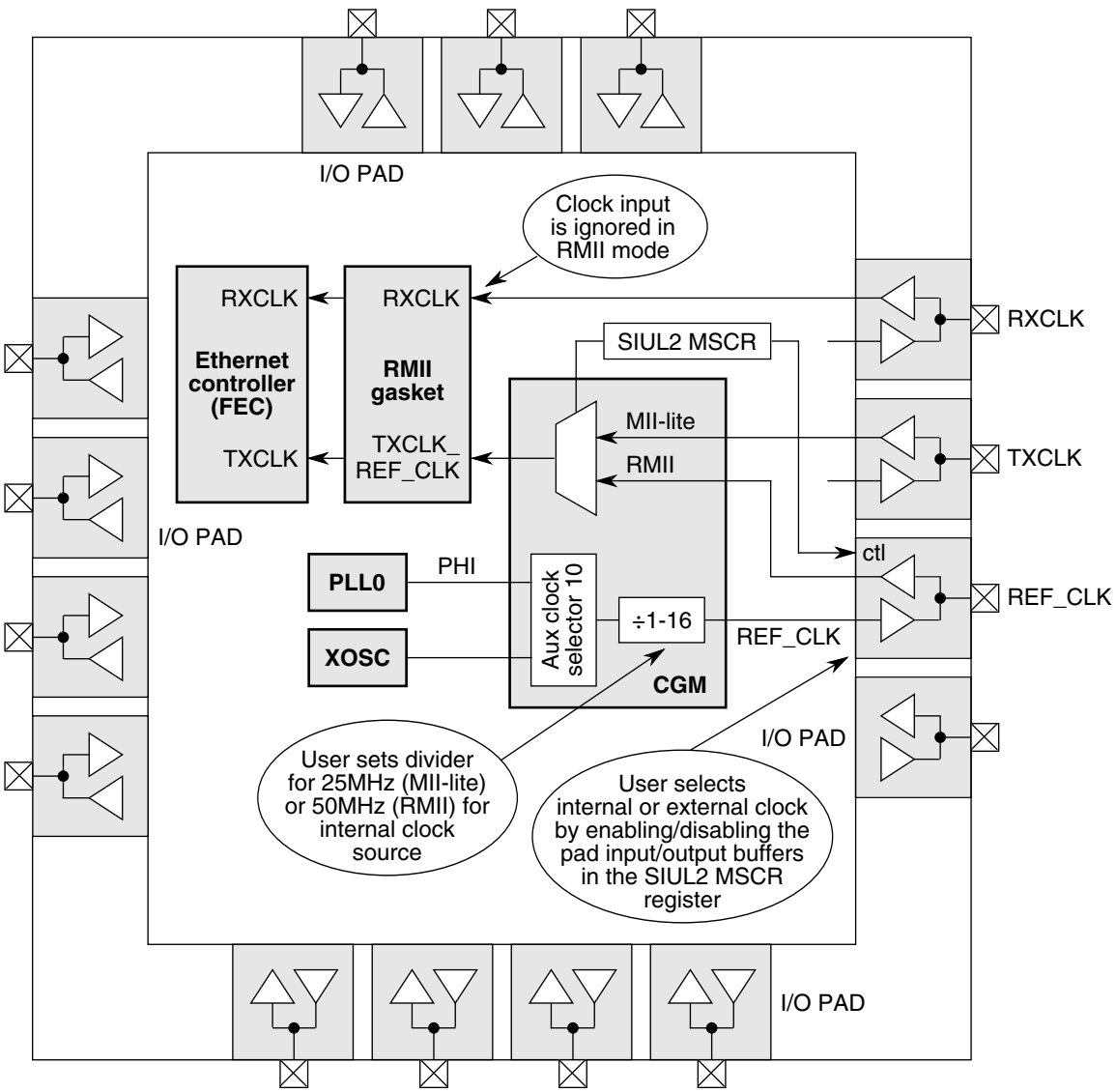


Figure 22-7. Ethernet clocking diagram

22.6.3 Sigma-Delta ADC clocking

The analog portion of the Sigma-Delta ADC on the device works only with an internal modulator. The internal clock (SD_CLK, see Figure 22-5) is always used when the internal modulator in the S-D ADC is enabled.

Each S-D ADC is connected as shown in Figure 22-8, with the SD_CLK being common to all modules.

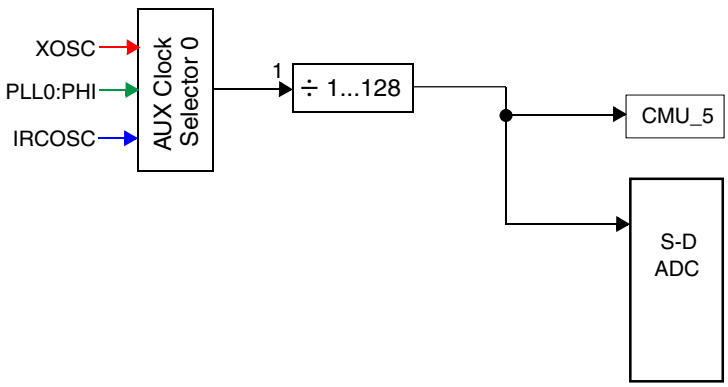


Figure 22-8. Device S-D ADC clocking

22.6.4 LINFlex clocking

The LINFlex protocol clock (LINCLK) domain selection may be configured via soft DCF client programming, as described in LINCLK_DOMAIN_SELECTION - BAF LINCLK Synch/Asynch Clock domain selection DCF Client.

Depending on the application's LINCLK source selection, the LinFlex protocol clock will operate as follows:

- System Clock Synchronous mode: the LINCLK source is connected to the Synchronous System Clock Selector / Clock Divider 1

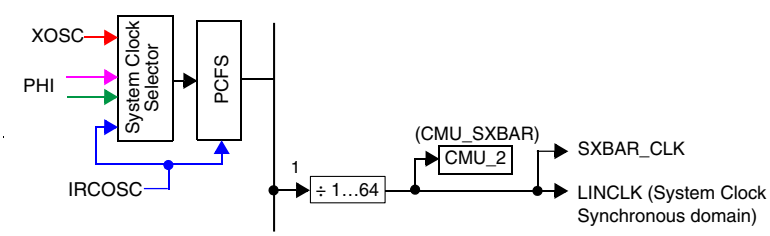


Figure 22-9. System Clock Synchronous domain

- System Clock Asynchronous mode: the LINCLK source is connected to the Auxiliary Clock Selector 0/ Clock Divider 4

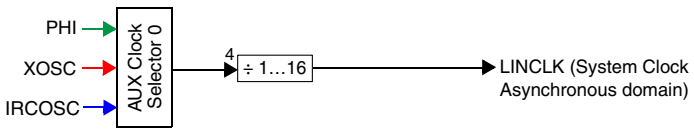


Figure 22-10. System Clock Asynchronous domain

In either mode the LINCLK must be enabled by setting the Auxiliary Clock 0 Divider Configuration Register (MC_CGM_AC0_DC4) Divider Enable (DE) bit to 0b1.

NOTE

The System Clock Selector / Clock Divider 1 (MC_CGM_SC_DC1) Divider Enable (DE) bit only controls SXBAR_CLK.

The Auxiliary Clock Selector 0 / Clock Divider 4 (MC_CGM_AC0_DC4) Divider Enable (DE) bit always controls LINCLK, regardless of the selected LINFlex operation mode, and defaults to 0b0 (LINCLK disabled).

22.7 Clock monitoring

For all safety critical clocks, the microcontroller detects a missing clock or incorrect frequency. To achieve this clock monitoring units (CMUs) are used. Each CMU is programmed independently. The IRCOSC and XOSC are used as the clock monitor references. Detailed information on the CMUs can be found in the Clock Monitor Unit chapter.

22.7.1 CMU configuration

This section explains the CMU configuration.

Figure 22-11 shows the block diagram for CMU_0 on the MPC5746R.

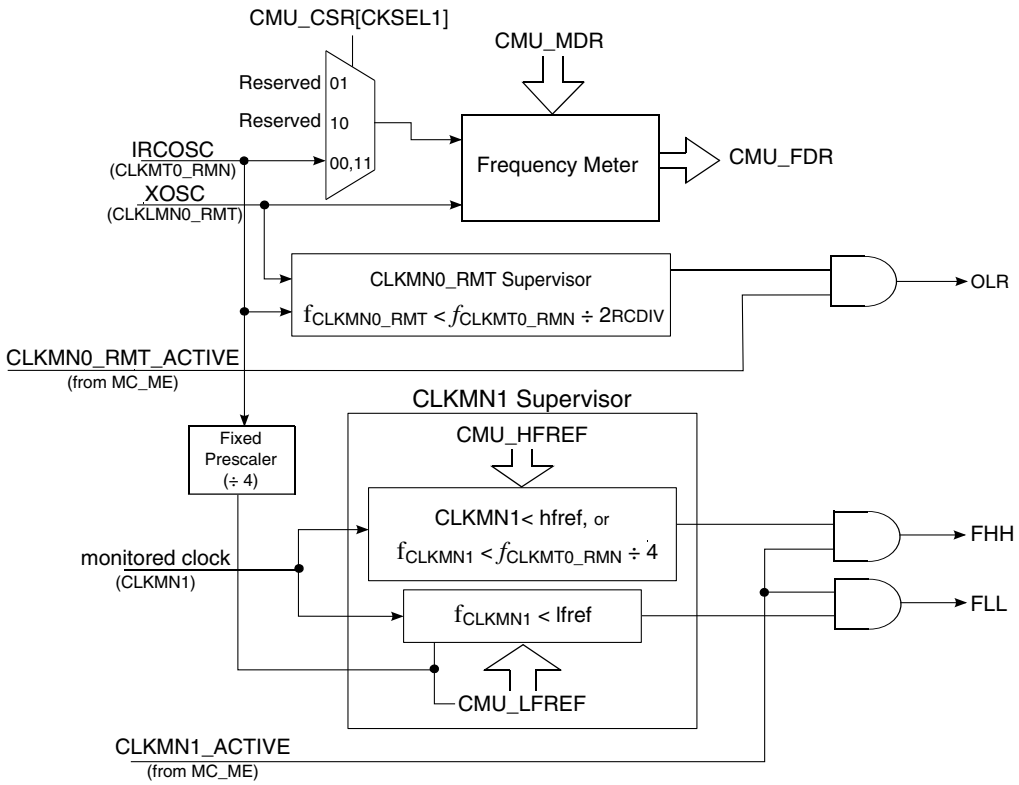


Figure 22-11. CMU_0 Block Diagram

Figure 22-12 shows the block diagram for CMU_[1:9,12] on the MPC5746R.

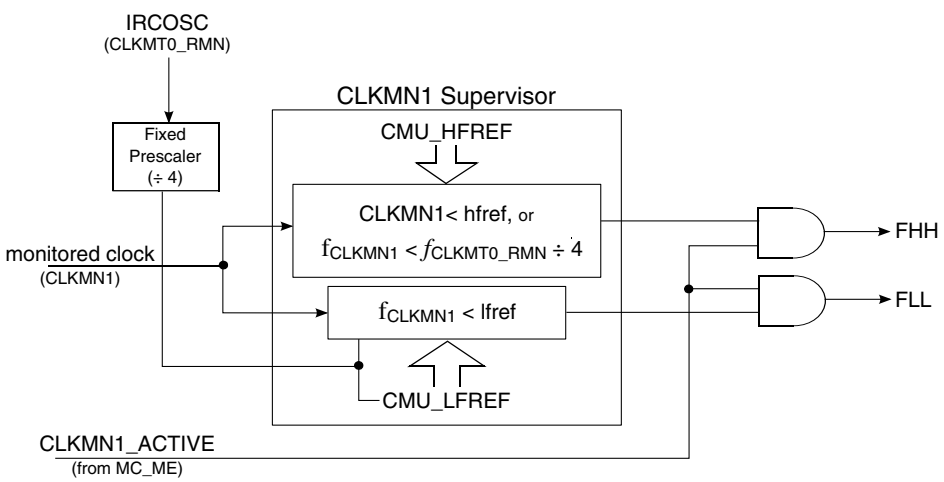


Figure 22-12. CMU_[1:9,12] Block Diagram

22.7.1.1 Clock input sources

The following table shows the clocks that are monitored by each CMU. These signals are connected internally on the chip, but are not accessible via pins on the boundary of the device. IRCOSC is the reference clock for all clock monitors. Only CMU_0 implements the XOSC monitor. CMU_0 uses the IRCOSC clock to measure if the XOSC is too low. CMU_0 can also be used to calibrate the IRCOSC frequency using the XOSC. All other CMUs are configured identically.

Table 22-8. Clock input sources

Clock module	Monitored clock
CMU_0 (CMU_PLL)	XOSC, IRCOSC, PLL0:PHI
CMU_1 (CMU_FXBAR)	CHKR_CLK
CMU_2 (CMU_SXBAR)	SXBAR_CLK
CMU_3 (CMU_AIPS)	PBRIDGE_0_CLK, PBRIDGE_1_CLK
CMU_4 (CMU_PER)	PER_CLK
CMU_5 (CMU_ADCSD)	SD_CLK
CMU_6 (CMU_SARADC)	SAR_CLK
CMU_7 (CMU_SENT)	SENT_CLK
CMU_8 (CMU_eTPU2)	eTPU_CLK
CMU_9 (CMU_eMIOS)	eMIOS_CLK
CMU_12 (CMU_CLKOUT)	CLKOUT

22.7.1.2 CMU registers and field availability

This table specifies which registers and fields are available on a given CMU for this device

Table 22-9. CMU registers availability

Address Offset	Register	CMU Instance	Note
0000h	CMU_CSR	All	CMU_CSR[SFM], CMU_CSR[RCDIV], and CMU_CSR[CLSEL1] are only used in CMU_0. CMU_CSR reset value is 0x06 for CMU_0 and 0x00 for all other CMU instances.
0004h	CMU_FDR	CMU0	—
0008h	CMU_HFREFR	All	—
000Ch	CMU_LFREFR	All	—
0010h	CMU_ISR	All	CMU_ISR[OLRI] is only used in CMU_0.

Table continues on the next page...

Table 22-9. CMU registers availability (continued)

Address Offset	Register	CMU Instance	Note
0014h	Reserved	—	—
0018h	CMU_MDR	CMU0	—

22.7.1.3 CMU register write protection

The CMU registers have write protection as shown in CMU protected registers.

22.7.2 External oscillator (XOSC) monitor

The XOSC frequency is compared to a minimum value limit. If the measured XOSC frequency is below the limit, a flag is set, and an interrupt is generated, if enabled. Frequency limits are given in the MPC5746R Data Sheet.

22.7.3 Internal RC oscillator (IRCOSC) monitor

The period of the IRCOSC can be measured in CMU_0, using the XOSC as a reference. This allows for application trimming of the IRCOSC frequency.

22.8 Loss of system clock behavior

This device has built-in mechanisms for detecting loss of the oscillator or PLL clocks, and provides several options for reaction to a loss of clock in the application. Figure 22-13 gives a high-level view of the loss of clock logic.

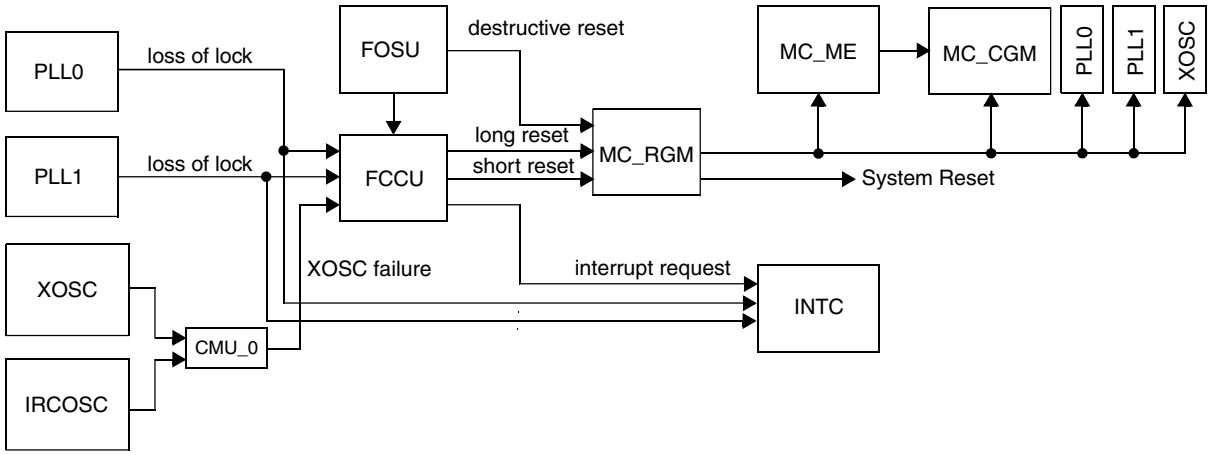


Figure 22-13. Loss of clock logic

22.8.1 Loss of PLL or XOSC

As shown in Figure 22-13, each loss of lock signal from the PLLs or XOSC failure are monitored by the FCCU. For the event when a clock failure occurs, the FCCU can be programmed to generate a short or long reset sequence or simply to generate an interrupt. In the case of the interrupt, only the unpredictable free running clock is still provided by the PLL. There is no automatic system clock switch in this case, and the user is required to program the switch through the Mode Entry module (MC_ME). Therefore, it is strongly recommended that the interrupt reaction to a PLL loss of lock is not selected when the PLL is used as the source of the system clock.

If a long or short reset is selected: the PLL, XOSC, MC_ME, and MC_CGM are reset to their default states, and the system clock is switched to the IRCOSC.

22.8.2 Loss of IRCOSC

The frequency of the IRCOSC is monitored by the frequency meter in CMU_0. There is no automated trigger of an FCCU error condition if the IRCOSC fails. Since the IRCOSC is the boot and backup clock, a failure is a catastrophic failure.

22.8.3 PLL loss of lock interrupt

Always write 1 to both PLL0CR[LOLIE] and PLL1CR[LOLIE] to keep the loss of lock interrupt function enabled.

The main interrupt functionality is determined by the FCCU via fault input NCF[32] for PLL0 and fault input NCF[33] for PLL1.

22.9 Progressive clock switching

The Progressive Clock Frequency Switching (PCFS) blocks are shown in the system-level clock diagram in Clock generation . They are used for ramping the system and peripheral clocks and for obtaining the precise clock frequency input for the SXBAR, FXBAR and Buddy Device (BD_CLK) module. See the "Progressive system clock switching" section in the "Clock Generation Module (MC_CGM)" chapter for details on PCFS.

Chapter 23
Dual PLL Digital Interface (PLLDIG)

23.1 Introduction

The Dual PLL system composed of PLL0 and PLL1 analog blocks and the digital interface (PLLDIG). The two analog PLL blocks are cascaded, with the PHI1 output of PLL0 feeding the clock input of PLL1.

A key feature of the dual PLL architecture is the ability to drive peripherals from the PLL0 PHI output, which is non-modulated and independent of the core clock frequency. The core and platform clocks are driven by PLL1.

23.2 Block Diagram

Please refer to the “Clocking chapter” of this *Reference Manual* to see the block diagram.

23.3 Features

The Dual PLLDIG has the following features:

- Supports dual PLL in cascaded mode with PLL0 clock out as reference clock to PLL1.
- Reference clock pre-divider for finer frequency synthesis resolution.
- Reduced frequency divider for decreasing the PLL0/PLL1 output clock frequency without causing the PLLs to lose lock.
- Programmable frequency modulation on PLL1.
- Lock detect circuitry reports when the PLLs have achieved frequency lock, and continuously monitors lock status to report loss of lock conditions.
    - The loss of lock indication is sent to the FCCU via the system glue logic.

23.4 Modes of operation

The operating mode of the PLLs is determined by the value of PLL*n*CR[CLKCFG]. The various modes of operation for the PLL are:

- Bypass mode with reference, and both PLLs disabled
- Normal mode with reference, and either PLL0 or both PLLs enabled

NOTE

Mode changes are controlled by configuring the MC_ME Mode Configuration registers (MC_ME_<*mode*>_MC).

Bypass mode is defined as the mode where all clocks are driven by an external clock driving the EXTAL signal. Normal mode is defined as the mode where the clocks are driven by the PLLs. When using a crystal for the reference clock, reset should remain asserted until the oscillator has stabilized.

When PLL0:PHI1 is the source for PLL1, and PLL0 is powered down and working in bypass mode, PLL1 should be bypassed and powered down. PLL1 should be configured to work in normal mode only after PLL0 has achieved lock (for example, bypass with PLL0 running or PLL0 in normal mode, PLL*n*SR[LOCK] = 1).

23.4.1 Bypass mode with reference, both PLLs disabled

In this mode, the PLLs are bypassed and system and peripheral clocks (PLL clock outputs) are driven from the external clock reference. The user must supply a clock that is within the appropriate frequency range. This mode is indicated by PLL0CR[CLKCFG] = 00b and PLL1CR[CLKCFG] = 00b. This is also the default power up mode of the PLLs.

23.4.2 Normal mode with reference, PLL0 or both PLLs enabled

In normal mode, PLL0 receives an input clock from the reference which can be prescaled by the pre-divider. PLL0 multiplies the frequency to create the PLL0 output clock. The user must supply a crystal that is within the appropriate frequency range, the crystal manufacturer recommended external support circuitry and short signal route from the MCU to the crystal.

PLL0 generates a non-modulated clock which drives PLL0 PHI1. PLL1 can generate a frequency modulated clock or a non-modulated clock (for example, locked on a single frequency). The modulation rate, modulation depth, and modulation enable are programmed by configuring the PLLFM register.

NOTE

While powering down the PLLs and if PLL0:PHI1 is used as the source for PLL1, user must ensure that PLL1 is powered down first followed by powering down PLL0 for proper shut off.

Configuring the appropriate MC_ME_<*mode*>_MC register to start PLL1 should only be done after PLL0 has achieved lock.

23.5 Memory map and register definition

This section provides the memory map and detailed descriptions of all registers for configuring the PLLs. The table below shows the memory map. Addresses are given as offsets from the module base address. All registers can be accessed using 8-bit, 16-bit or 32-bit addressing.

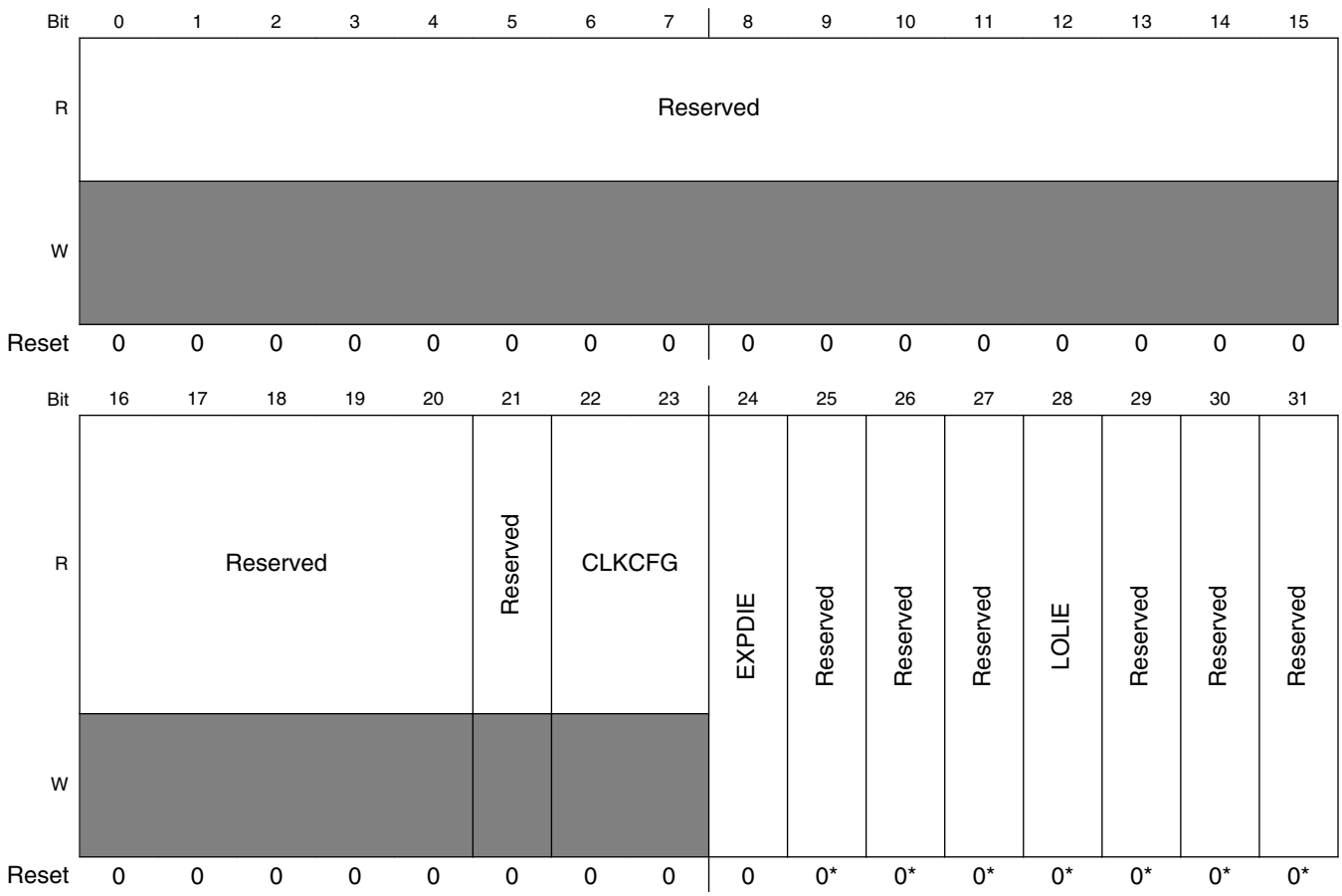
PLLDIG memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	PLLDIG PLL0 Control Register (PLLDIG_PLL0CR)	32	R/W	See section	23.5.1/668
4	PLLDIG PLL0 Status Register (PLLDIG_PLL0SR)	32	R/W	See section	23.5.2/670
8	PLLDIG PLL0 Divider Register (PLLDIG_PLL0DV)	32	R/W	See section	23.5.3/672
20	PLLDIG PLL1 Control Register (PLLDIG_PLL1CR)	32	R/W	See section	23.5.4/674
24	PLLDIG PLL1 Status Register (PLLDIG_PLL1SR)	32	R/W	See section	23.5.5/676
28	PLLDIG PLL1 Divider Register (PLLDIG_PLL1DV)	32	R/W	See section	23.5.6/677
2C	PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)	32	R/W	See section	23.5.7/679
30	PLLDIG PLL1 Fractional Divide Register (PLLDIG_PLL1FD)	32	R/W	0000_0000h	23.5.8/681

23.5.1 PLLDIG PLL0 Control Register (PLLDIG_PLL0CR)

The PLL0CR is shown in the following table.

Address: 0h base + 0h offset = 0h



* Notes:
- This field can be written at any time, but writes are ignored. Read returns previously written value.
- LOLIE field: See the "Clocking" chapter for details on configuring this field.

PLLDIG_PLL0CR field descriptions

Field	Description
0–20 Reserved	This field is reserved.
21 Reserved	This field is reserved.
22–23 CLKCFG	Clock Configuration This field indicates the operational state of PLL.

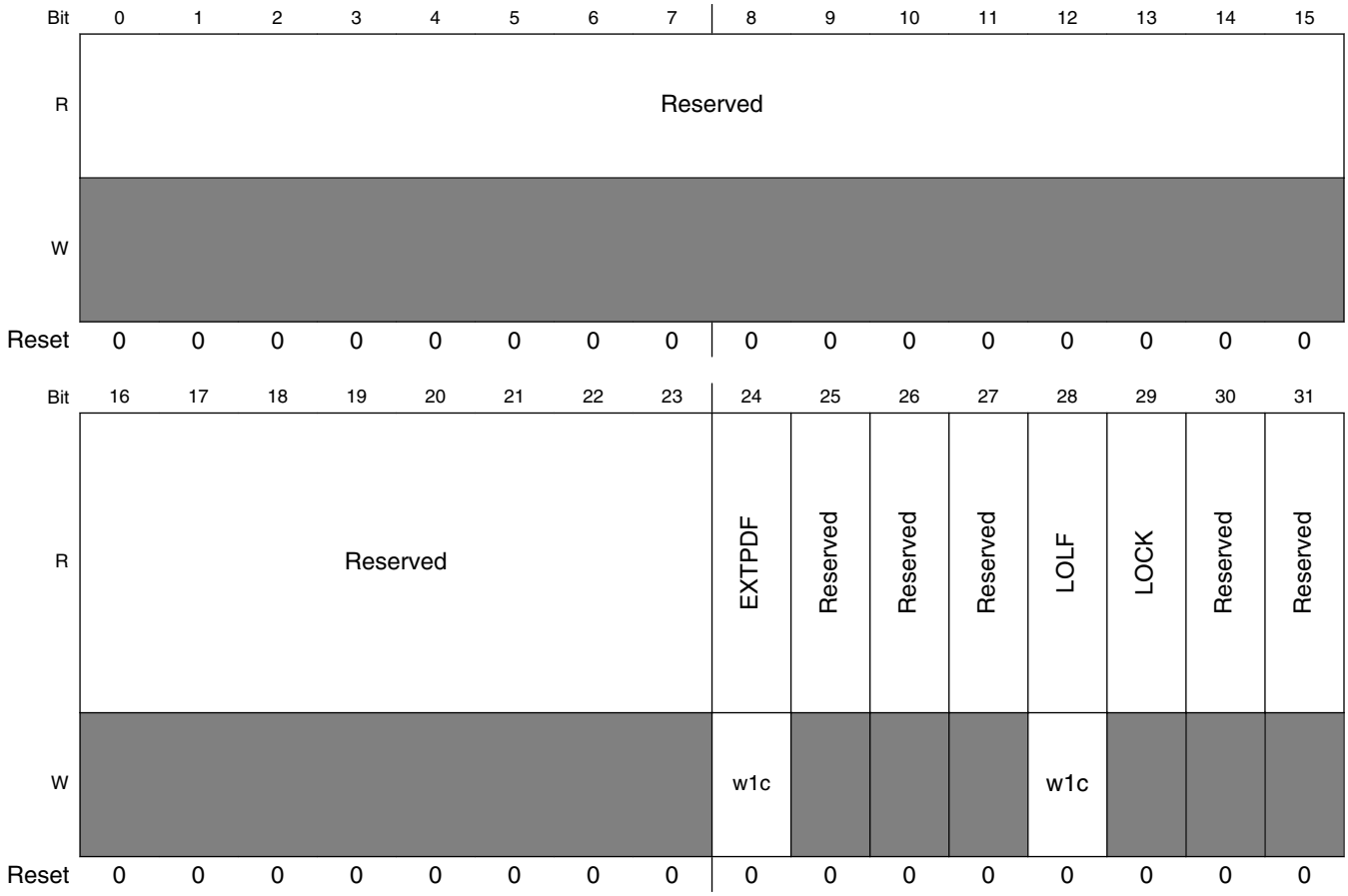
Table continues on the next page...

PLLDIG_PLL0CR field descriptions (continued)

Field	Description
	When PLLs are externally powered down the CLKCFG field changes to 00b. **NOTE:** In cascaded PLL configuration (PLL0 driving the reference clock for PLL1) the external power down signal for PLL1 must be removed only when PLL0 has locked, PLL0SR[LOCK] = 1. **NOTE:** CLKCFG can be written, but writes have no effect. Mode changes are implemented by writing to the appropriate MC_ME_< *mode* >_MC register (see the "Mode Entry Module (MC_ME)" chapter and Clock configuration for details). 00 Bypass mode with PLL off 01 Reserved 10 Reserved 11 Normal mode with PLL running.
24 EXPDIE	External Power Down Cycle Complete indication interrupt enable. This bit enables generation of an interrupt when the external power down of the PLL is complete (for example, PLLs are powered down and powered back up). 0 Ignore interrupt. Interrupt not requested 1 Enable interrupt request on power down cycle completion
25 Reserved	This field is reserved.
26 Reserved	This field is reserved.
27 Reserved	This field is reserved.
28 LOLIE	Loss-of-lock interrupt enable. The LOLIE bit enables a loss-of-lock interrupt request when PLL0SR[LOLF] = 1. If PLL0SR[LOLF] = 0 or PLL0CR[LOLIE] = 0, the interrupt request is ignored. The interrupt request only occurs in normal mode, therefore LOLIE bit has no effect in bypass mode. 0 Ignore loss-of-lock. Interrupt not requested. 1 Enable interrupt request upon loss-of-lock.
29 Reserved	This field is reserved.
30 Reserved	This field is reserved.
31 Reserved	This field is reserved.

23.5.2 PLLDIG PLL0 Status Register (PLLDIG_PLL0SR)

Address: 0h base + 4h offset = 4h



PLLDIG_PLL0SR field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24 EXTPDF	External Power Down Cycle Complete indication interrupt flag. This bit indicates that the external power down of the PLL is complete (for example, PLLs are powered down and then powered up). User must write 1 to clear this bit. 0 PLLs not power cycled. Interrupt not requested 1 PLLs power down cycle is complete. Interrupt is requested
25 Reserved	This field is reserved.
26 Reserved	This field is reserved.
27 Reserved	This field is reserved.

Table continues on the next page...

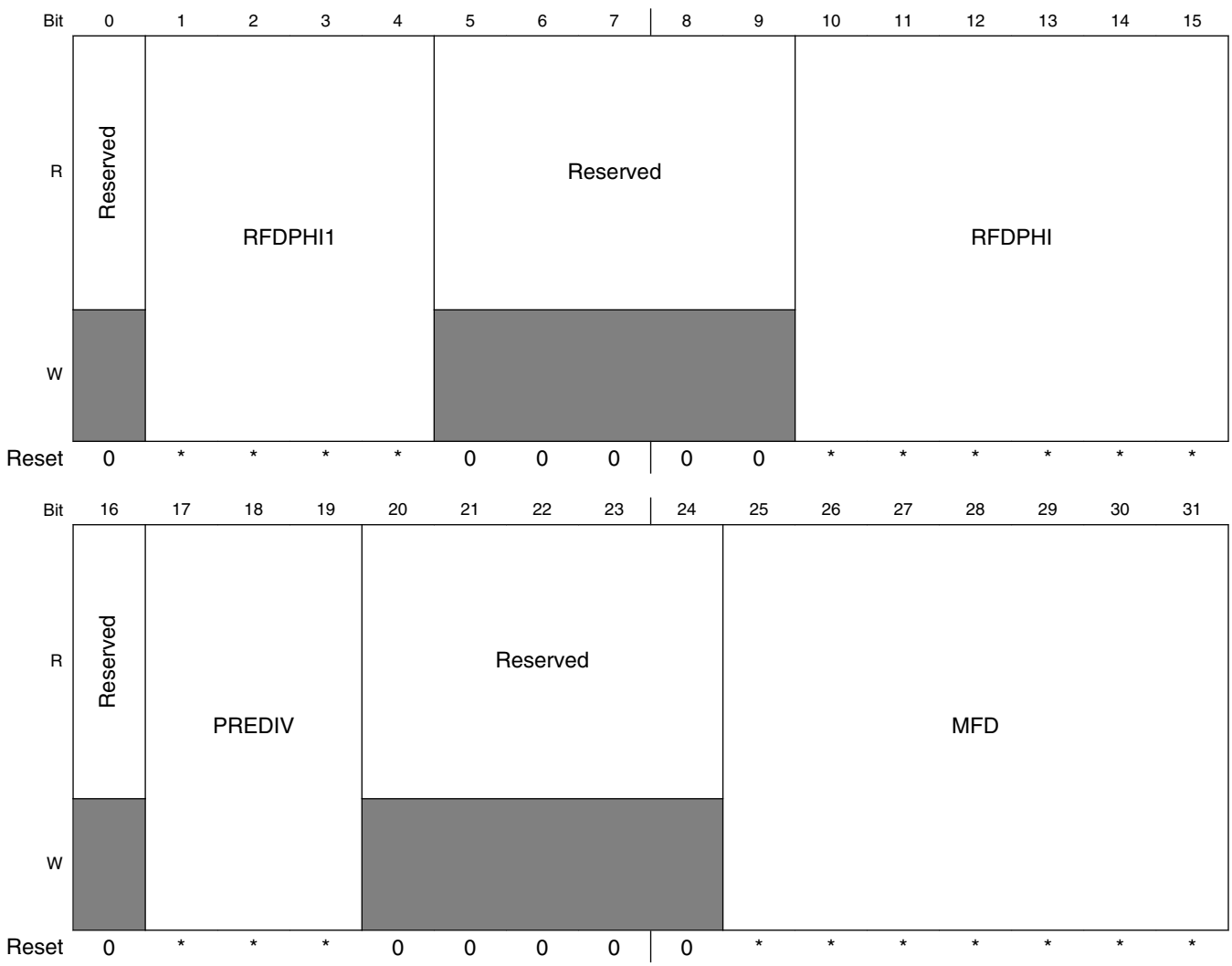
PLLDIG_PLL0SR field descriptions (continued)

Field	Description
28 LOLF	Loss-of-lock flag. This bit provides the interrupt request flag for the loss-of-lock condition. Software must write 1 to LOLF to clear it, writing 0 has no effect. This flag bit is sticky in the sense that if lock is reacquired, the bit will remain set until cleared by either writing 1 or asserting reset. LOLF is not asserted if the loss-of-lock condition was detected while the PLL is in bypass mode, but changing from normal to bypass mode will not automatically clear the LOLF if it was previously asserted while the PLL was in normal mode. 0 No loss of lock detected. Interrupt service not requested. 1 Loss of lock detected. Interrupt service requested.
29 LOCK	Lock status bit. Indicates whether PLL has acquired lock. If operating in bypass mode, the LOCK bit will still operate normally (for example, assert when the PLL acquires lock or negate when it loses lock). 0 PLL is unlocked. 1 PLL is locked.
30 Reserved	This field is reserved.
31 Reserved	This field is reserved.

23.5.3 PLLDIG PLL0 Divider Register (PLLDIG_PLL0DV)

The PLL0DV register provides the PHI/PHI1 output clock reduced frequency dividers, pre-divider, and loop divider. PLL0DV can be modified at anytime, but the changes become effective only after the PLL is disabled, then reenabled. If these fields are changed without powering down the PLL, the PLL will lose lock and generate either a reset or interrupt based on which is enabled. The reduced frequency divider bitfields (RFDPHI, RFDPHI1) can be modified at anytime, but the changes only become effective when PLL0 is disabled, then reenabled.

Address: 0h base + 8h offset = 8h



* Notes:

- MFD field: See Clocking chapter for reset value
- PREDIV field: See Clocking chapter for reset value
- RFDPHI field: See Clocking chapter for reset value

- RFDPHI1 field: See Clocking chapter for reset value

PLLDIG_PLL0DV field descriptions

Field	Description
0 Reserved	This field is reserved.
1–4 RFDPHI1	PHI1 reduced frequency divider. This 4-bit field determines the VCO clock post divider for driving the PHI1 output clock. RFDPHI1 has a range of 4..15 (4h..Fh). All other values are reserved.
5–9 Reserved	This field is reserved.
10–15 RFDPHI	PHI reduced frequency divider. This 6-bit field determines the VCO clock post divider for driving the PHI output clock. RFDPHI has a range of 1..63 (1h..3Fh). All other values are reserved.
16 Reserved	This field is reserved.
17–19 PREDIV	Input clock predivider. This field controls the value of the divider on the input clock. The output of the predivider circuit generates the reference clock to the PLL analog loop. The PREDIV value 000b causes the input clock to be inhibited. 000 Clock inhibit 001 Divide by 1 010 Divide by 2 011 Divide by 3 100 Divide by 4 101 Divide by 5 110 Divide by 6 111 Divide by 7
20–24 Reserved	This field is reserved.
25–31 MFD	Loop multiplication factor divider. This field controls the value of the divider in the feedback loop. The value specified by the MFD bits establishes the multiplication factor applied to the reference frequency. Divider value = MFD, where MFD has the range 8...127 (8h...7Fh). All other values are reserved (see Clock configuration for details).

23.5.4 PLLDIG PLL1 Control Register (PLLDIG_PLL1CR)

The PLL1CR is shown in the following table. It contains the operational state of PLL1, and is used to enable/disable PLL specific interrupts.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved					Reserved	CLKCFG		EXPDIE	Reserved	Reserved	Reserved	LOLIE	Reserved	Reserved	Reserved
W																
Reset	0	0	0	0	0	0*	0	0	0	0*	0*	0*	0*	0*	0*	0*

* Notes:
- This field can be written at any time, but writes are ignored. Read returns previously written value.
- LOLIE field: Please refer to device "Clocking" chapter for details on configuring this field.

PLLDIG_PLL1CR field descriptions

Field	Description
0–20 Reserved	This field is reserved.
21 Reserved	This field is reserved.
22–23 CLKCFG	Clock Configuration This field indicates the operational state of the PLL. When the PLLs are externally powered down the CLKCFG field changes to 00b. **NOTE:** In cascaded PLL configuration (PLL0 driving the reference clock for PLL1) the external power down signal for PLL1 must be removed only when PLL0 has locked, PLL0SR[LOCK] = 1.

Table continues on the next page...

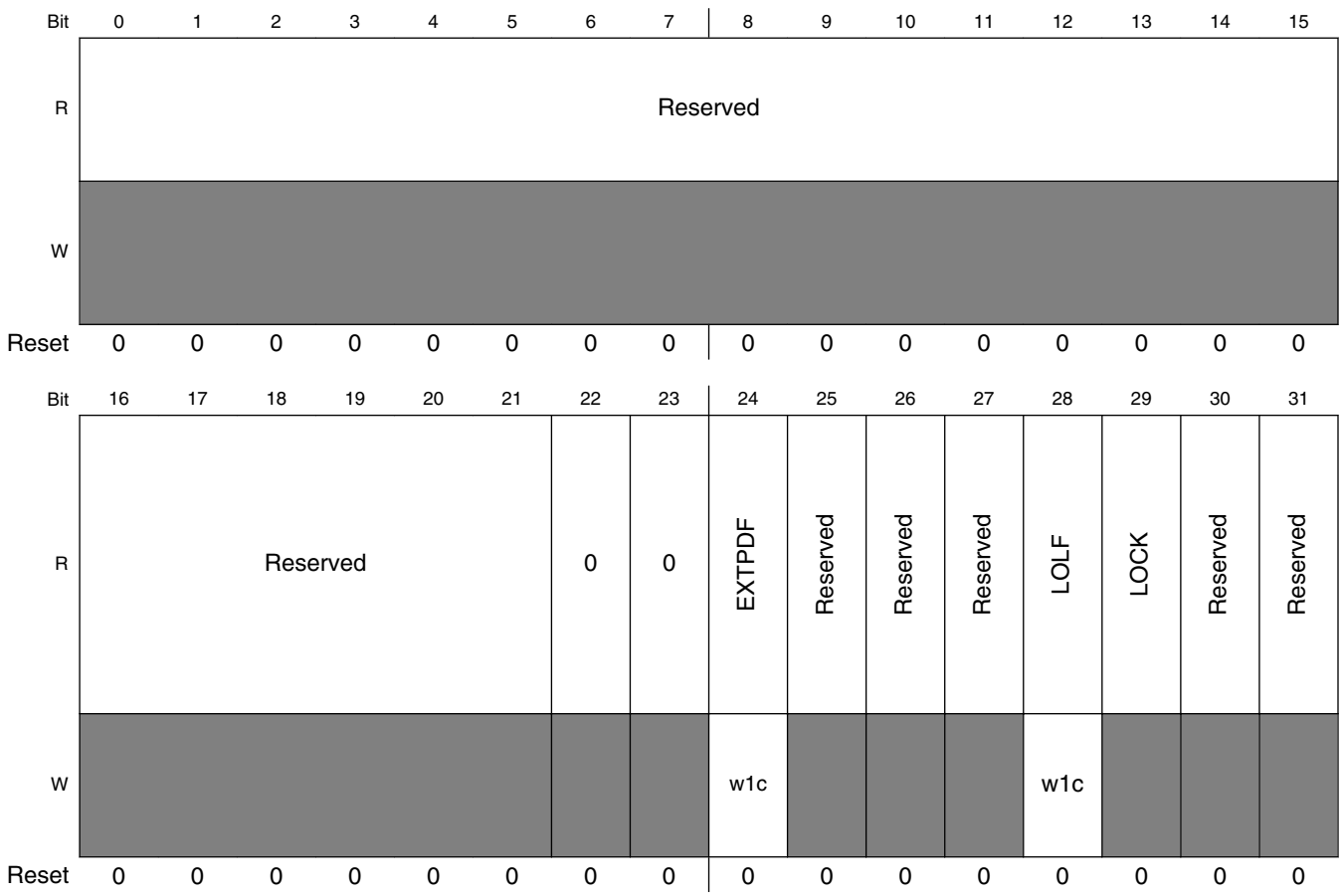
PLLDIG_PLL1CR field descriptions (continued)

Field	Description
	NOTE: CLKCFG can be written, but writes have no effect. Mode changes are implemented by writing to the appropriate MC_ME_< *mode* >_MC register (see the "Mode Entry Module (MC_ME)" chapter and Clock configuration for details). 00 Bypass mode with PLL off 01 Reserved 10 Reserved 11 Normal mode with PLL running.
24 EXPDIE	External Power Down Cycle Complete indication interrupt enable. This bit enables generation of an interrupt when external power down of the PLL is complete (for example, PLLs are powered down and powered back up). 0 Ignore interrupt. Interrupt not requested 1 Enable interrupt request on power down cycle completion
25 Reserved	This field is reserved.
26 Reserved	This field is reserved.
27 Reserved	This field is reserved.
28 LOLIE	Loss-of-lock interrupt enable. The LOLIE bit enables a loss-of-lock interrupt request when PLL1SR[LOLF] = 1. If PLL1SR[LOLF] = 0 or PLL1CR[LOLIE] = 0, the interrupt request is ignored. The interrupt request only occurs in normal mode, therefore LOLIE bit has no effect in bypass mode. 0 Ignore loss-of-lock. Interrupt not requested. 1 Enable interrupt request upon loss-of-lock.
29 Reserved	This field is reserved.
30 Reserved	This field is reserved.
31 Reserved	This field is reserved.

23.5.5 PLLDIG PLL1 Status Register (PLLDIG_PLL1SR)

PLL1SR contains status flags showing the current state of the PLL.

Address: 0h base + 24h offset = 24h



PLLDIG_PLL1SR field descriptions

Field	Description
0–21 Reserved	This field is reserved.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 EXTPDF	External Power Down Cycle Complete indication interrupt flag. This bit indicates that the external power down of the PLL is complete (for example, PLLs are powered down and then powered up). 0 PLLs not power cycled. Interrupt not requested 1 PLLs power down cycle is complete. Interrupt is requested

Table continues on the next page...

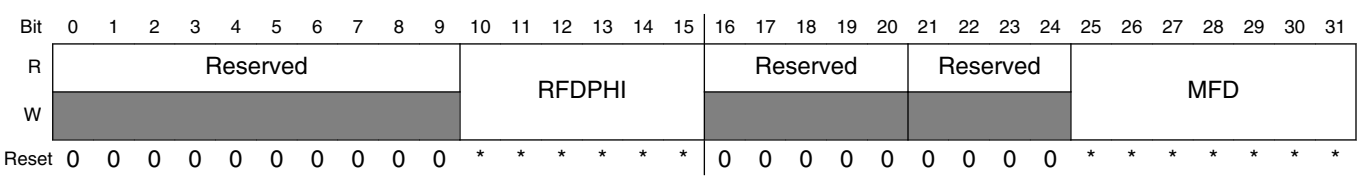
PLLDIG_PLL1SR field descriptions (continued)

Field	Description
25 Reserved	This field is reserved.
26 Reserved	This field is reserved.
27 Reserved	This field is reserved.
28 LOLF	Loss-of-lock flag. This bit provides the interrupt request flag for the loss-of-lock condition. Software must write 1 to LOLF to clear it, writing 0 has no effect. This flag bit is sticky in the sense that if lock is reacquired, the bit will remain set until cleared by either writing 1 or asserting reset. LOLF is not asserted if the loss-of-lock condition was detected while the PLL is in bypass mode, but changing from normal to bypass mode will not automatically clear the LOLF if it was previously asserted while the PLL was in normal mode. 0 No loss of lock detected. Interrupt service not requested. 1 Loss of lock detected. Interrupt service requested.
29 LOCK	Lock status bit. Indicates whether PLL has acquired lock. If operating in bypass mode, the LOCK bit will still operate normally (for example, assert when the PLL acquires lock or negate when it loses lock). 0 PLL is unlocked. 1 PLL is locked.
30 Reserved	This field is reserved.
31 Reserved	This field is reserved.

23.5.6 PLLDIG PLL1 Divider Register (PLLDIG_PLL1DV)

The values of the reduced frequency divider (RFDPHI) and loop multiplication factor divider (MFD) can be modified at anytime, but the new values only become effective after the PLL is disabled, then reenabled.

Address: 0h base + 28h offset = 28h



* Notes:
- MFD field: See Clocking chapter for reset value
- RFDPHI field: See Clocking chapter for reset value

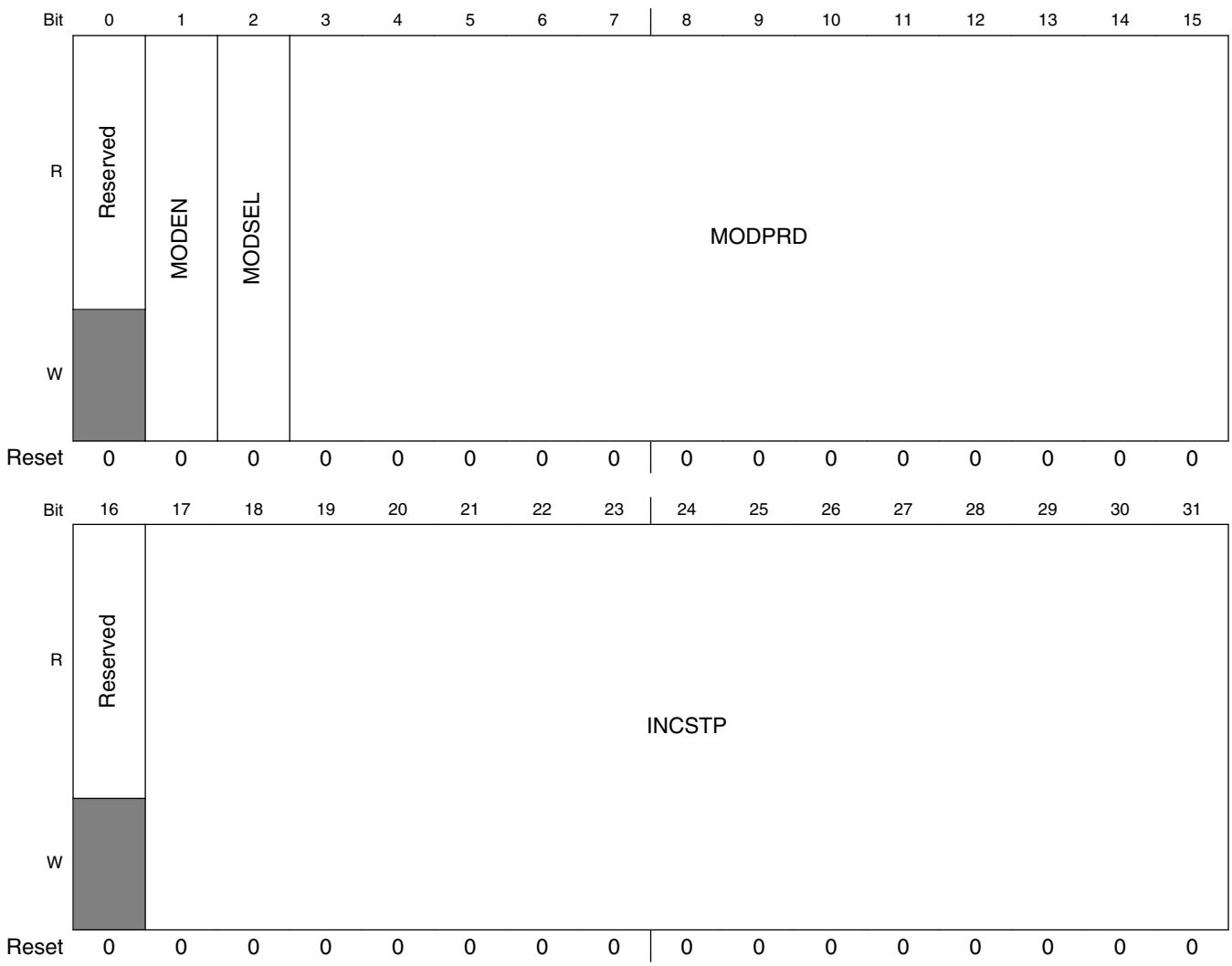
PLLDIG_PLL1DV field descriptions

Field	Description
0–9 Reserved	This field is reserved.
10–15 RFDPHI	PHI reduced frequency divider. This 6-bit field determines the VCO clock post divider for driving the PHI output clock. RFDPHI has a range of 1..63 (1h..3Fh). All other values are reserved.
16–20 Reserved	This field is reserved.
21–24 Reserved	This field is reserved.
25–31 MFD	Loop multiplication factor divider. This field controls the value of the divider in the feedback loop. The value specified by the MFD bits establishes the multiplication factor applied to the reference frequency. Divider value = MFD, where MFD has the range 16...34 (10h...22h). All other values are reserved.

23.5.7 PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)

The PLL1FM register controls enables frequency modulation, and provides controls for modulation mode selection, modulation depth and modulation rate. This register should be written when PLL1CR[CLKCFG] = 00b (PLL1 powered down). The PLL would then be required to be power cycled to regain normal functionality.

Address: 0h base + 2Ch offset = 2Ch



PLLDIG_PLL1FM field descriptions

Field	Description
0 Reserved	This field is reserved.

Table continues on the next page...

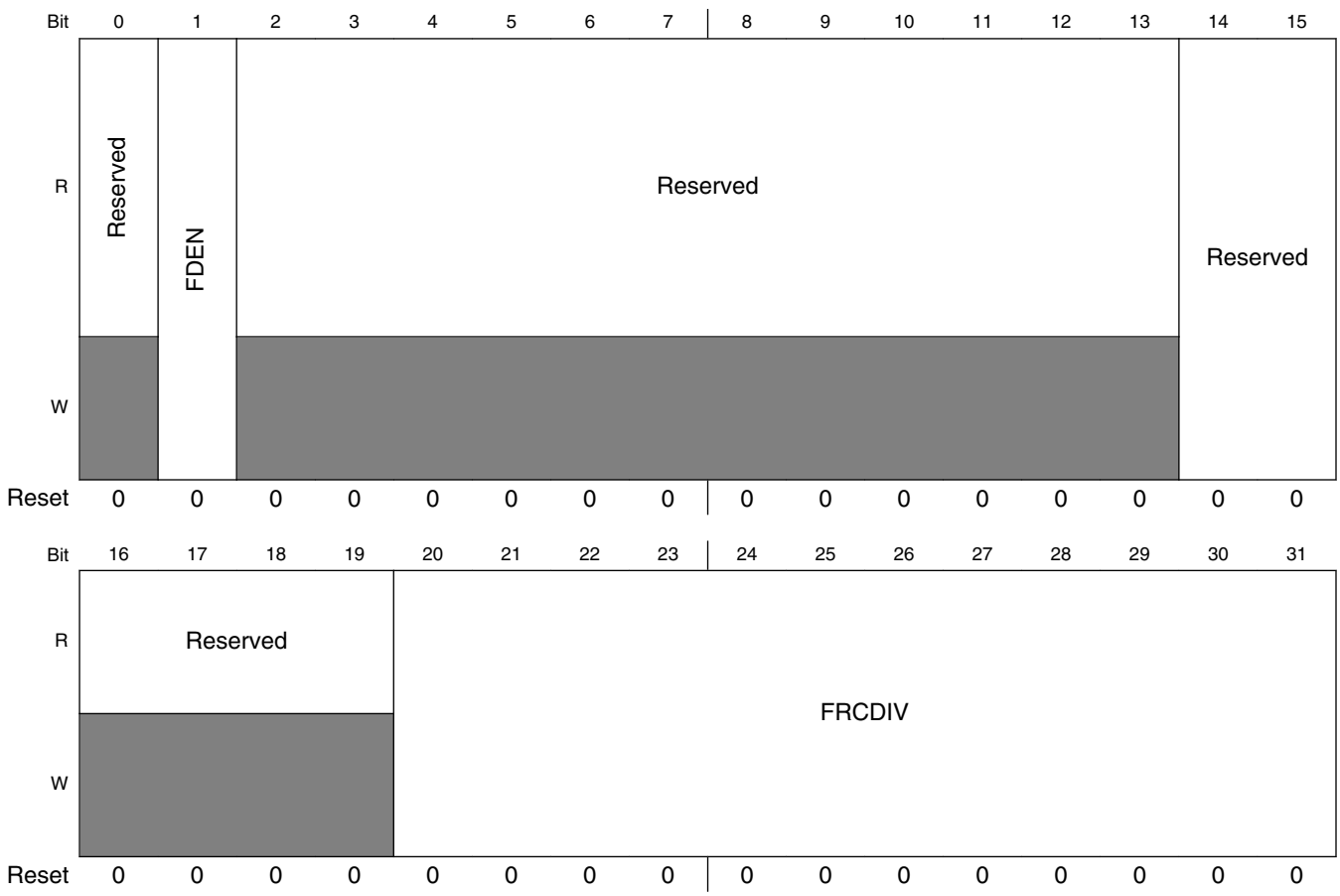
PLLDIG_PLL1FM field descriptions (continued)

Field	Description
1 MODEN	Modulation enable. This bit enables the frequency modulation. 0 Frequency modulation disabled. 1 Frequency modulation enabled.
2 MODSEL	Modulation selection. This bit selects whether modulation is centered around the nominal frequency or spread below the nominal frequency. 0 Centre Spread Modulation - Modulation centred around nominal frequency. 1 Down Spread Modulation - Modulation spread below nominal frequency.
3–15 MODPRD	Modulation period. MODPRD is the modulation period variable derived from the formulas shown in Frequency modulation.
16 Reserved	This field is reserved.
17–31 INCSTP	Increment step. This field is the INCSTP variable derived from the formulas shown in Frequency modulation.

23.5.8 PLLDIG PLL1 Fractional Divide Register (PLLDIG_PLL1FD)

The PLL1FD register provides the enable and fractional divide factor for the loop divider. This register should be written when PLL1CR[CLKCFG] = 00b (PLL1 powered down). Changing the values of FDEN once the PLL is running, and has locked, can result in the PLL losing its lock and the device being reset (PLL*n*CR[LOLRE] = 1). The PLL would then require power cycling to regain normal functionality.

Address: 0h base + 30h offset = 30h



PLLDIG_PLL1FD field descriptions

Field	Description
0 Reserved	This field is reserved.
1 FDEN	Fractional Divide Enable This bit enables the fractional divider in the loop divider for PLL1.

Table continues on the next page...

PLLDIG_PLL1FD field descriptions (continued)

Field	Description
	0 Fractional divide disabled. 1 Fractional divide enabled.
2–13 Reserved	This field is reserved.
14–15 Reserved	Reserved. This field is reserved. **Important:** This field must be written 00b.
16–19 Reserved	This field is reserved.
20–31 FRCDIV	Fractional divide input. When the fractional divide is disabled, the VCO clock frequency is the product of the input clock and the loop divide factor (MFD). When fractional divide is enabled, the mean VCO clock frequency is given by the equation in the section Clock configuration.

23.6 Register classification for safety requirements

This module is classified as ViMo (Vital Modules) for functional safety. Hence, the registers of this module have been classified as shown in the Functional Safety chapter.

23.7 Functional description

This section explains the operation and configuration of the Dual PLLDIG module.

23.7.1 Input clock frequency

PLL0 and PLL1 are designed to operate over an input clock frequency range. The operating ranges for the PLLs are discussed in detail in the *Data Sheet*.

23.7.2 Clock configuration

The relationship between input and output frequency is determined by programming the PLL0DV, PLL1DV, and PLL1FD registers, and calculated according to the following equations:

$$f_{pll0_phi} = f_{pll0_ref} \times \left(\frac{PLL0DV[MFD]}{PLL0DV[PREDIV] \times PLL0DV[RFDPHI]} \right)$$

Figure 23-9. PLL0 PHI Output Frequency

$$f_{pll0_phi1} = f_{pll0_ref} \times \left(\frac{PLL0DV[MFD]}{PLL0DV[PREDIV] \times PLL0DV[RFDPHI1]} \right)$$

Figure 23-10. PLL0 PHI1 Output Frequency

$$f_{pll1_phi} = f_{pll1_ref} \times \left(\frac{PLL1DV[MFD] + \frac{PLL1FD[FRCDIV]}{2^{12}} + \frac{1}{2^{13}}}{2 \times PLL1DV[RFDPHI]} \right)$$

Figure 23-11. PLL1 PHI Output Frequency

The relationship between the VCO frequency (f_{VCO}) and the output frequency of the PLLs is determined by the configuration of the PLL1DV, PLL1FD, and PLL0DV registers, according to the following equations:

$$f_{pll0_VCO} = \frac{f_{pll0_ref} \times PLL0DV[MFD] \times 2}{PLL0DV[PREDIV]}$$

Figure 23-12. PLL0 VCO Frequency

$$f_{pll1_VCO} = f_{pll1_ref} \times \left(PLL1DV[MFD] + \frac{PLL1FD[FRCDIV]}{2^{12}} + \frac{1}{2^{13}} \right)$$

Figure 23-13. PLL1 VCO Frequency

NOTE

f_{pll0_phi1} is the reference clock generated by PLL0 for PLL1

When programming the PLLs, user software must not violate the maximum system clock frequency or max/min VCO frequency specification of PLL0 and PLL1. Furthermore, the PLL0DV[PREDIV] value must not be set to any value that causes the input frequency to the phase detector of analog PLL blocks to go below the prescribed ranges.

The lock signal and PLL*n*SR[LOCK] flag are immediately negated if the fields of the PLL*n*DV registers are changed without powering down the analog PLLs.

When any of these events occur, an internal timer is initialized to count 64 cycles of the PLL input clock. During this period (64 cycles and a few extra clock cycles for synchronization, for example, 64 to 72 cycles), the PLL*n*SR[LOCK] flag is held negated. After the timer expires, the PLL*n*SR[LOCK] flag reflects the value coming from the PLL lock detection circuitry. To prevent an immediate reset, the PLL*n*CR[LOLRE] bit of the respective PLLs must be cleared before doing any of the above operations.

Note

The PLL must be powered down and powered up by configuring, then executing, a mode change in the MC_ME_<*mode*>_MC before PLL0DV[PREDIV], PLL0DV[MFD], PLL1DV[MFD], or the input clocks are modified.

The recommended procedure to program the PLLs and engage normal mode is shown in Initialization information.

Coming out of reset, PLL1 and PLL0 are off and in bypass mode. The recommended procedure to program the PLLs and engage normal mode is described in the 'Clocking' chapter.

23.7.3 Frequency modulation

Frequency modulation uses a triangular profile as shown in Figure 23-14. The modulation frequency and depth are controlled using PLL1FM[MODPRD] and PLL1FM[INCSTP].

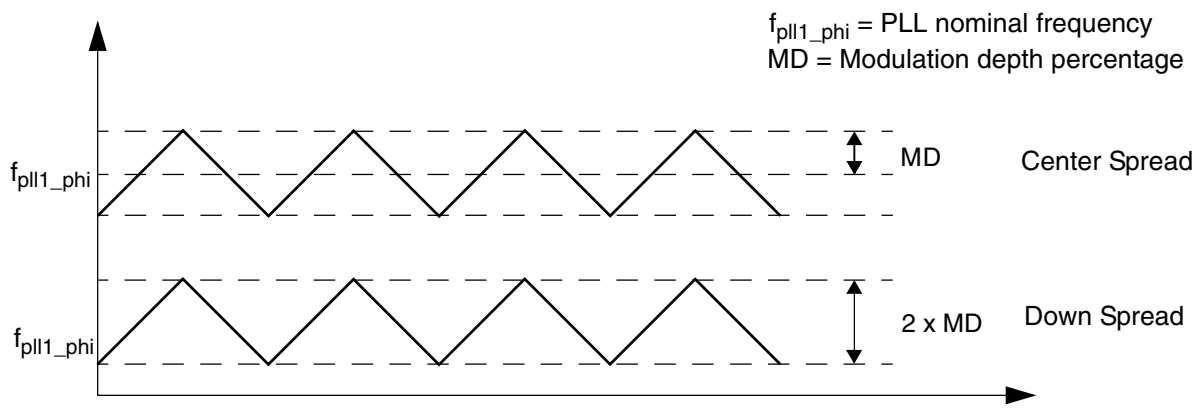


Figure 23-14. Triangular frequency modulation

NOTE

The device maximum operating frequency includes the frequency modulation. If center modulation is used, the f_{sys} must be below f_{max} by MD percentage such that $f_{max} = f_{sys} \times (1 + (MD\% / 100))$.

The following equations define how to calculate PLL1FM[MODPRD] and PLL1FM[INCSTP] based on the output frequency of the feedback divider (f_{ref}), the modulation frequency (f_{mod}) and the modulation depth percentage (MD).

$$PLL1FM[MODPRD] = round\left(\frac{f_{pll1_ref}}{4 \times f_{mod}}\right)$$

Figure 23-15. PLL1FM[MODPRD] Calculation

The equation to determine PLL1FM[INCSTP] is shown in Figure 23-16.

$$PLL1FM[INCSTP] = round\ \left(\frac{(2^{15}-1) \times MD \times PLL1DV[MFD]}{100 \times 5 \times PLL1FM[MODPRD]}\right)$$

Figure 23-16. PLL1FM[INCSTP] Calculation

PLL1FM[MODPRD] and PLL1FM[INCSTP] are subject to the following restriction:

$$(PLL1FM[MODPRD] \times PLL1FM[INCSTP]) < 2^{15}$$

Figure 23-17. PLL1FM[MODPRD] and PLL1FM[INCSTP] Restriction

Because of the above rounding operations, the effective modulation depth applied to the FMPLL is shown in Figure 23-18.

$$ModulationDepth = round\left(\frac{PLL1FM[MODPRD] \times PLLIFM[INCSTP] \times 100 \times 5}{(2^{15} - 1) \times PLL1DV[MFD]}\right)$$

Figure 23-18. Modulation Depth

FM parameters should only be changed, and FM enabled, after PLL0 has obtained lock. The sequence for reprogramming the FM is:

1. Power down PLL1 by writing MC_ME_<*mode*>_MC[PLL1ON] = 0, followed by a mode change.
2. Write a mode change to MC_ME_<*mode*>_MC[PLL1ON].

3. Program the PLL1FM while PLL1 is powered down.
4. Power up the PLL1 by writing MC_ME_<*mode*>_MC[PLL1ON] = 1, followed by a mode change.
5. Write a mode change to MC_ME_<*mode*>_MC[PLL1ON].

23.8 Maximum lock time

This describes the maximum lock time of the PLL across process, supply voltages, and junction temperature. Maximum lock time is for the condition when the operating mode of the PLL has changed from power-down to Normal mode, input control bits are stable, and supplies (analog and digital) have attained levels shown in the *Data Sheet*. It is also the time in which the PLLs regain lock after a loss of lock event, assuming all inputs are stable and supplies are within specifications.

23.9 Initialization information

Coming out of reset PLL0 and PLL1 are disabled per the DRUN mode configuration register, MC_ME_DRUN_MC. The Dual PLLDIG initialization procedure is described in the "Clocking" chapter.

NOTE

See the "Mode Entry Module (MC_ME)" chapter in this *Reference Manual* more details.

Chapter 24
Clock Monitor Unit (CMU)

24.1 Introduction

The Clock Monitor Unit (CMU), also referred to as Clock Quality Checker or Clock Fault Detector, serves three purposes:

- Measures the frequency of clock source CLKMT0_RMN with CLKMN0_RMT as the reference clock
- Monitors CLKMN0_RMT frequency with CLKMT0_RMN as reference clock
- Monitors CLKMN1 frequency with CLKMT0_RMN as reference clock and detects if the monitored clock frequency leaves an upper or lower frequency boundary

NOTE

See the "Clocking" chapter for chip-specific sources used by the CMU.

One of the tasks is to supervise the integrity of the various clock sources on the chip, for example CLKMN0_RMT or CLKMN1. If the monitored clock frequency is less than the reference clock, or it violates an upper or lower frequency boundary, the CMU detects and reports this event. These events signal the FCCU, which can take the necessary corrective actions as its configuration dictates.

The CMU can monitor CLKMN0_RMT, which must have a frequency higher than that of CLKMT0_RMN divided by the factor shown in CMU_CSR[RCDIV], and reports this event. The CMU can also monitor CLKMN1 and generate an event if CLKMN1 is greater than a high frequency boundary or less than a low frequency boundary. The upper and lower frequency boundaries are defined by the CMU High Frequency Reference Register (CMU_HFREFR) and CMU Low Frequency Reference Register (CMU_LFREFR).

The second task of the CMU is to provide a frequency meter, which allows measuring the frequency of a clock source against a reference clock. This is useful to allow the calibration of the metered clocks (such as CLKMT0_RMN), as well as to be able to correct/calculate the time deviation of a counter that is clocked by the metered clocks.

NOTE

See the "Clocking" chapter for the number of CMU instances on this chip.

24.1.1 Main features

- CLKMT0_RMN frequency measurement with CLKMN0_RMT as reference clock.
- CLKMN0_RMT monitoring with respect to CLKMT0_RMN ÷ $2^{CSR[RCDIV]}$ clock.
- Upper or lower frequency boundary monitoring of CLKMN1 with respect to CLKMT0_RMN ÷ 4.
- Event generation for various failures detected inside monitoring unit.

24.2 Block diagram

The block diagram of the CMU module(s) is shown in the “Clocking" chapter of this Reference Manual.

24.3 Signals

The table below describes the signals on the boundary of the CMU (in alphabetical order).

Table 24-1. Signal description

Signal	I/O	Description
CLKMN0_RMT	I	Monitored Clock Signal 0/Metered Clock Signal Reference — Receives a clock signal that the CMU compares to a specified low-limit frequency to determine whether the frequency of the clock signal is greater than the specified limit. Also provides a reference clock signal for all metered clock signals.
CLKMN1	I	Monitored Clock Signal 1 — Receives a clock signal that the CMU compares to specified low-limit and high-limit frequencies to determine whether the frequency of the clock signal is between the specified limits.
CLKMT0_RMN	I	Metered Clock Signal 0/Monitored Clock Signal Reference — Receives a clock signal that the CMU measures against a reference clock frequency. Also provides a reference clock signal for all monitored clock signals.

NOTE

See the "Clocking" chapter for device specific clock sources of each CMU.

24.4 Register description and memory map

This section describes in address order all the CMU registers. Each description includes a standard register diagram with an associated figure number. The CMU memory map is listed in the following table.

NOTE

See "Clocking" chapter for register and field availability details.

CMU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	CMU Control Status Register (CMU_CSR)	32	R/W	See section	24.4.1/690
4	CMU Frequency Display Register (CMU_FDR)	32	R	0000_0000h	24.4.2/691
8	CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)	32	R/W	0000_0FFFh	24.4.3/692
C	CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)	32	R/W	0000_0000h	24.4.4/692
10	CMU Interrupt Status Register (CMU_ISR)	32	w1c	See section	24.4.5/693
18	CMU Measurement Duration Register (CMU_MDR)	32	R/W	0000_0000h	24.4.6/695

24.4.1 CMU Control Status Register (CMU_CSR)

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								SFM	0						
W																
Reset	0	0	0	0	0	0	0	0	0*	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						CKSEL1		0					RCDIV		CME
W																
Reset	0	0	0	0	0	0	0*	0*	0	0	0	0	0	0*	0*	0

* Notes:
- RCDIV field: Not all CMU blocks will utilize this feature. See the “Clocking” chapter for CMU implementation details.
- CKSEL1 field: Not all CMU blocks will utilize this feature. See the “Clocking” chapter for CMU implementation details.
- SFM field: Not all CMU blocks will utilize this feature. See the “Clocking” chapter for CMU implementation details.

CMU_CSR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 SFM	Start frequency measure. The software can only set this bit to start a clock frequency measure. It is reset by hardware when the measure is ready in the CMU_FDR. 0 Frequency measurement is completed or not yet started 1 Frequency measurement is not completed
9–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 CKSEL1	Frequency measure clock selection bit. CKSEL1 selects the clock to be measured by the frequency meter. This only affects CMU instances that utilizes clock metering. Not all CMU blocks will utilize this feature. See the “Clocking” chapter for device specific CMU implementation details. 00 CLKMT0_RMN is selected 01 Reserved 10 Reserved 11 CLKMT0_RMN is selected
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 RCDIV	CLKMT0_RMN division factor.

Table continues on the next page...

CMU_CSR field descriptions (continued)

Field	Description
	These bits specify the CLKMT0_RMN division factor. The output clock frequency is $f_{CLKMT0_RMN} \div 2^{CMU_CSR[RCDIV]}$. This output clock is used as reference clock to compare with CLKMN0_RMT for crystal clock monitor feature. 00 CLKMT0_RMN ÷ 1 (No division) 01 CLKMT0_RMN ÷ 2 10 CLKMT0_RMN ÷ 4 11 CLKMT0_RMN ÷ 8
31 CME	CLKMN1 monitor enable. 0 CLKMN1 monitor is disabled 1 CLKMN1 monitor is enabled

24.4.2 CMU Frequency Display Register (CMU_FDR)

The CMU_FDR is used to determine the measured frequency of:

- CLKMT0_RMN

with respect to the reference clock CLKMN0_RMT.

NOTE

The CMU_FDR should be read only when CMU_CSR[SFM] = 0.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												FD																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

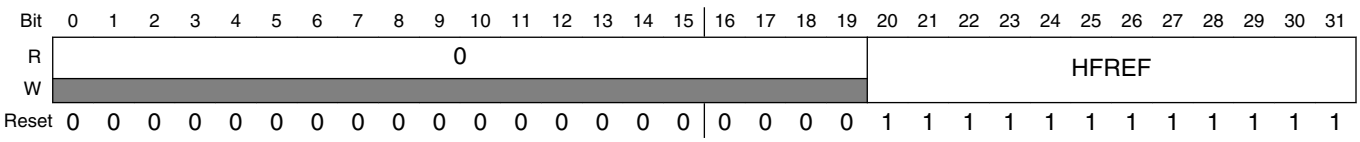
CMU_FDR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 FD	Measured frequency bits. This register displays the measured frequency (f_{sel}) with respect to the reference clock (f_{CLKMN0_RMT}). The measured value is given by the following formula: $f_{sel} = (f_{CLKMN0_RMT} \times CMU_MDR[MD]) \div CMU_FDR[FD]$

24.4.3 CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)

The HFREFR is configured for the high frequency reference that the CMU will use for comparing against the monitored clock. The figure and table below show the CMU_HFREFR register.

Address: 0h base + 8h offset = 8h



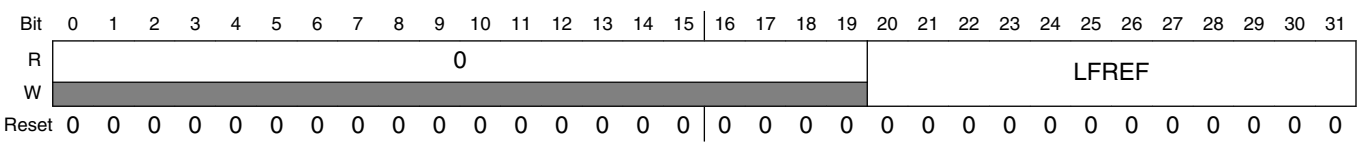
CMU_HFREFR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 HFREF	High Frequency reference value. These bits determine the high reference value for the CLKMN1 frequency. The reference value is given by: (HFREF ÷ 16) × (f_{CLKMT0_RMN} ÷ 4).

24.4.4 CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)

The LFREFR is configured for the low frequency reference that the CMU will use for comparing against the monitored clock. The figure and table below show the CMU_LFREFR register.

Address: 0h base + Ch offset = Ch



CMU_LFREFR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

CMU_LFREFR field descriptions (continued)

Field	Description
20–31 LFREF	Low Frequency reference value. These bits determine the low reference value for the CLKMN1 frequency. The reference value is given by: (LFREF ÷ 16) × (f_{CLKMT0_RMN} ÷ 4).

24.4.5 CMU Interrupt Status Register (CMU_ISR)

NOTE

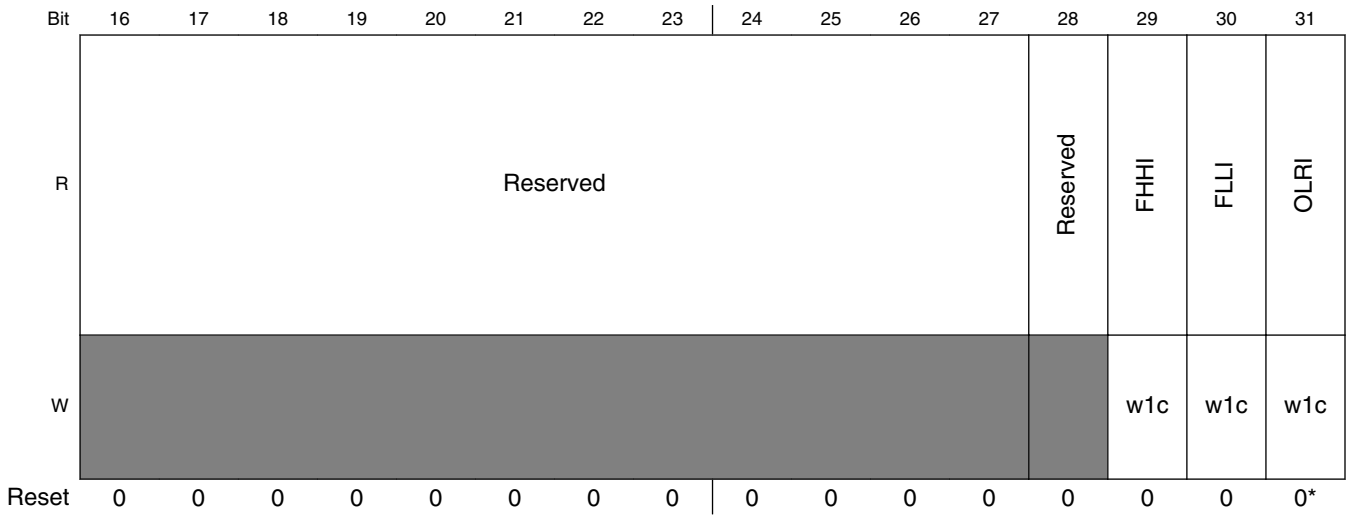
All flags in the CMU_ISR are set asynchronously. This register must be read only after an interrupt(safe mode entry) is triggered by the CMU. Otherwise, a read access on this register may fetch an incorrect value (see "Clocking" chapter for interrupt operation) .

NOTE

Before entering low-power stop modes, all clock frequency measurements in the CMU should be disabled. Failing to do so may result in spurious interrupts.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



* Notes:
- OLRI field: Not all CMU blocks will utilize this feature. See the "Clocking" chapter for specific CMU implementation details.

CMU_ISR field descriptions

Field	Description
0–27 Reserved	This field is reserved.
28 Reserved	This field is reserved.
29 FHHI	CLKMN1 frequency higher than high reference event status. This bit is set by hardware when CLKMN1 frequency becomes higher than the high frequency reference value determined by CMU_HFREFR[HFREF] and CLKMN1 is 'ON' as signaled by the MC_ME. It can be cleared by software by writing '1'. 0 No FHH event 1 FHH event occurred
30 FLLI	CLKMN1 frequency less than low reference event status. This bit is set by hardware when CLKMN1 frequency becomes lower than the low frequency reference value determined by CMU_LFREFR[LFREF], and CLKMN1 is 'ON' as signaled by the MC_ME. Software clears this field by writing a '1'. 0 No FLL event 1 FLL event occurred
31 OLRI	Oscillator frequency less than $f_{CLKMT0_RMN} \div 2^{CMU_CSR[RCDIV]}$event status. This bit is set by hardware when the f_{CLKMN0_RMT} is less than $f_{CLKMT0_RMN} \div 2^{CMU_CSR[RCDIV]}$frequency and CLKMN0_RMT is 'ON' as signaled by the MC_ME. It can be cleared by software by writing '1'. **NOTE:** While entering STOP mode, OLRI may be triggered. To avoid this when the system is running on the PLL, disable the XOSC before entering STOP mode. 0 No OLR event 1 OLR event occurred

24.4.6 CMU Measurement Duration Register (CMU_MDR)

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												MD																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CMU_MDR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 MD	Measurement duration bits This field displays the measurement duration in terms of selected clock (CLKMT0_RMN) cycles. This value is loaded in the frequency meter down-counter. The down-counter starts counting when CMU_CSR [SFM] = 1.

24.5 Functional description

24.5.1 Frequency meter

The purpose of the frequency meter is to evaluate the deviation from the nominal metered source (such as CLKMT0_RMN) frequencies. This in turn allows either recalibration of these clocks or other timing corrections. Programming of the CMU_CSR[CKSEL1] field is used to select one of the metered clocks from a multiplexer that drives a simple Frequency Meter (see the CMU Block Diagram in the "Clocking" chapter). The reference clock for the Frequency Meter is the CLKMN0_RMT signal. The measurement starts when CMU_CSR[SFM] = 1. The measurement duration is given by the contents of CMU_MDR[MD] in terms of number of clock cycles of the selected metered clock. The CMU_CSR[SFM] bit is cleared by hardware once the frequency measurement is complete and the count is loaded in CMU_FDR[FD]. The frequency of the selected clock (f_{sel}) can be derived from the value loaded in the CMU_FDR[FD] as shown in the following equation:

$$f_{sel} = f_{CLKMN0_RMT} \times \frac{CMU_MDR[MD]}{CMU_FDR[FD]}$$

24.5.2 CLKMN0_RMT supervisor

If frequency of CLKMN0_RMT is smaller than the frequency of CLKMT0_RMN ÷ $2^{CMU_CSR[RCDIV]}$ and CLKMN0_RMT is 'ON' as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[OLRI].
- The CMU asserts the OLR signal.

Note

f_{CLKMN0_RMT} must be greater than $f_{CLKMT0_RMN} \div 2^{CMU_CSR[RCDIV]}$ by at least 0.5 MHz in order to guarantee correct f_{CLKMN0_RMT} monitoring.

24.5.3 CLKMN1 supervisor

The frequency of CLKMN1(f_{CLKMN1}) can be monitored by programming CMU_CSR[CME] = 1. CLKMN1 monitoring starts as soon as CMU_CSR[CME] = 1. This monitor can be disabled at any time by programming CMU_CSR[CME] = 0.

If f_{CLKMN1} is greater than the reference value determined by fields CMU_HFREFR[HFREF] and CLKMN1 is 'ON' as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[FHHI].
- The CMU asserts the FHH signal.

If f_{CLKMN1} is less than a reference value determined by the bits CMU_LFREFR[LFREF] and the CLKMN1 is 'ON' as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[FLLI].
- The CMU asserts the FLL signal.

Note

An example of determining the $HFREF_{Actual}$ is as follows. Assume a f_{CLKMT0_RMN} = 16 MHz with a accuracy of +/-5%. In order to monitor a f_{CLKMN1} = 200 MHz, the ideal $HFREF_{Ideal}$ = 800. The actual HFREF value will be 842 when the accuracy is taken into consideration ($HFREF_{Actual}$ = ($HFREF_{Ideal}$ ÷ 0.95).

The actual LFREF value will be 762 when accuracy is taken into consideration ($LFREF_{Actual} = LFREF_{Ideal} \div 1.05$).

Chapter 25
Clock Generation Module (MC_CGM)

25.1 Introduction

This chapter describes the Clock Generation Module (MC_CGM).

25.1.1 Overview

The clock generation module (MC_CGM) generates reference clocks for all the chip blocks. The MC_CGM selects one of the system clock sources to supply the system clock. The MC_ME module controls the system clock selection (see the MC_ME chapter for more details). Peripheral clock selection is controlled by MC_CGM control registers. A set of MC_CGM registers controls the clock dividers which are used for divided system and peripheral clock generation. The memory spaces of system and peripheral clock sources which have addressable memory spaces can be accessed through the MC_CGM memory space.

The following figure shows the MC_CGM block diagram.

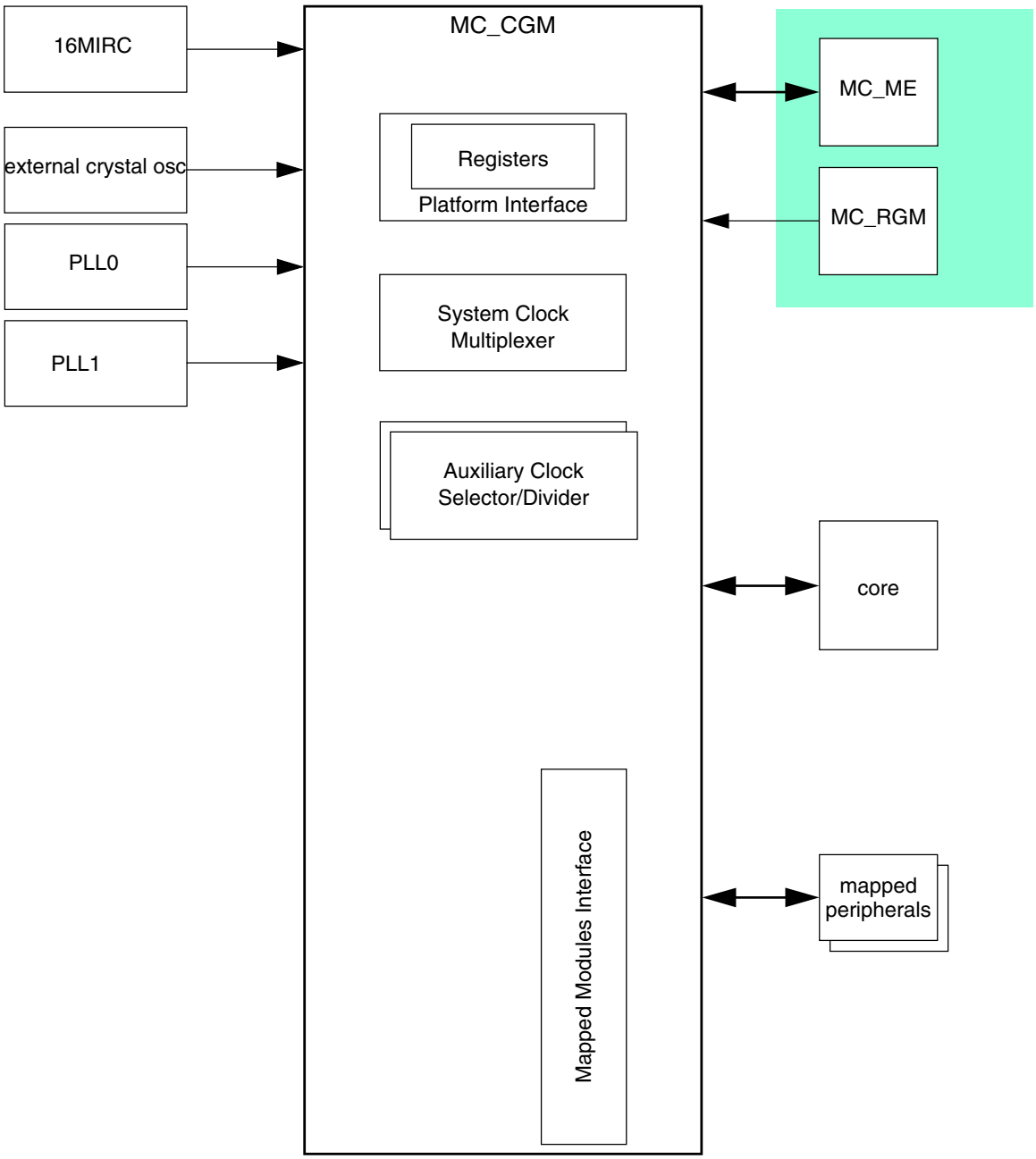


Figure 25-1. MC_CGM Block Diagram

25.1.2 Features

The MC_CGM includes the following features:

- generates system and peripheral clocks
- selects and enables/disables the system clock supply from system clock sources according to MC_ME control

- performs progressive system clock frequency change depending on MC_ME mode configuration
- contains a set of registers to control clock dividers for divided clock generation
- contains a set of registers to control peripheral clock selection
- supports multiple clock sources and maps their address spaces to its memory map
- contains IPS interface for mapped clock source modules
- supports PLL control by PLL-based-scan controller
- supports 8, 16, and 32-bit wide read/write accesses

25.2 External Signal Description

The MC_CGM delivers output clocks to the clkout pins for off-chip use and/or observation.

25.3 MC_CGM Registers

The following subsections show the MC_CGM registers memory map and the detailed definition of each register.

MC_CGM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	PCS Switch Duration (MC_CGM_PCS_SDUR)	8	R/W	00h	25.3.1/704
4	PCS Divider Change for external crystal oscillator (MC_CGM_PCS_DIVC1)	32	R/W	03E7_0000h	25.3.2/704
8	PCS Divider End for external crystal oscillator (MC_CGM_PCS_DIVE1)	32	R/W	0000_03E7h	25.3.3/705
C	PCS Divider Start for external crystal oscillator (MC_CGM_PCS_DIVS1)	32	R/W	0000_03E7h	25.3.4/705
10	PCS Divider Change for PLL0 PHI (MC_CGM_PCS_DIVC2)	32	R/W	03E7_0000h	25.3.5/706
14	PCS Divider End for PLL0 PHI (MC_CGM_PCS_DIVE2)	32	R/W	0000_03E7h	25.3.6/707
18	PCS Divider Start for PLL0 PHI (MC_CGM_PCS_DIVS2)	32	R/W	0000_03E7h	25.3.7/707
28	PCS Divider Change for PLL1 PHI (MC_CGM_PCS_DIVC4)	32	R/W	03E7_0000h	25.3.8/708
2C	PCS Divider End for PLL1 PHI (MC_CGM_PCS_DIVE4)	32	R/W	0000_03E7h	25.3.9/708
30	PCS Divider Start for PLL1 PHI (MC_CGM_PCS_DIVS4)	32	R/W	0000_03E7h	25.3.10/ 709

Table continues on the next page...

MC_CGM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D0	System Clock Divider Ratio Change Register (MC_CGM_SC_DIV_RC)	32	R/W	0000_0001h	25.3.11/ 710
D4	Divider Update Type (MC_CGM_DIV_UPD_TYPE)	32	R/W	0000_0000h	25.3.12/ 711
D8	Divider Update Trigger (MC_CGM_DIV_UPD_TRIG)	32	R/W	0000_0000h	25.3.13/ 712
DC	Divider Update Status (MC_CGM_DIV_UPD_STAT)	32	R/W	0000_0000h	25.3.14/ 713
E4	System Clock Select Status (MC_CGM_SC_SS)	32	R	0008_0000h	25.3.15/ 715
E8	System Clock Divider Configuration 0 (MC_CGM_SC_DC0)	32	R/W	8000_0000h	25.3.16/ 716
EC	System Clock Divider Configuration 1 (MC_CGM_SC_DC1)	32	R/W	8000_0000h	25.3.17/ 717
F0	System Clock Divider Configuration 2 (MC_CGM_SC_DC2)	32	R/W	8000_0000h	25.3.18/ 717
100	Aux Clock 0 Select Control (MC_CGM_AC0_SC)	32	R/W	0000_0000h	25.3.19/ 718
104	Aux Clock 0 Select Status (MC_CGM_AC0_SS)	32	R	0000_0000h	25.3.20/ 719
108	Aux Clock 0 Divider Configuration 0 (MC_CGM_AC0_DC0)	32	R/W	8000_0000h	25.3.21/ 720
10C	Aux Clock 0 Divider Configuration 1 (MC_CGM_AC0_DC1)	32	R/W	0000_0000h	25.3.22/ 721
110	Aux Clock 0 Divider Configuration 2 (MC_CGM_AC0_DC2)	32	R/W	0000_0000h	25.3.23/ 721
114	Aux Clock 0 Divider Configuration 3 (MC_CGM_AC0_DC3)	32	R/W	0000_0000h	25.3.24/ 722
118	Aux Clock 0 Divider Configuration 4 (MC_CGM_AC0_DC4)	32	R/W	0000_0000h	25.3.25/ 723
120	Aux Clock 1 Select Control (MC_CGM_AC1_SC)	32	R/W	0000_0000h	25.3.26/ 724
124	Aux Clock 1 Select Status (MC_CGM_AC1_SS)	32	R	0000_0000h	25.3.27/ 725
128	Aux Clock 1 Divider Configuration 0 (MC_CGM_AC1_DC0)	32	R/W	0000_0000h	25.3.28/ 726
140	Aux Clock 2 Select Control (MC_CGM_AC2_SC)	32	R/W	0000_0000h	25.3.29/ 726
144	Aux Clock 2 Select Status (MC_CGM_AC2_SS)	32	R	0000_0000h	25.3.30/ 727
148	Aux Clock 2 Divider Configuration 0 (MC_CGM_AC2_DC0)	32	R/W	0000_0000h	25.3.31/ 728
160	Aux Clock 3 Select Control (MC_CGM_AC3_SC)	32	R/W	0000_0000h	25.3.32/ 729

Table continues on the next page...

MC_CGM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
164	Aux Clock 3 Select Status (MC_CGM_AC3_SS)	32	R	0000_0000h	25.3.33/ 730
180	Aux Clock 4 Select Control (MC_CGM_AC4_SC)	32	R/W	0000_0000h	25.3.34/ 731
184	Aux Clock 4 Select Status (MC_CGM_AC4_SS)	32	R	0000_0000h	25.3.35/ 732
1A0	Aux Clock 5 Select Control (MC_CGM_AC5_SC)	32	R/W	0000_0000h	25.3.36/ 732
1A4	Aux Clock 5 Select Status (MC_CGM_AC5_SS)	32	R	0000_0000h	25.3.37/ 733
1A8	Aux Clock 5 Divider Configuration 0 (MC_CGM_AC5_DC0)	32	R/W	8000_0000h	25.3.38/ 734
1AC	Aux Clock 5 Divider Configuration 1 (MC_CGM_AC5_DC1)	32	R/W	0000_0000h	25.3.39/ 735
1C0	Aux Clock 6 Select Control (MC_CGM_AC6_SC)	32	R/W	0000_0000h	25.3.40/ 736
1C4	Aux Clock 6 Select Status (MC_CGM_AC6_SS)	32	R	0000_0000h	25.3.41/ 737
1C8	Aux Clock 6 Divider Configuration 0 (MC_CGM_AC6_DC0)	32	R/W	0000_0000h	25.3.42/ 737
200	Aux Clock 8 Select Control (MC_CGM_AC8_SC)	32	R/W	0000_0000h	25.3.43/ 738
204	Aux Clock 8 Select Status (MC_CGM_AC8_SS)	32	R	0000_0000h	25.3.44/ 739
208	Aux Clock 8 Divider Configuration 0 (MC_CGM_AC8_DC0)	32	R/W	0000_0000h	25.3.45/ 740
220	Aux Clock 9 Select Control (MC_CGM_AC9_SC)	32	R/W	0000_0000h	25.3.46/ 741
224	Aux Clock 9 Select Status (MC_CGM_AC9_SS)	32	R	0000_0000h	25.3.47/ 742
228	Aux Clock 9 Divider Configuration 0 (MC_CGM_AC9_DC0)	32	R/W	0000_0000h	25.3.48/ 743
240	Aux Clock 10 Select Control (MC_CGM_AC10_SC)	32	R/W	0000_0000h	25.3.49/ 743
244	Aux Clock 10 Select Status (MC_CGM_AC10_SS)	32	R	0000_0000h	25.3.50/ 744
248	Aux Clock 10 Divider Configuration 0 (MC_CGM_AC10_DC0)	32	R/W	0000_0000h	25.3.51/ 745

25.3.1 PCS Switch Duration (MC_CGM_PCS_SDUR)

This register contains the progressive system clock switching duration for each step. See Section "Progressive System Clock Switching" for details on how to set this value.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7
Read Write	SDUR							
Reset	0	0	0	0	0	0	0	0

MC_CGM_PCS_SDUR field descriptions

Field	Description
0–7 SDUR	Switch Duration - This value defines the duration of one PCS clock switch step in terms of 16 MHz internal RC osc. cycles.

25.3.2 PCS Divider Change for external crystal oscillator (MC_CGM_PCS_DIVC1)

These registers define the rate of frequency change and initial change value on frequency ramp-up for the progressive system clock switching when switching to or from the external crystal oscillator. See Section "Progressive System Clock Switching" for details on how to set these values.

NOTE

Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INIT																0								RATE							
W																																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_PCS_DIVC1 field descriptions

Field	Description
0–15 INIT	Divider Change Initial Value - This is initial change value of the clock divider for the clock ramp-up phase when switching to external crystal oscillator.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RATE	Divider Change Rate - This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching from the external crystal oscillator.

25.3.3 PCS Divider End for external crystal oscillator (MC_CGM_PCS_DIVE1)

These registers define the final division value on frequency ramp-down for the progressive system clock switching when switching from or to external crystal oscillator on ramp up and ramp down. See Section "Progressive System Clock Switching" for details on how to set these values.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DIVE																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

MC_CGM_PCS_DIVE1 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVE	Divider End Value - This is the clock divider end value for the clock ramp-down phase when switching from the external crystal oscillator.

25.3.4 PCS Divider Start for external crystal oscillator (MC_CGM_PCS_DIVS1)

These registers define the initial division value on frequency ramp-up for the progressive system clock switching when switching the system clock from or to the external crystal oscillator on ramp up and ramp down. See Section "Progressive System Clock Switching" for details on how to set these values.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DIVS																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

MC_CGM_PCS_DIVS1 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVS	Divider Start Value - This is the start value of the clock divider for the clock ramp-up phase when switching to the external crystal oscillator.

25.3.5 PCS Divider Change for PLL0 PHI (MC_CGM_PCS_DIVC2)

These registers define the rate of frequency change and initial change value on frequency ramp-up for the progressive system clock switching when switching to or from the PLL0 PHI. See Section "Progressive System Clock Switching" for details on how to set these values.

NOTE

Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INIT																0								RATE							
W																																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_PCS_DIVC2 field descriptions

Field	Description
0–15 INIT	Divider Change Initial Value - This is initial change value of the clock divider for the clock ramp-up phase when switching to PLL0 PHI.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RATE	Divider Change Rate - This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching from the PLL0 PHI.

25.3.6 PCS Divider End for PLL0 PHI (MC_CGM_PCS_DIVE2)

These registers define the final division value on frequency ramp-down for the progressive system clock switching when switching from or to PLL0 PHI on ramp up and ramp down. See Section "Progressive System Clock Switching" for details on how to set these values.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DIVE																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

MC_CGM_PCS_DIVE2 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVE	Divider End Value - This is the clock divider end value for the clock ramp-down phase when switching from the PLL0 PHI.

25.3.7 PCS Divider Start for PLL0 PHI (MC_CGM_PCS_DIVS2)

These registers define the initial division value on frequency ramp-up for the progressive system clock switching when switching the system clock from or to the PLL0 PHI on ramp up and ramp down. See Section "Progressive System Clock Switching" for details on how to set these values.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DIVS																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

MC_CGM_PCS_DIVS2 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVS	Divider Start Value - This is the start value of the clock divider for the clock ramp-up phase when switching to the PLL0 PHI.

25.3.8 PCS Divider Change for PLL1 PHI (MC_CGM_PCS_DIVC4)

These registers define the rate of frequency change and initial change value on frequency ramp-up for the progressive system clock switching when switching to or from the PLL1 PHI. See Section "Progressive System Clock Switching" for details on how to set these values.

NOTE

Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INIT																0								RATE							
W																																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_PCS_DIVC4 field descriptions

Field	Description
0–15 INIT	Divider Change Initial Value - This is initial change value of the clock divider for the clock ramp-up phase when switching to PLL1 PHI.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RATE	Divider Change Rate - This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching from the PLL1 PHI.

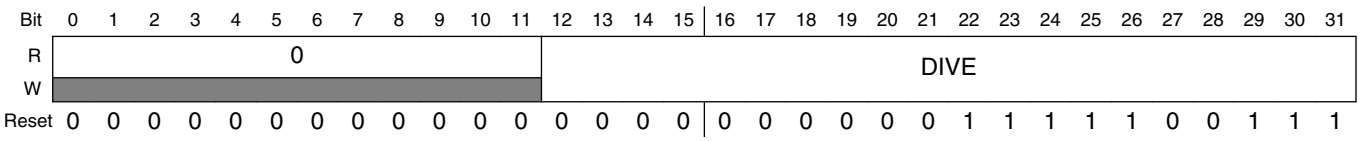
25.3.9 PCS Divider End for PLL1 PHI (MC_CGM_PCS_DIVE4)

These registers define the final division value on frequency ramp-down for the progressive system clock switching when switching from or to PLL1 PHI on ramp up and ramp down. See Section "Progressive System Clock Switching" for details on how to set these values.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 2Ch offset = 2Ch



MC_CGM_PCS_DIVE4 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVE	Divider End Value - This is the clock divider end value for the clock ramp-down phase when switching from the PLL1 PHI.

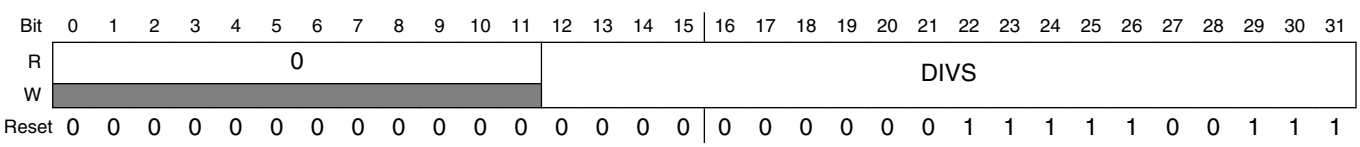
25.3.10 PCS Divider Start for PLL1 PHI (MC_CGM_PCS_DIVS4)

These registers define the initial division value on frequency ramp-up for the progressive system clock switching when switching the system clock from or to the PLL1 PHI on ramp up and ramp down. See Section "Progressive System Clock Switching" for details on how to set these values.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 30h offset = 30h



MC_CGM_PCS_DIVS4 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVS	Divider Start Value - This is the start value of the clock divider for the clock ramp-up phase when switching to the PLL1 PHI.

25.3.11 System Clock Divider Ratio Change Register (MC_CGM_SC_DIV_RC)

This register selects whether the system clock divider ratios will change from the current configuration to the configuration in the next update. If the value of SYS_DIV_RATIO_CHNG is ‘1’ at the time of a write to the CGM_DIV_UPD_TRIG register and a pre-loaded system clock divider update is pending, the crossbar is halted while the system clock divider update takes place. If the value of SYS_DIV_RATIO_CHNG is ‘0’, the crossbar switch is not halted. This register has no effect if the system clock divider update is configured in the CGM_DIV_UPD_TYPE register to be immediate.

An example of a divider ratio not changing is divider 0 changes from divide by 1 to divide by 2, and divider 1 changes from divide by 2 to divide by 4. The ratio is 1:2 in both configurations.

An example of a divider ratio changing is divider 0 changes from divide by 1 to divide by 2, and divider 1 changes from divide by 2 to divide by 6. The ratio changes from 1:2 to 1:3.

The value of SYS_DIV_RATIO_CHNG is returned to ‘1’ automatically on each write to the CGM_DIV_UPD_TRIG register.

For usage information, see Section "Divider Update Triggering".

Address: 0h base + D0h offset = D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															SYS_DIV_RATIO_CHNG
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

MC_CGM_SC_DIV_RC field descriptions

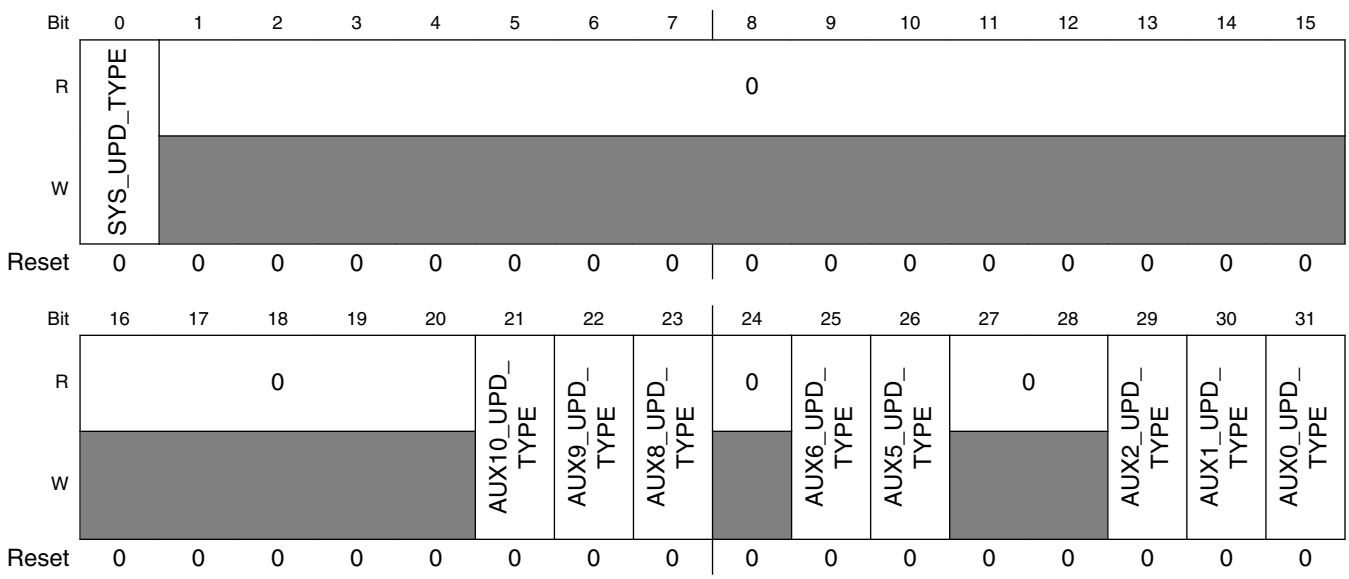
Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 SYS_DIV_ RATIO_CHNG	System Clock Divider Ratio Change 0 The system clock divider ratios will not change with the next system clock divider configuration update. 1 The system clock divider ratios will change with the next system clock divider configuration update.

25.3.12 Divider Update Type (MC_CGM_DIV_UPD_TYPE)

This register selects whether the dividers associated with a clock are updated immediately on writing to the corresponding divider configuration register or only on writing to the divider update trigger register CGM_DIV_UPD_TRIG, after which the pre-loaded configuration in the divider configuration registers wil take effect.

For usage information, see Section“Divider Update Triggering”.

Address: 0h base + D4h offset = D4h



MC_CGM_DIV_UPD_TYPE field descriptions

Field	Description
0 SYS_UPD_TYPE	System Clock Divider Update Type 0 the configuration for each system clock divider is updated immediately on writing to the corresponding CGM_SC_DCi register 1 the configuration for each system clock divider is preloaded on writing to the corresponding CGM_SC_DCi register, and the pre-loaded configurations of all the system clock dividers are updated on writing to the CGM_DIV_UPD_TRIG register

Table continues on the next page...

MC_CGM_DIV_UPD_TYPE field descriptions (continued)

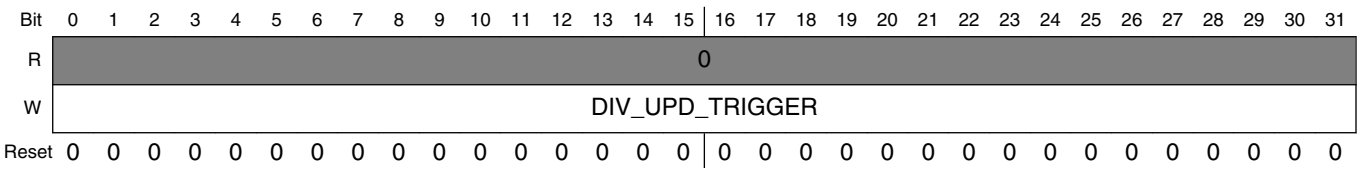
Field	Description
1–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 AUX10_UPD_TYPE	Auxiliary Clock n Divider Update Type 0 the configuration for each auxiliary clock n divider is updated immediately on writing to the corersponding CGM_ACn_DCi register 1 the configuration for each auxiliary clock n divider is preloaded on writing to the corersponding CGM_ACn_DCi register, and the pre-loaded configurations of all the auxiliary clock n dividers are updated on writing to the CGM_DIV_UPD_TRIG register
22 AUX9_UPD_TYPE	(See the Description for the AUX10_UPD_TYPE bit.)
23 AUX8_UPD_TYPE	(See the Description for the AUX10_UPD_TYPE bit.)
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 AUX6_UPD_TYPE	(See the Description for the AUX10_UPD_TYPE bit.)
26 AUX5_UPD_TYPE	(See the Description for the AUX10_UPD_TYPE bit.)
27–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 AUX2_UPD_TYPE	(See the Description for the AUX10_UPD_TYPE bit.)
30 AUX1_UPD_TYPE	(See the Description for the AUX10_UPD_TYPE bit.)
31 AUX0_UPD_TYPE	(See the Description for the AUX10_UPD_TYPE bit.)

25.3.13 Divider Update Trigger (MC_CGM_DIV_UPD_TRIG)

Writing any value to this register will cause all dividers that have corresponding bits set to "1" in the CGM_DIV_UPD_TYPE register and have pre-loaded configurations to be updated immediately. Reading this register will always return all zeroes.

For usage information, see Section “Divider Update Triggering”.

Address: 0h base + D8h offset = D8h



MC_CGM_DIV_UPD_TRIG field descriptions

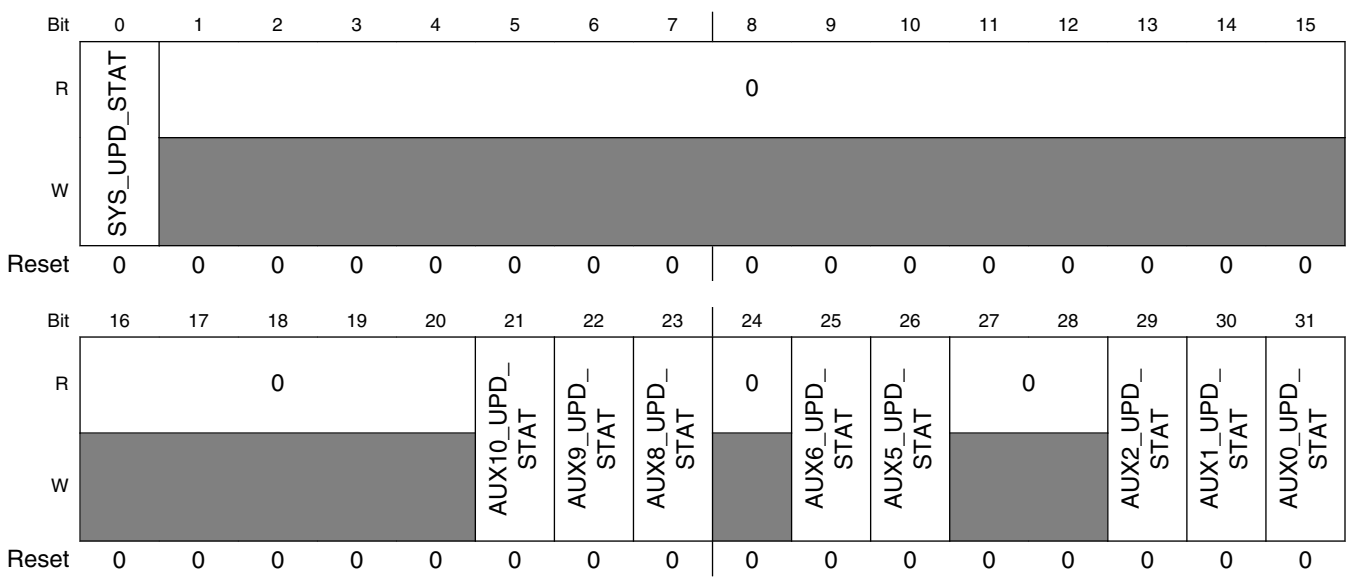
Field	Description
0–31 DIV_UPD_ TRIGGER	Writing any value to this register will cause all dividers that have corresponding bits set to "1" in the CGM_DIV_UPD_TYPE register and have pre-loaded configurations to be updated immediately.

25.3.14 Divider Update Status (MC_CGM_DIV_UPD_STAT)

This register indicates whether a divider for each clock is in the process of being updated. If the CGM_DIV_UPD_TYPE[i] is "0", CGM_DIV_UPD_STAT[i] is set to "1" immediately on writing to one of the corresponding divider configuration registers. Otherwise, CGM_DIV_UPD_STAT[i] is set to "1" immediately on writing to the CGM_DIV_UPD_TRIG register if one of the corresponding divider configurations has been pre-loaded. CGM_DIV_UPD_STAT[i] is cleared to "0" as soon as all the corresponding dividers have been updated and the dividers generate the desired frequency.

For usage information, see Section “Divider Update Triggering”.

Address: 0h base + DCh offset = DCh



MC_CGM_DIV_UPD_STAT field descriptions

Field	Description
0 SYS_UPD_STAT	System Clock Divider Update Status 0 the configuration for at least one system clock divider is not being updated 1 the configuration for at least one system clock divider is being updated
1–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 AUX10_UPD_ STAT	Auxiliary Clock n Divider Update Status 0 the configuration for at least one auxiliary clock n divider is not being updated 1 the configuration for at least one auxiliary clock n divider is being updated
22 AUX9_UPD_ STAT	(See the Description for the AUX10_UPD_STAT bit.)
23 AUX8_UPD_ STAT	(See the Description for the AUX10_UPD_STAT bit.)
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 AUX6_UPD_ STAT	(See the Description for the AUX10_UPD_STAT bit.)
26 AUX5_UPD_ STAT	(See the Description for the AUX10_UPD_STAT bit.)
27–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 AUX2_UPD_ STAT	(See the Description for the AUX10_UPD_STAT bit.)
30 AUX1_UPD_ STAT	(See the Description for the AUX10_UPD_STAT bit.)
31 AUX0_UPD_ STAT	(See the Description for the AUX10_UPD_STAT bit.)

25.3.15 System Clock Select Status (MC_CGM_SC_SS)

This register provides the current system clock source selection.

Address: 0h base + E4h offset = E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0				SELSTAT				0				SWTRG			SWIP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_SC_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	System Clock Source Selection Status - This value indicates the current source for the system clock. This value corresponds to the current value of the clk_sys_sel[3:0] input. 0000 16MHz internal RC oscillator 0001 external crystal oscillator 0010 PLL0 PHI 0011 reserved 0100 PLL1 PHI 0101 reserved 0110 reserved 0111 - 1110 reserved 1111 system clock is disabled
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–14 SWTRG	Switch Trigger cause - This value indicates the cause for the latest clock source switch. 000 reserved 001 reserved 010 switch after request from MC_ME failed due inactive target clock 011 switch after request from MC_ME failed due inactive current clock 100 switch to 16 MHz internal RC osc. due to SAFE mode request or reset succeeded 101 switch to 16 MHz internal RC osc. due to SAFE mode request or reset succeeded, but current clock source was inactive 110 reserved

Table continues on the next page...

MC_CGM_SC_SS field descriptions (continued)

Field	Description
	111 reserved
15 SWIP	Switch In Progress 0 clock source switching has completed 1 clock source switching is in progress
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.16 System Clock Divider Configuration 0 (MC_CGM_SC_DC0)

This register controls system clock divider 0. See Section "Divider Update Synchronization" for details on division value limitations.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + E8h offset = E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0									DIV					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_SC_DC0 field descriptions

Field	Description
0 DE	Divider 0 Enable 0 Disable system clock divider 0 1 Enable system clock divider 0
1–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 DIV	Divider i Division Value -0 The resultant FXBAR clock will have a period ‘DIV + 1’ times that of the system clock. If DE is set to ‘0’ (divider 1 is disabled), any write access to the DIV field is ignored and the FXBAR clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

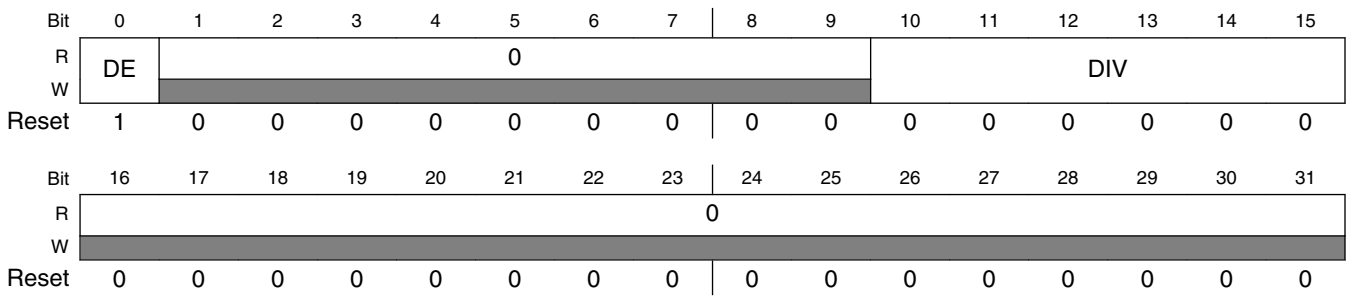
25.3.17 System Clock Divider Configuration 1 (MC_CGM_SC_DC1)

This register controls system clock divider 1. See Section "Divider Update Synchronization" for details on division value limitations.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + ECh offset = ECh



MC_CGM_SC_DC1 field descriptions

Field	Description
0 DE	Divider 1 Enable 0 Disable system clock divider 1 1 Enable system clock divider 1
1–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 DIV	Divider 1 Division Value The resultant SXBAR clock will have a period ‘DIV + 1’ times that of the system clock. If DE is set to ‘0’ (divider 1 is disabled), any write access to the DIV field is ignored and the SXBAR clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

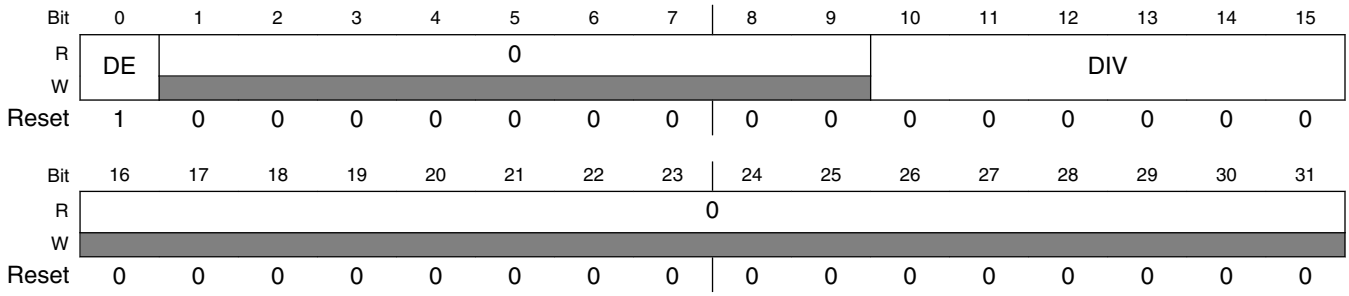
25.3.18 System Clock Divider Configuration 2 (MC_CGM_SC_DC2)

This register controls system clock divider 2. See Section "Divider Update Synchronization" for details on division value limitations.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + F0h offset = F0h



MC_CGM_SC_DC2 field descriptions

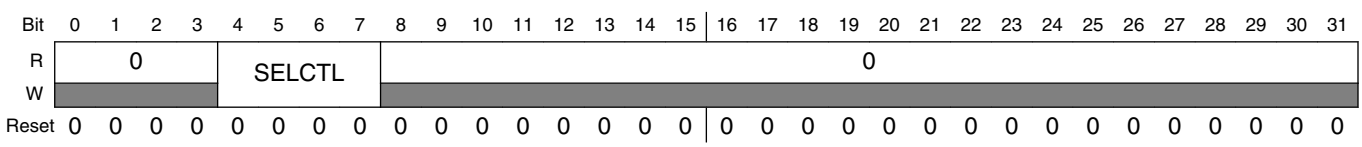
Field	Description
0 DE	Divider 2 Enable 0 Disable system clock divider 2 1 Enable system clock divider 2
1–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 DIV	Divider 2 Division Value The resultant PBRIDGE clock will have a period 'DIV + 1' times that of the system clock. If DE is set to '0' (divider 1 is disabled), any write access to the DIV field is ignored and the PBRIDGE clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.19 Aux Clock 0 Select Control (MC_CGM_AC0_SC)

This register is used to select the current clock source for the following clocks:

- undivided: unused
- divided by auxiliary clock 0 divider 0: peripheral clock
- divided by auxiliary clock 0 divider 1: sigma delta ADC clock
- divided by auxiliary clock 0 divider 2: SAR ADC clock
- divided by auxiliary clock 0 divider 3: DSPI clock0
- divided by auxiliary clock 0 divider 4: DSPI clock 1

Address: 0h base + 100h offset = 100h



MC_CGM_AC0_SC field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELCTL	Auxiliary Clock 0 Source Selection Control - This value selects the current source for auxiliary clock 0. 0000 internal RC osc. 0001 external crystal osc. 0010 PLL0 PHI Other values are reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.20 Aux Clock 0 Select Status (MC_CGM_AC0_SS)

This register provides the current auxiliary clock 0 source selection.

Address: 0h base + 104h offset = 104h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELSTAT				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC0_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	Auxiliary Clock 0 Source Selection Status - This value indicates the current source for auxiliary clock 0. 0000 16 Mhz Internal RC osc 0001 external crystal osc. 0010 PLL0 PHI 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved

Table continues on the next page...

MC_CGM_AC0_SS field descriptions (continued)

Field	Description
	1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.21 Aux Clock 0 Divider Configuration 0 (MC_CGM_AC0_DC0)

This register controls auxilliary clock 0 divider 0.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 108h offset = 108h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0											DIV			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC0_DC0 field descriptions

Field	Description
0 DE	Divider 0 Enable 0 Disable auxiliary clock 0 divider 0 1 Enable auxiliary clock 0 divider 0
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DIV	Divider 0 Division Value The resultant peripheral clock will have a period ‘DIV + 1’ times that of the aixiliary clock 0. If DE is set to ‘0’ (divider 1 is disabled), any write access to the DIV field is ignored and the peripheral clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.22 Aux Clock 0 Divider Configuration 1 (MC_CGM_AC0_DC1)

This register controls auxilliary clock 0 divider 1.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 10Ch offset = 10Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0								DIV						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC0_DC1 field descriptions

Field	Description
0 DE	Divider 1 Enable 0 Disable auxiliary clock 0 divider 1 1 Enable auxiliary clock 0 divider 1
1–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 DIV	Divider 1 Division Value The resultant SD-ADC clock will have a period ‘DIV + 1’ times that of the auxiliary clock 0. If DE is set to ‘0’ (divider 1 is disabled), any write access to the DIV field is ignored and the SD_ADC clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

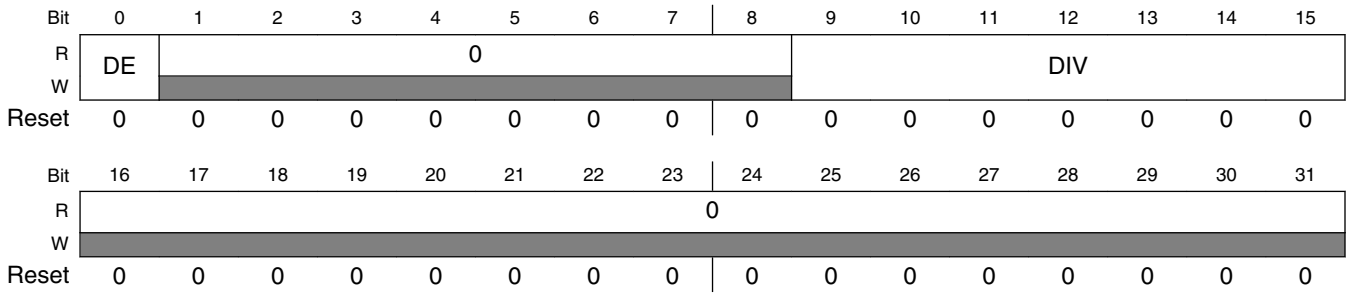
25.3.23 Aux Clock 0 Divider Configuration 2 (MC_CGM_AC0_DC2)

This register controls auxilliary clock 0 divider 2.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 110h offset = 110h



MC_CGM_AC0_DC2 field descriptions

Field	Description
0 DE	Divider 2 Enable 0 Disable auxiliary clock 0 divider 2 1 Enable auxiliary clock 0 divider 2
1–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 DIV	Divider 2 Division Value The divided SAR ADC clock outputs will be generated such that it will have a period 'DIV + 1' times that of the auxliary clock 0. If DE is set to '0' (divider 2 is disabled), any write access to the DIV field is ignored
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

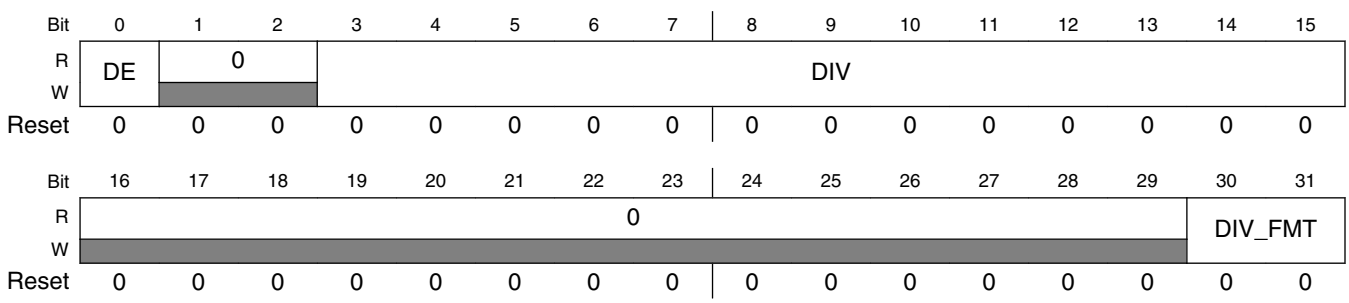
25.3.24 Aux Clock 0 Divider Configuration 3 (MC_CGM_AC0_DC3)

This register controls auxilliary clock 0 divider 3.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 114h offset = 114h



MC_CGM_AC0_DC3 field descriptions

Field	Description
0 DE	Divider 3 Enable 0 Disable auxiliary clock 0 divider 3 1 Enable auxiliary clock 0 divider 3
1–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–15 DIV	Divider 3 Division Value - For DSPI_Mx or DSPI supporting MSC capabilities. The divided DSPI 0 clock output will be generated such that it will have a period division factor (as given by DIV and DIV_FMT) times that of the auxliary clock 0. If DE is set to '0' (divider 3 is disabled), any write access to the DIV field is ignored
16–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 DIV_FMT	Divider i Value Format This falue indicates the format of the division value for calculating the division factor 00 Division factor for clock divider 3= (DIV + 1) / 1 01 Division factor for clock divider 3= (DIV + 1) / 10 10 Division factor for clock divider 3= (DIV + 1) / 100 11 Division factor for clock divider 3= (DIV + 1) / 1,000

25.3.25 Aux Clock 0 Divider Configuration 4 (MC_CGM_AC0_DC4)

This register controls auxilliary clock 0 divider 4.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

Address: 0h base + 118h offset = 118h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0									DIV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC0_DC4 field descriptions

Field	Description
0 DE	Divider reserved Enable 0 Disable auxiliary clock 0 divider reserved 1 Enable auxiliary clock 0 divider reserved

Table continues on the next page...

MC_CGM_AC0_DC4 field descriptions (continued)

Field	Description
1–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 DIV	Divider reserved Division Value - For DSPI not supporting MSC. Divider range between 1 to 16. The divided DSPI 1 clock output will be generated such that it will have a period 'DIV + 1' times that of the auxiliary clock 0. If DE is set to '0' (divider reserved is disabled), any write access to the DIV field is ignored
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.26 Aux Clock 1 Select Control (MC_CGM_AC1_SC)

This register is used to select the current clock source for the following clocks:

- undivided: unused
- divided by auxiliary clock 1 divider 0: BD LFAST PLL clock

Address: 0h base + 120h offset = 120h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELCTL				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC1_SC field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELCTL	Auxiliary Clock 1 Source Selection Control -? This value selects the current source for auxiliary clock 1. 0000 reserved 0001 external crystal osc. 0010 PLL0 PHI 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 SIPI Clock In 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved

Table continues on the next page...

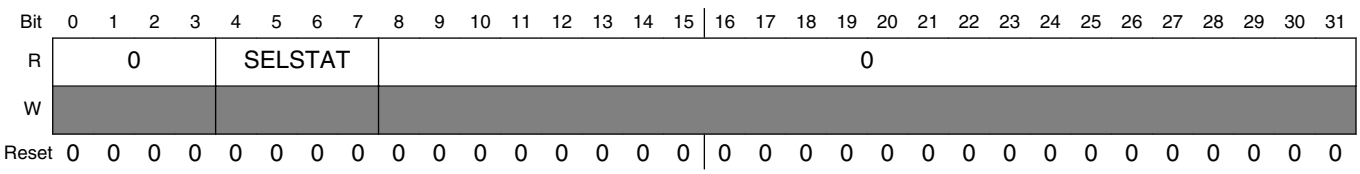
MC_CGM_AC1_SC field descriptions (continued)

Field	Description
	1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.27 Aux Clock 1 Select Status (MC_CGM_AC1_SS)

This register provides the current auxiliary clock 0 source selection.

Address: 0h base + 124h offset = 124h



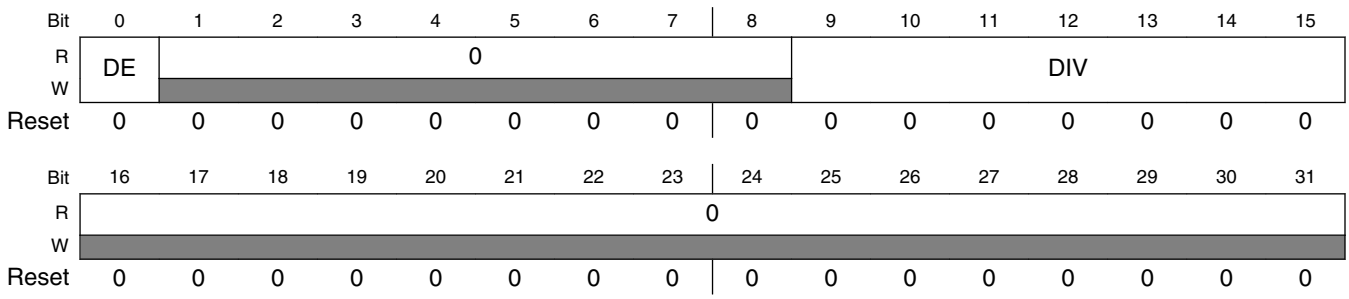
MC_CGM_AC1_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	Auxiliary Clock 1 Source Selection Status - This value indicates the current source for auxiliary clock 1. 0000 reserved 0001 external crystal osc. 0010 PLL0 PHI 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 SIPI Clock In 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.28 Aux Clock 1 Divider Configuration 0 (MC_CGM_AC1_DC0)

These registers controls auxiliary clock 1 divider 0.

Address: 0h base + 128h offset = 128h



MC_CGM_AC1_DC0 field descriptions

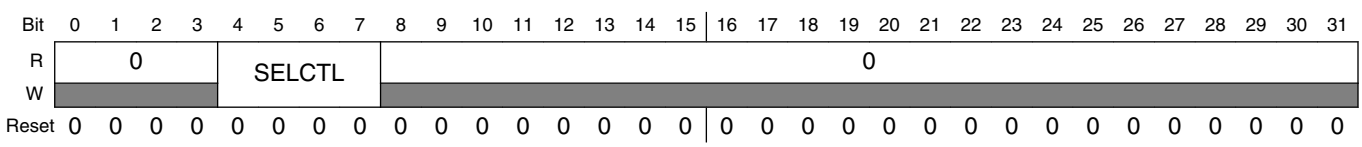
Field	Description
0 DE	Divider 0 Enable 0 Disable auxiliary clock 1 divider 0 1 Enable auxiliary clock 1 divider 0
1–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 DIV	Divider 0 Division Value - The resultant BD LFAST PLL clock will have a period 'DIV0 + 1' times that of auxiliary clock 1. If the DE0 is set to 0 (Divider 0 is disabled), any write access to the DIV0 field is ignored and the BD LFAST PLL clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.29 Aux Clock 2 Select Control (MC_CGM_AC2_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 2 divider 0: SYSCLK1 pin clock

Address: 0h base + 140h offset = 140h



MC_CGM_AC2_SC field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELCTL	Auxiliary Clock 2 Source Selection Control This value selects the current source for auxiliary clock 2. 0000 16 MHz int. RC osc. 0001 reserved 0010 PLL0 PHI 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.30 Aux Clock 2 Select Status (MC_CGM_AC2_SS)

This register provides the current auxiliary clock 2 source selection.

Address: 0h base + 144h offset = 144h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELSTAT				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC2_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	Auxiliary Clock 2 Source Selection Status This value indicates the current source for auxiliary clock 2. 0000 16 MHz int. RC osc.

Table continues on the next page...

MC_CGM_AC2_SS field descriptions (continued)

Field	Description
	0001 reserved 0010 PLL0 PHI 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.31 Aux Clock 2 Divider Configuration 0 (MC_CGM_AC2_DC0)

This register controls auxiliary clock 2 divider 0 which provides the clock to SENT module.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0h base + 148h offset = 148h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0							DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC2_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 2 divider 0 1 Enable auxiliary clock 2 divider 0
1–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 DIV	Divider Division Value - The resultant SENT clock will have a period 'DIV + 1' times that of auxiliary clock 2. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the SENT clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.32 Aux Clock 3 Select Control (MC_CGM_AC3_SC)

This register is used to select the current clock source for the PLL0 reference clock.

Address: 0h base + 160h offset = 160h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELCTL				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC3_SC field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELCTL	Auxiliary Clock 3 Source Selection Control This value selects the current source for auxiliary clock 3. 0000 16MHz internal RC osc 0001 external crystal osc 0010 reserved 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved

Table continues on the next page...

MC_CGM_AC3_SC field descriptions (continued)

Field	Description
	1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.33 Aux Clock 3 Select Status (MC_CGM_AC3_SS)

This register provides the current auxiliary clock 3 source selection.

Address: 0h base + 164h offset = 164h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELSTAT				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC3_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	Auxiliary Clock 3 Source Selection Status This value indicates the current source for auxiliary clock 3. 0000 16MHz internal RC osc 0001 external crystal osc 0010 reserved 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved

Table continues on the next page...

MC_CGM_AC3_SS field descriptions (continued)

Field	Description
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.34 Aux Clock 4 Select Control (MC_CGM_AC4_SC)

This register is used to select the current clock source for the PLL1 reference clock.

Address: 0h base + 180h offset = 180h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELCTL				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC4_SC field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELCTL	Auxiliary Clock 4 Source Selection Control This value selects the current source for auxiliary clock 4. 0000 16MHz internal RC osc 0001 external crystal osc 0010 reserved 0011 PLL0 PHI1 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.35 Aux Clock 4 Select Status (MC_CGM_AC4_SS)

This register provides the current auxiliary clock 4 source selection.

Address: 0h base + 184h offset = 184h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELSTAT				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC4_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	Auxiliary Clock 4 Source Selection Status This value indicates the current source for auxiliary clock 4. 0000 16MHz internal RC osc 0001 external crystal osc 0010 reserved 0011 PLL0 PHI1 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.36 Aux Clock 5 Select Control (MC_CGM_AC5_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 5 divider 0: SYSCLK1 pin clock

Address: 0h base + 1A0h offset = 1A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELCTL				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC5_SC field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELCTL	Auxiliary Clock 5 Source Selection Control This value selects the current source for auxiliary clock 5. 0000 16 MHz int. RC osc. 0001 reserved. 0010 PLL0 PHI 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.37 Aux Clock 5 Select Status (MC_CGM_AC5_SS)

This register provides the current auxiliary clock 5 source selection.

Address: 0h base + 1A4h offset = 1A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELSTAT				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC5_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	Auxiliary Clock 5 Source Selection Status This value indicates the current source for auxiliary clock 5. 0000 16 MHz int. RC osc. 0001 reserved. 0010 PLL0 PHI 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.38 Aux Clock 5 Divider Configuration 0 (MC_CGM_AC5_DC0)

This register controls auxiliary clock 5 divider 0.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0h base + 1A8h offset = 1A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0									DIV					
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC5_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 divider 0 1 Enable auxiliary clock 5 divider 0
1–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 DIV	Divider Division Value - The resultant eTPU2 and REACM2 clock will have a period 'DIV + 1' times that of auxiliary clock 5 . If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the eTPU2 and REACM2 clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.39 Aux Clock 5 Divider Configuration 1 (MC_CGM_AC5_DC1)

This register controls auxiliary clock 5 divider 1.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0h base + 1ACh offset = 1ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0									DIV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC5_DC1 field descriptions

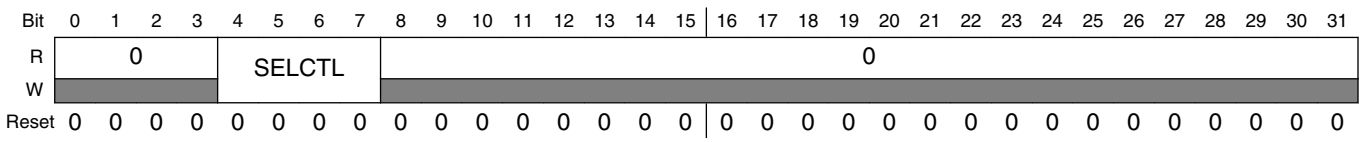
Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 5 divider 1 1 Enable auxiliary clock 5 divider 1
1–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 DIV	Divider Division Value - The resultant eMIOS and BCTU clock will have a period 'DIV + 1' times that of auxiliary clock 5. If DE is set to 0 (divider 1 is disabled), any write access to the DIV field is ignored and the eMIOS and BCTU clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.40 Aux Clock 6 Select Control (MC_CGM_AC6_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 6 divider 0: Clock Out pin clock

Address: 0h base + 1C0h offset = 1C0h



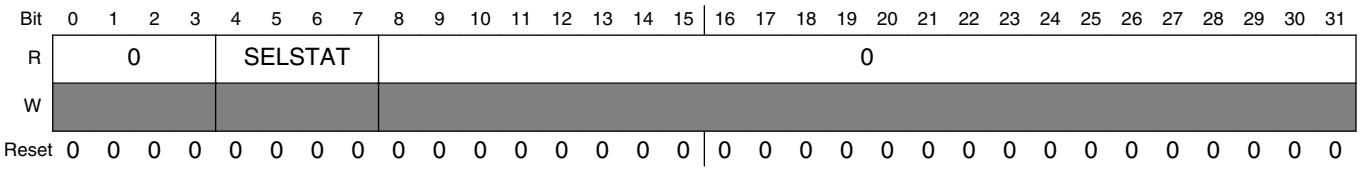
MC_CGM_AC6_SC field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELCTL	0000 Internal RC osc 0001 external crystal osc 0010 PLL0 PHI 0011 reserved 0100 PLL1 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.41 Aux Clock 6 Select Status (MC_CGM_AC6_SS)

This register provides the current auxiliary clock 6 source selection.

Address: 0h base + 1C4h offset = 1C4h



MC_CGM_AC6_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	Auxiliary Clock 6 Source Selection Status This value indicates the current source for auxiliary clock 6. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 PLL0 PHI 0011 reserved 0100 PLL1 PHI 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

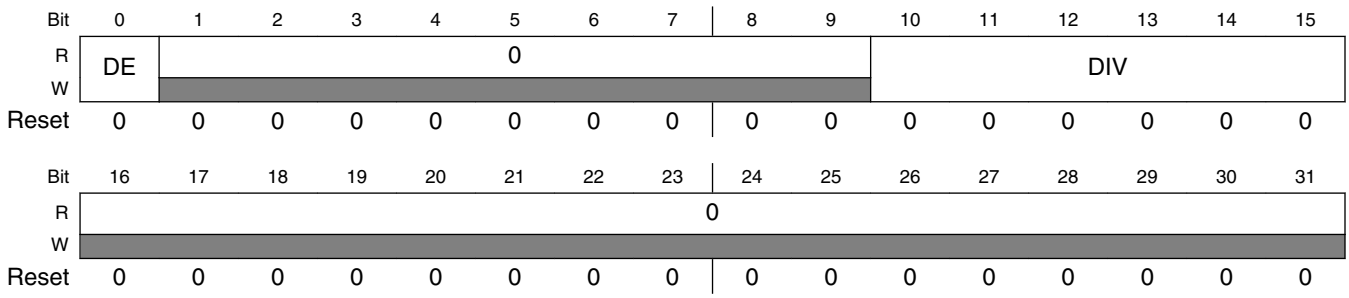
25.3.42 Aux Clock 6 Divider Configuration 0 (MC_CGM_AC6_DC0)

This register controls auxiliary clock 6 divider 0.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0h base + 1C8h offset = 1C8h



MC_CGM_AC6_DC0 field descriptions

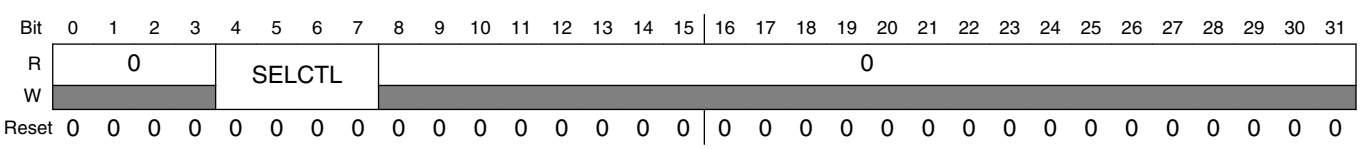
Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 6 divider 0 1 Enable auxiliary clock 6 divider 0
1–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 DIV	Divider Division Value - The resultant Clock out pin clock will have a period 'DIV + 1' times that of auxiliary clock 6. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the Clock out pin clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.43 Aux Clock 8 Select Control (MC_CGM_AC8_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 8 divider 0: FlexCAN clock

Address: 0h base + 200h offset = 200h



MC_CGM_AC8_SC field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_CGM_AC8_SC field descriptions (continued)

Field	Description
4–7 SELCTL	Auxiliary Clock 8 Source Selection Control This value selects the current source for auxiliary clock 8. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 PLL0 PHI 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.44 Aux Clock 8 Select Status (MC_CGM_AC8_SS)

This register provides the current auxiliary clock 8 source selection.

Address: 0h base + 204h offset = 204h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELSTAT				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC8_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	Auxiliary Clock 8 Source Selection Status This value indicates the current source for auxiliary clock 8. 0000 16 MHz int. RC osc. 0001 external crystal osc.

Table continues on the next page...

MC_CGM_AC8_SS field descriptions (continued)

Field	Description
	0010 PLL0 PHI 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

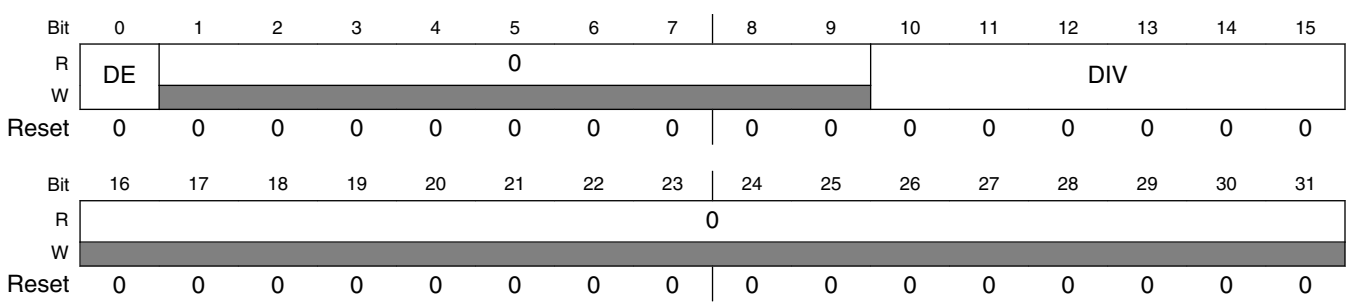
25.3.45 Aux Clock 8 Divider Configuration 0 (MC_CGM_AC8_DC0)

This register controls auxiliary clock 8 divider 0.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0h base + 208h offset = 208h



MC_CGM_AC8_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 8 divider 0

Table continues on the next page...

MC_CGM_AC8_DC0 field descriptions (continued)

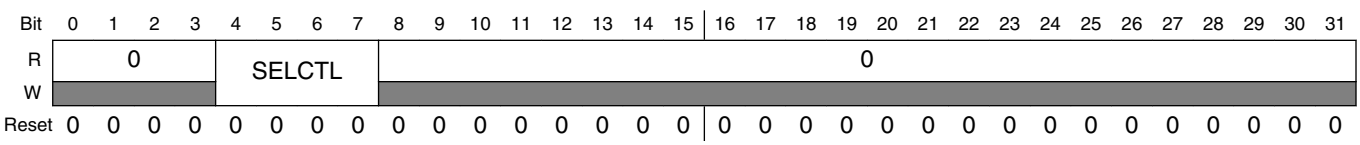
Field	Description
	1 Enable auxiliary clock 8 divider 0
1–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 DIV	Divider Division Value - The resultant FlexCAN clock will have a period 'DIV + 1' times that of auxiliary clock 8. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the CAN clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.46 Aux Clock 9 Select Control (MC_CGM_AC9_SC)

This register is used to select the current clock source for the following clocks:

- undivided: (unused)
- divided by auxiliary clock 9 divider 0: RTI clock

Address: 0h base + 220h offset = 220h



MC_CGM_AC9_SC field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELCTL	Auxiliary Clock 9 Source Selection Control This value selects the current source for auxiliary clock 9. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 reserved 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved

Table continues on the next page...

MC_CGM_AC9_SC field descriptions (continued)

Field	Description
	1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.47 Aux Clock 9 Select Status (MC_CGM_AC9_SS)

This register provides the current auxiliary clock 9 source selection.

Address: 0h base + 224h offset = 224h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELSTAT				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC9_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	Auxiliary Clock 9 Source Selection Status This value indicates the current source for auxiliary clock 9. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 reserved 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.48 Aux Clock 9 Divider Configuration 0 (MC_CGM_AC9_DC0)

This register controls auxiliary clock 9 divider 0.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0h base + 228h offset = 228h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0									DIV					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC9_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 9 divider 0
1–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 DIV	Divider Division Value - The resultant RTI clock will have a period ‘DIV + 1’ times that of auxiliary clock 9. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the RTI clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.49 Aux Clock 10 Select Control (MC_CGM_AC10_SC)

This register is used to select the current clock source for the Ethernet TXCLK_REF clock.

Address: 0h base + 240h offset = 240h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELCTL				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC10_SC field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELCTL	Auxiliary Clock 10 Source Selection Control This value selects the current source for auxiliary clock 10. 0000 16 MHz int. RC osc. 0001 external crystal osc. 0010 PLL0 PHI 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.50 Aux Clock 10 Select Status (MC_CGM_AC10_SS)

This register provides the current auxiliary clock 10 source selection.

Address: 0h base + 244h offset = 244h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELSTAT				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC10_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	Auxiliary Clock 10 Source Selection Status This value indicates the current source for auxiliary clock 10. 0000 16 MHz int. RC osc.

Table continues on the next page...

MC_CGM_AC10_SS field descriptions (continued)

Field	Description
	0001 external crystal osc. 0010 PLL0 PHI 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.3.51 Aux Clock 10 Divider Configuration 0 (MC_CGM_AC10_DC0)

This register controls auxiliary clock 10 divider 0.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception; however, the value will not be loaded with the new value.

Address: 0h base + 248h offset = 248h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0											DIV			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC10_DC0 field descriptions

Field	Description
0 DE	Divider Enable 0 Disable auxiliary clock 10 divider 0 1 Enable auxiliary clock 10 divider 0
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DIV	Divider Division Value - The resultant Ethernet TXCLK_REF clock will have a period 'DIV + 1' times that of auxiliary clock 10. If DE is set to 0 (divider 0 is disabled), any write access to the DIV field is ignored and the Ethernet TXCLK_REF clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

25.4 Functional Description

This section describes the functions of the MC_CGM module.

25.4.1 System Clock Generation

The figure below shows the block diagram of the system clock generation logic.The MC_ME provides the system clock select, progressive clock switching control, and switch mask (see MC_ME chapter for more details), and the MC_RGM provides the safe clock request (see MC_RGM chapter for more details). The safe clock request forces the selector to select the clk_sys_src[0] as the system clock and to ignore the system clock select.

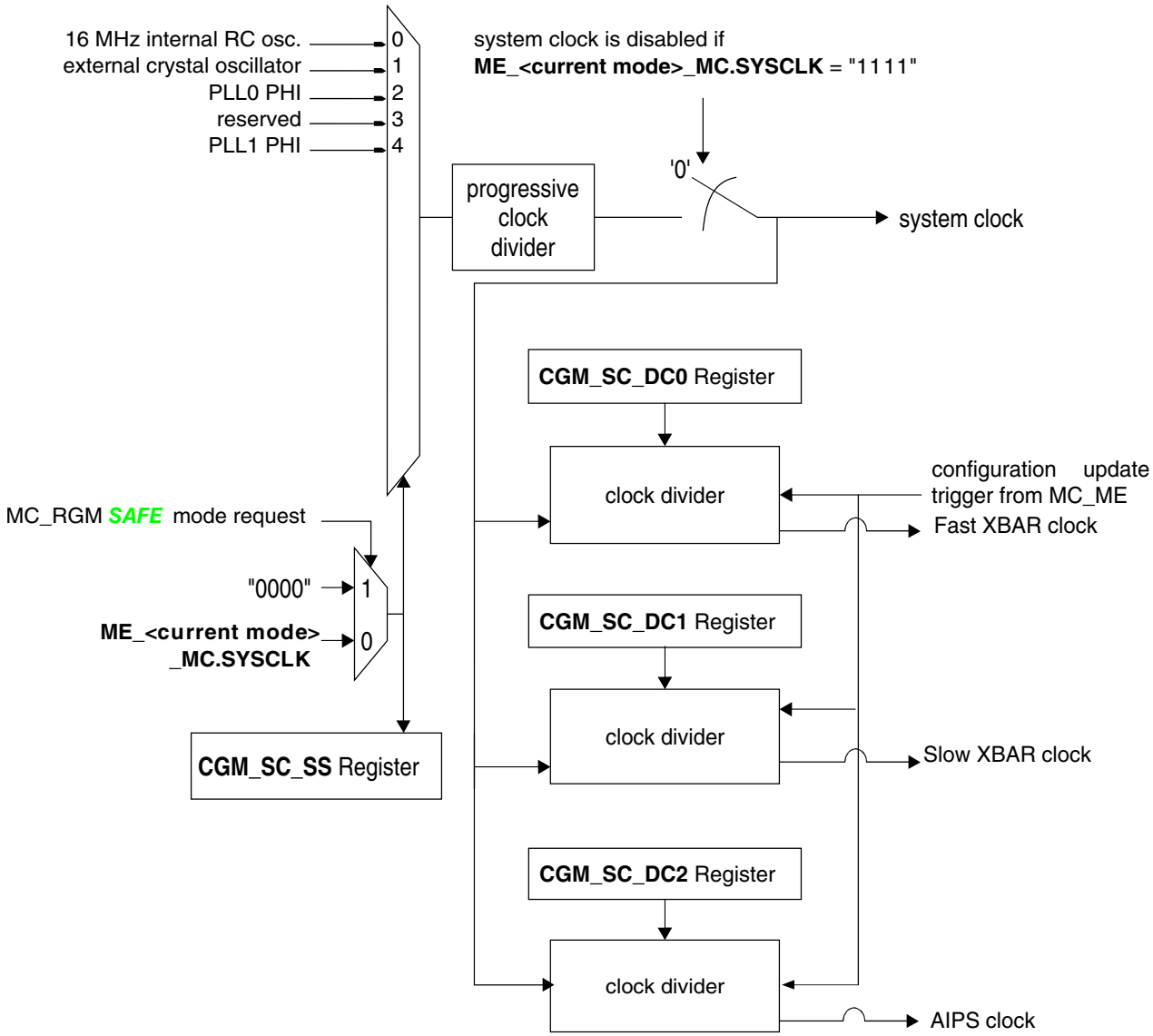


Figure 25-53. MC_CGM System clock generation overview

25.4.1.1 System Clock Source Selection

During normal operation the system clock selection is controlled

- on a ***SAFE*** mode or reset event, by the the MC_RGM
- otherwise, by the MC_ME

25.4.1.2 Progressive System Clock Switching

Based on the power level values of the current and target modes, the MC_ME requests the MC_CGM to ramp the system clock frequency down and/or up. During ramp-down, the rate of the frequency change is based on the **CGM_PCS_SDUR**, **CGM_PCS_DIVC*n***, and **CGM_PCS_DIVE*n*** registers, where *n* corresponds to the current system clock source selection. During ramp-up, the rate of the frequency change is based on the **CGM_PCS_SDUR**, **CGM_PCS_DIVC*n***, and **CGM_PCS_DIVS*n*** registers, where *n* corresponds to the target system clock source selection.

25.4.1.2.1 Generic Clock Change Requirements

For a maximum allowed change of the frequency (f_{chg}) and for a given source clock frequency f_{src} (current or target clock source), the maximum allowed frequency change rate a_{max} is given in the following equation.

$$a_{max} = \frac{f_{chg}}{f_{src}}$$

25.4.1.2.2 Configuration of CGM_PCS_SDUR

The switch duration field **CGM_PCS_SDUR[SDUR]** defines the duration of one system clock switch step in terms of 16 MHz internal RC osc. After the expiration of this time, the module changes the clock divider value which changes the frequency of the system clock.

25.4.1.2.3 Configuration of CGM_PCS_DIVC*n*[RATE]

For a given maximum allowed frequency change rate a_{max} the value d to be programmed into the **PCS_DIVC*n*[RATE]** register is given in the following table.

Table 25-53. MC_CGM CGM_PCS_DIVC*n*[RATE] Values

a_{max}	d = CGM_PCS_DIVC*n*[RATE]	RATE Field Value
0.05	0.012	00001100
0.10	0.048	00110000
0.15	0.112	01110000
0.20	0.184	10111000

25.4.1.2.4 Generic Clock Change Properties

The number *k* of required steps to reach or leave the divided system clock with frequency f_{safe} from the source clock with frequency f_{src} is given in the following equation.

$$k = \left\lceil 0.5 + \sqrt{0.25 - \frac{2\left(1 - \frac{f_{src}}{f_{safe}}\right)}{d}} \right\rceil$$

Where f_{safe} is the frequency of the 16 MHz internal RC osc. clock.

25.4.1.2.5 Clock Ramp-Down

The clock ramp-down starts with the divider value 1 and with the given divider increment value **PCD_DIVC*n*[RATE]** and ends with the given divider value **PCS_DIVE*n*[DIVE]**.

The divider end value for clock ramp-down is given in the following equation.

$$\text{PCS_DIVEn[DIVE]} = \frac{f_{curr}}{f_{safe}} \times 1000 - 1$$

Where f_{curr} is the frequency of the currently selected system clock source.

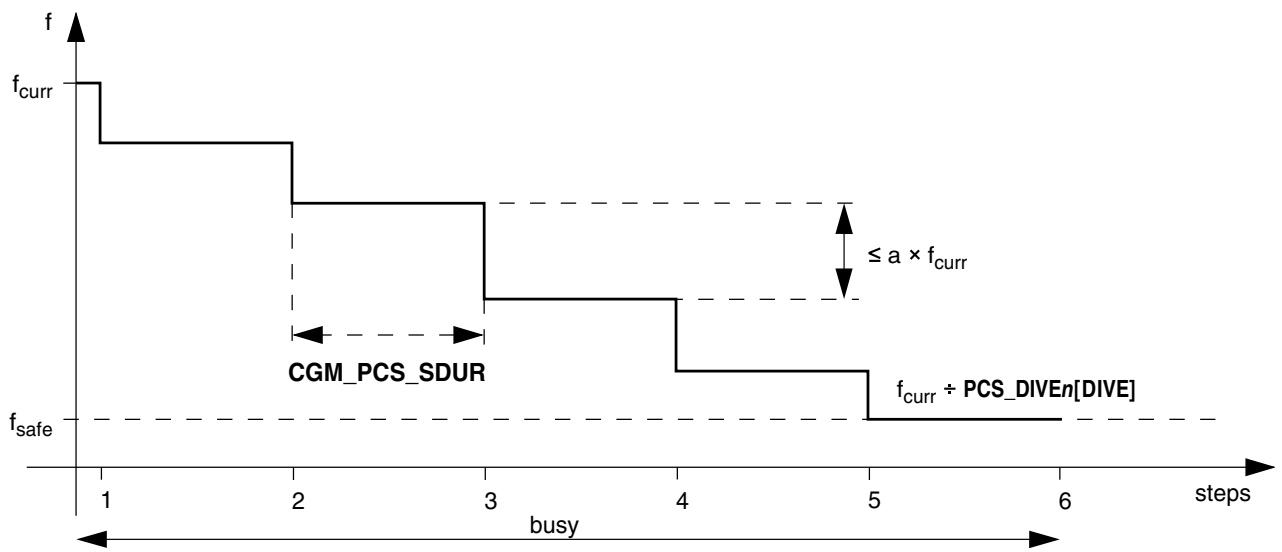


Figure 25-54. MC_CGM System Clock Ramp-Down Timing (k = 6 example)

25.4.1.2.6 Clock Ramp-Up

The clock ramp-up starts with the given divider value **PCS_DIVS*n*[DIVS]** and with the given divider decrement value **PCD_DIVC*n*[INIT]** and ends with the divider value 1.

The initial divider change start value **PCS_DIVC*n*[INIT]** for system clock ramp-up is given in the following equation.

PCS_DIVCn[INIT] = d ×k ×1000

Where k is calculated from the equation in section Generic Clock Change Properties using the target system clock source frequency f_{targ}. The divider start value for clock ramp-up is given in the following equation.

$$PCS_DIVSn[DIVS] = \left(1 + d\frac{k(k+1)}{2}\right) \times 1000 - 1$$

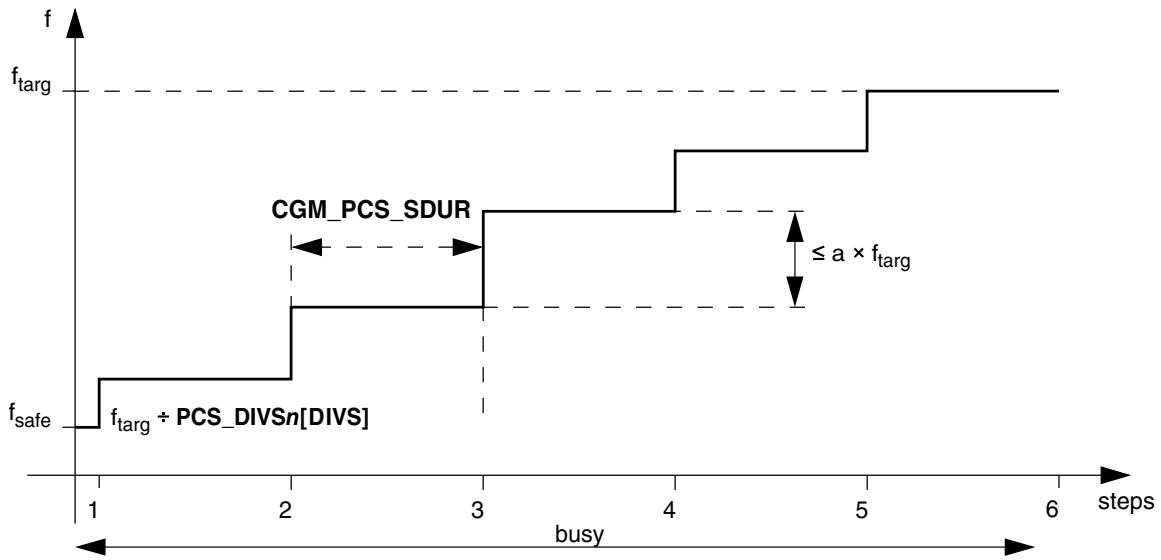


Figure 25-55. MC_CGM System Clock Ramp-Up Timing (k = 6 example)

25.4.1.2.7 PCS code sample

It is highly recommended to use PCS in both directions whenever the system clock frequency changes, while going from one MC_ME mode to another. If system clock frequency changes abruptly, it'll cause sharp current load transition on the voltage supply which may lead to unexpected voltage overshoots and undershoot that may even lead to an LVD / HVD reset.

```
INFO("irc to pll1 phi", "*** CONFIGURE PCFS SETTINGS ***");
W32(MC_CGM_PCS_SDUR,  0x000000A0);                              //PCFS settings
W32(MC_CGM_PCS_DIVC4, 0x021B000C);
W32(MC_CGM_PCS_DIVE4, 0x0000346B);
W32(MC_CGM_PCS_DIVS4, 0x000030D3);

INFO(STIM, " *** GENERATE FBAR_CLK AT 1 Sys Clock ***");
INFO(STIM, " *** GENERATE SBAR_CLK AT 1/2 Sys Clock ***");
INFO(STIM, " *** GENERATE PBRIDGE_0_CLK/PBRIDGE_1_CLK AT 1/4 Sys Clock ***");

BSET32(MC_CGM_SC_DC0,0x00000000);
BSET32(MC_CGM_SC_DC1,0x00010000);
BSET32(MC_CGM_SC_DC2,0x00030000);
REM32(MC_CGM_SC_DC0, 0x00000000, 0x00010000);
REM32(MC_CGM_SC_DC1, 0x00010000, 0x00010000);
REM32(MC_CGM_SC_DC2, 0x00030000, 0x00030000);

INFO(STIM, "*** PROGRAM NDIV, RFD AND MFD FOR PLL0 ***");
W32(PLLD_PLL0_DIVR, PREDIV2 | MFD_47 | RFD_PHI1_8 | RFD_PHI_2);
```

```
INFO(STIM, "*** SET PLL0ON TO EXIT PD MODE ***");
BSET32(MC_ME_RUN0_MC, 0x00000060);
entry_to_target_mode(RUN0);

INFO(STIM, "*** WAIT FOR LOCK0 TO ASSERT ***");
REM32(PLLD_PLL0_SR, LOCK, LOCK);
INFO(STIM, "*** PLL0 HAS ACHIEVED LOCK ***");

INFO(STIM, "*** Select PLL0 AS REF CLK FOR PLL1 ***");
W32(MC_CGM_AC4_SC, 0x03000000);

INFO(STIM, "*** PROGRAM RFD, MFD to PLL1 ***");
W32(PLLD_PLL1_DIVR,  MFD_16 | RFD_PHI_2);

INFO(STIM, "*** SET PLL1ON TO EXIT PD MODE ***");
BSET32(MC_ME_RUN0_MC, 0x00000080);
entry_to_target_mode(RUN0);

INFO(STIM, "*** WAIT FOR LOCK1 TO ASSERT ***");
REM32(PLLD_PLL1_SR, LOCK, LOCK);
INFO(STIM, "*** PLL1 HAS ACHIEVED LOCK ***");


INFO(STIM, "*** SWITCHING SYSTEM CLOCK TO PLL1_PHI ***");
W32(MC_ME_RUN0_MC, 0x701300f4);                    //Changing the power level which is
crucial for PCS activation
entry_to_target_mode(RUN0);                                  //Actual PCS would be visible
at this point
```

25.4.1.3 System Clock Disable

During the ***TEST*** mode, the system clock can be disabled by the ME.

25.4.1.4 System Clock Dividers

The MC_CGM generates the following derived clocks from the system clock.

- Fast XBAR clock - controlled by the **CGM_SC_DC0** register
- Slow XBAR clock - controlled by the **CGM_SC_DC1** register
- AIPS clock - controlled by the **CGM_SC_DC2** register

25.4.2 Auxiliary Clock Generation

The figures below show the block diagram of the auxiliary clock generation logic. See the following sections for auxiliary clock selection control:

- "Auxiliary Clock 0 Select Control Register (CGM_AC0_SC)"

- "Auxiliary Clock 1 Select Control Register (CGM_AC1_SC)"
- "Auxiliary Clock 3 Select Control Register (CGM_AC3_SC)"
- "Auxiliary Clock 4 Select Control Register (CGM_AC4_SC)"
- "Auxiliary Clock 6 Select Control Register (CGM_AC6_SC)"
- "Auxiliary Clock 8 Select Control Register (CGM_AC8_SC)"
- "Auxiliary Clock 9 Select Control Register (CGM_AC9_SC)"
- "Auxiliary Clock 10 Select Control Register (CGM_AC10_SC)"

NOTE

All dividers default to the IRC 16MHz clock. The user needs to program the selection register properly to enable the right clock.

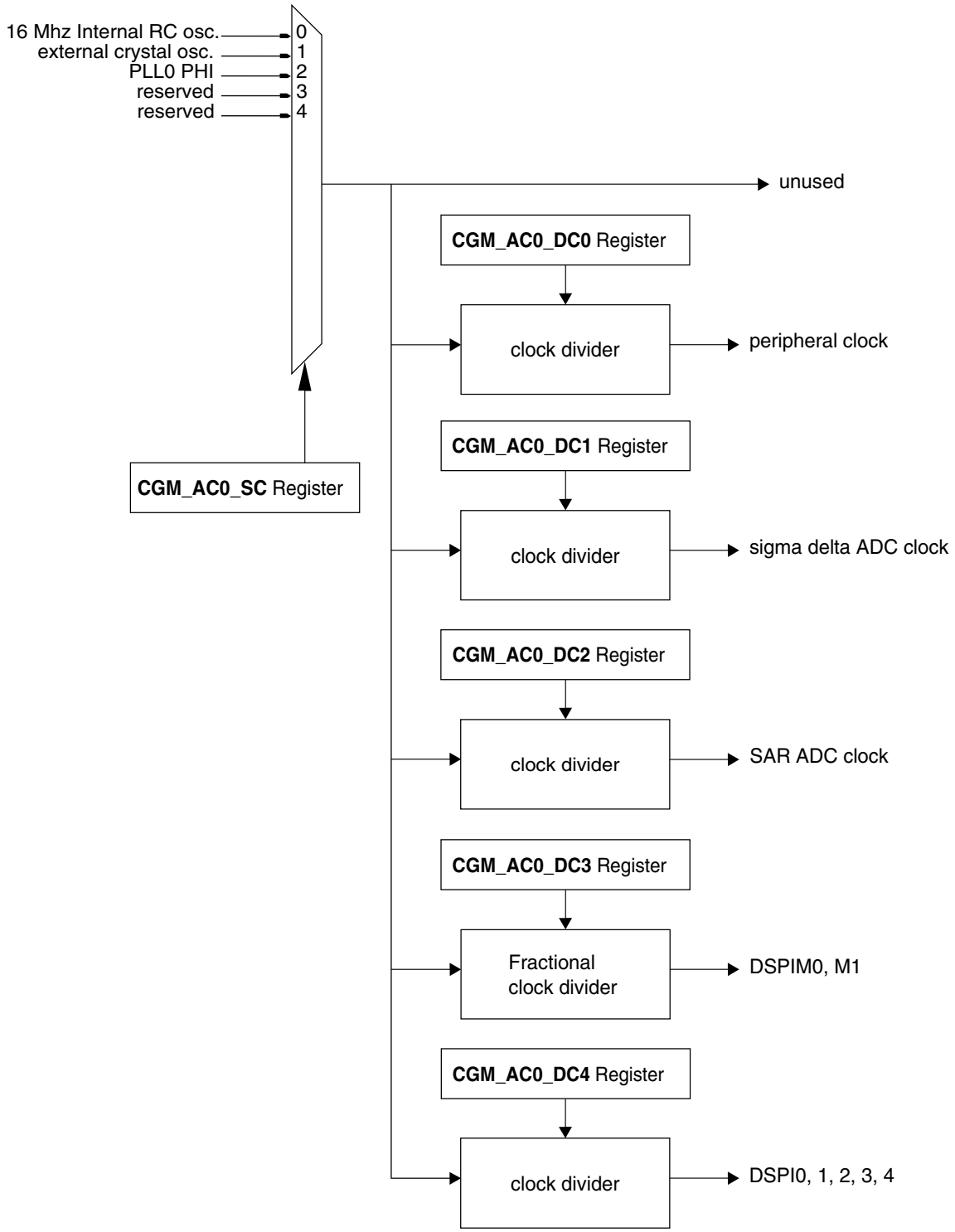


Figure 25-56. MC_CGM auxiliary clock 0 generation

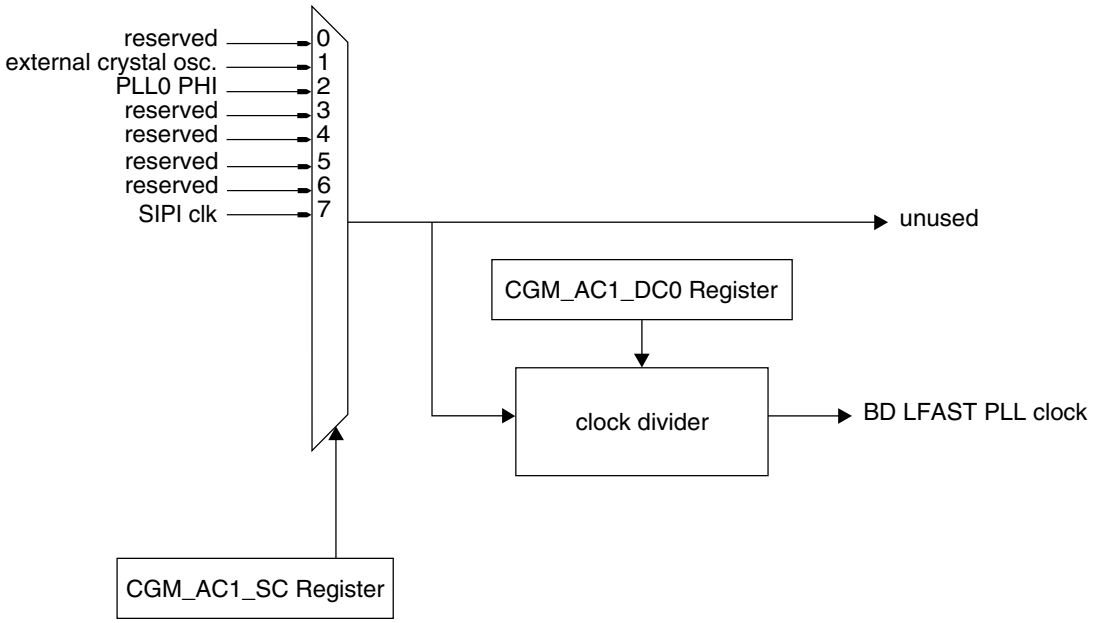


Figure 25-57. MC_CGM auxiliary clock 1 generation

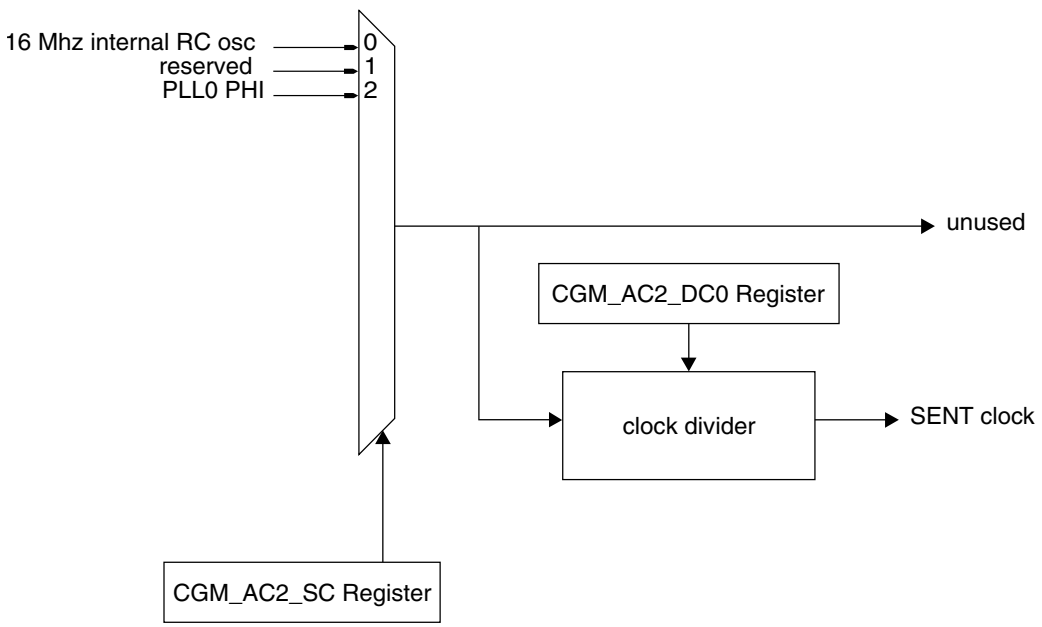


Figure 25-58. MC_CGM auxiliary clock 2 generation

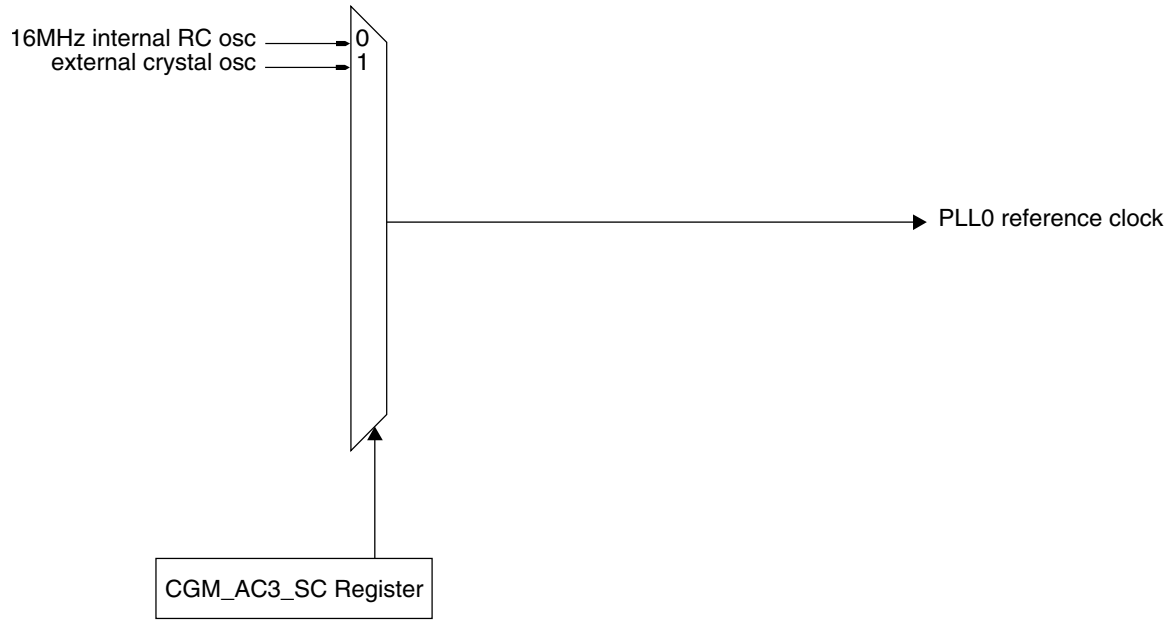


Figure 25-59. MC_CGM auxiliary clock 3 generation overview

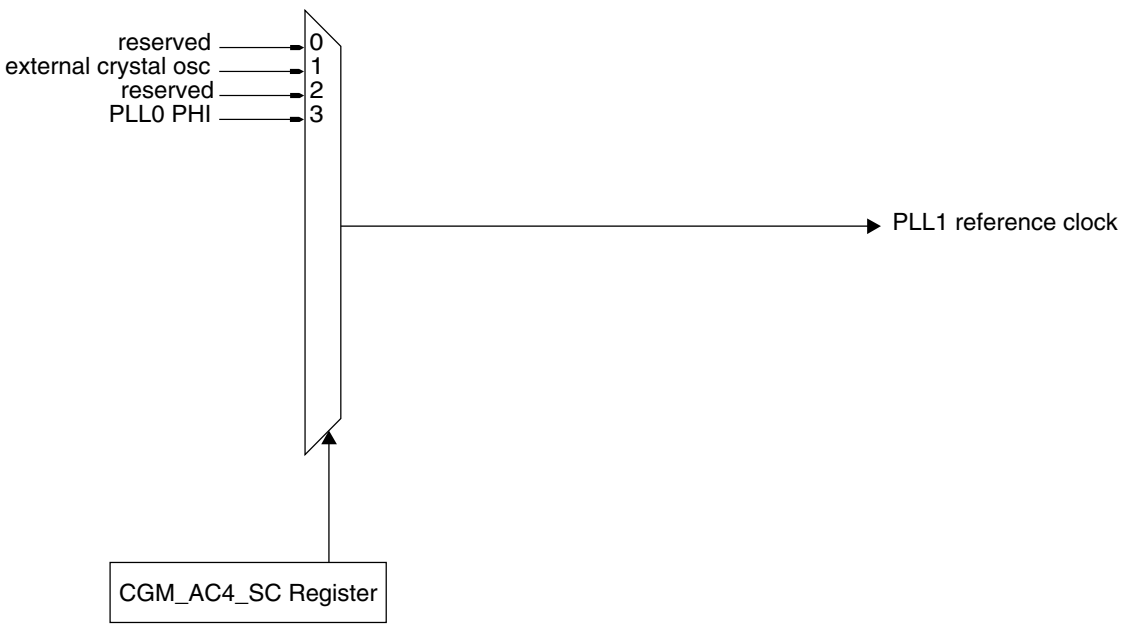


Figure 25-60. MC_CGM auxiliary clock 4 generation overview

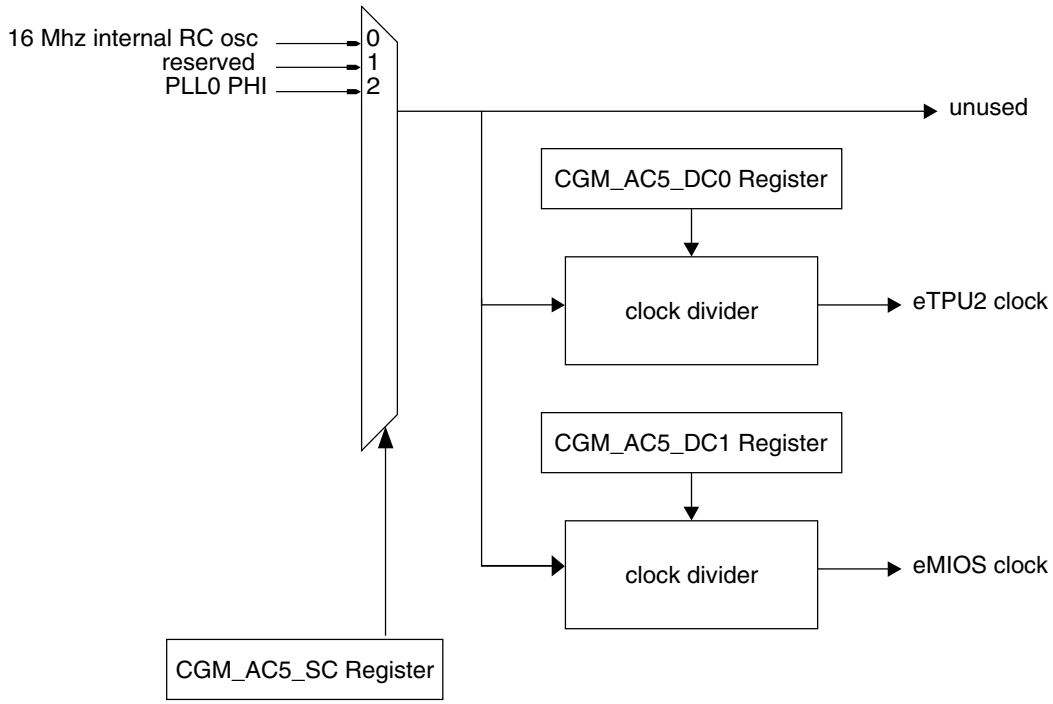


Figure 25-61. MC_CGM auxiliary clock 5 generation overview

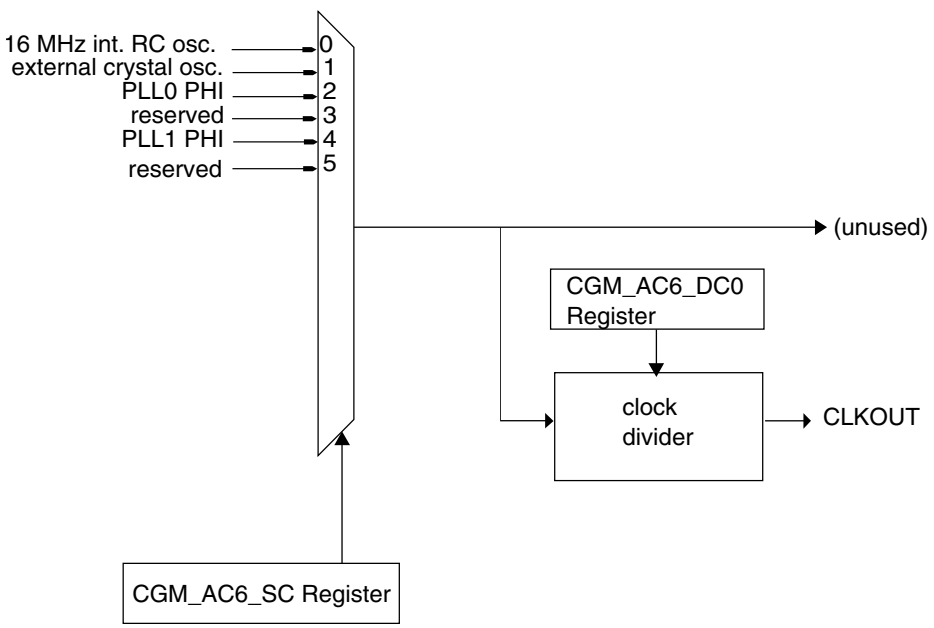


Figure 25-62. MC_CGM auxiliary clock 6 generation overview

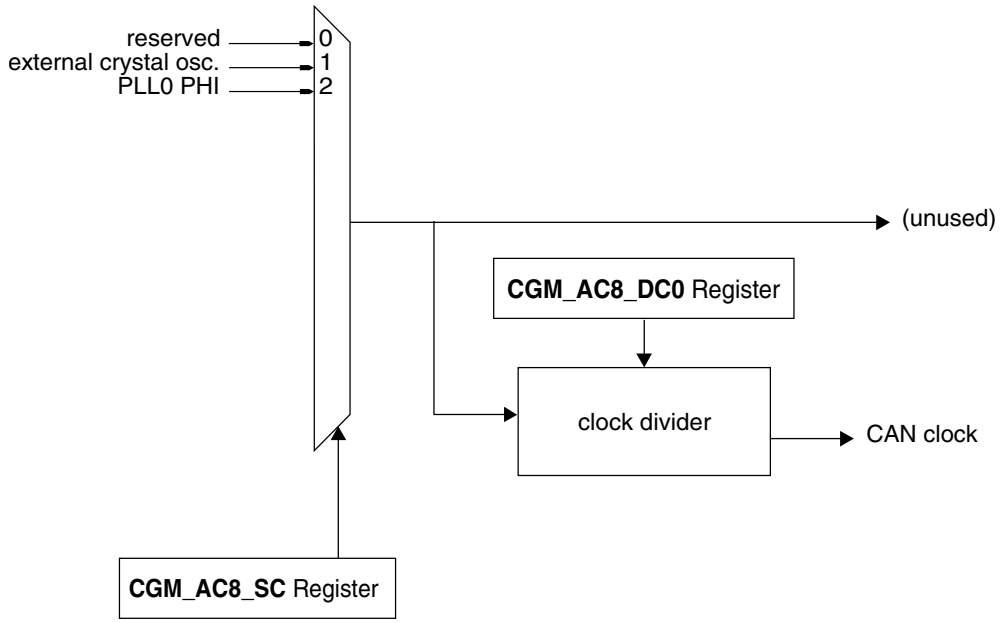


Figure 25-63. MC_CGM auxiliary clock 8 generation overview

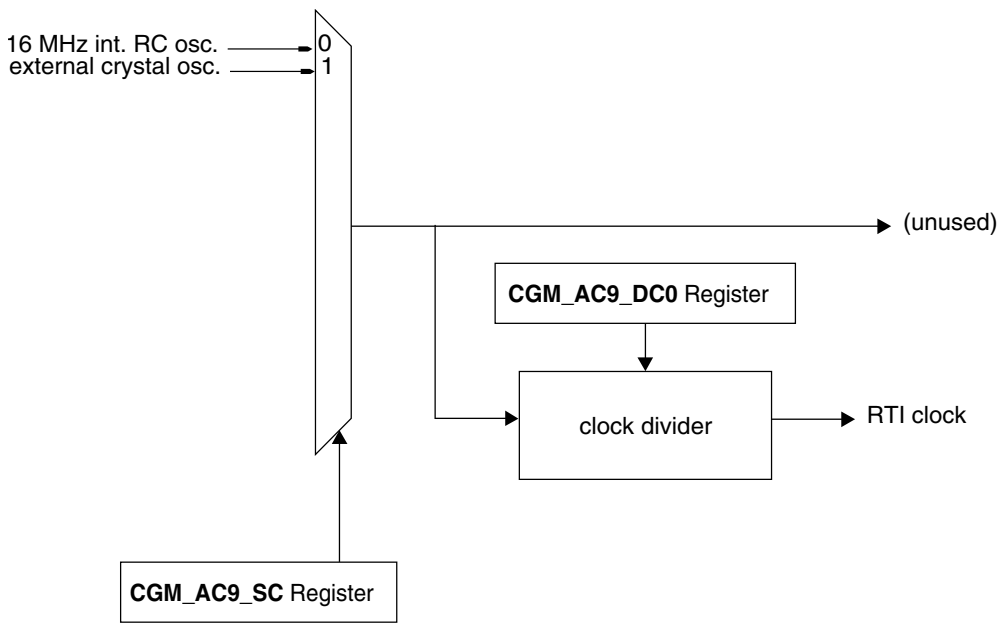


Figure 25-64. MC_CGM auxiliary clock 9 generation overview

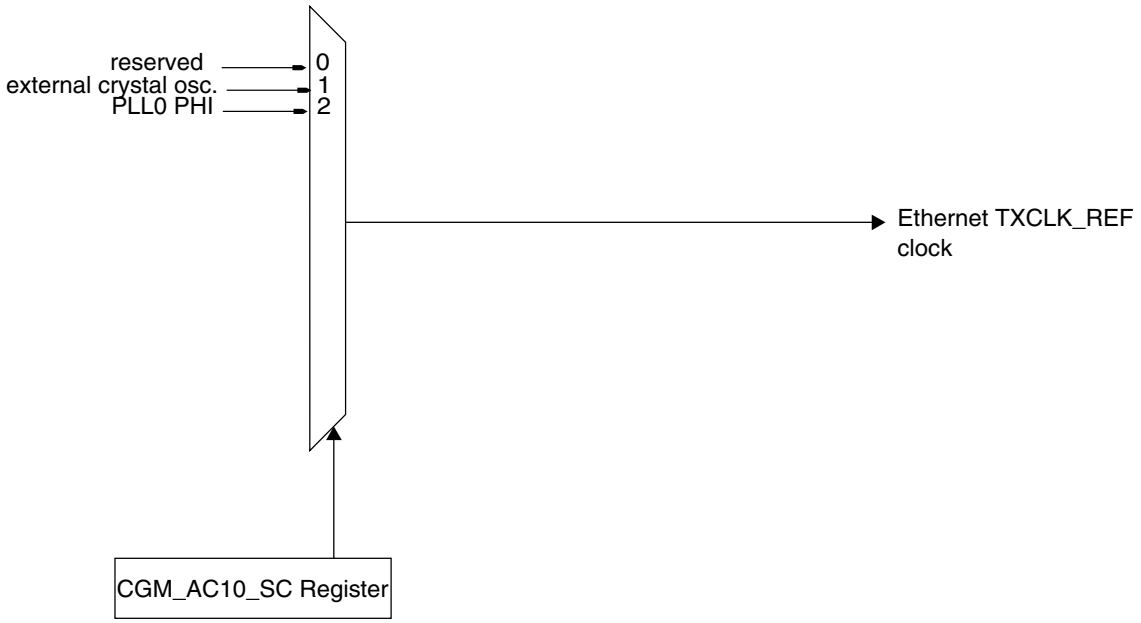


Figure 25-65. MC_CGM auxiliary clock 10 generation overview

25.4.2.1 Auxiliary Clock Dividers

The MC_CGM generates the following derived clocks

- peripheral clock - controlled by the CGM_AC0_DC0 register
- sigma delta ADC clock - controlled by the CGM_AC0_DC1 register
- SAR ADC clock - controlled by the CGM_AC0_DC2 register
- DSPI clock 0 - controlled by the CGM_AC0_DC3 register
- DSPI clock 1 - controlled by the CGM_AC0_DC4 register
- BD LFAST PLL clock - controlled by the CGM_AC1_DC0 register
- SENT clock - controlled by AC2_DC0 register
- eTPU clock - controlled by AC5_DC0 register
- eMIOS clock - controlled by AC5_DC1 register
- CLKOUT - controlled by AC6_DC0 register
- FLEXCAN clock - controlled by AC8_DC0 register
- RTI clock - controlled by AC9_DC0 register
- TXCLK ref clock - controlled by AC10_DC0 register

25.4.3 Dividers Functional Description

Dividers are used for the generation of divided system and peripheral clocks. The MC_CGM has the following control registers for built-in dividers:

- System Clock Divider 0 Configuration Register (CGM_SC_DC0)
- Auxiliary Clock 0 Divider 0 Configuration Register (CGM_AC0_DC0)
- Auxiliary Clock 0 Divider 1 Configuration Registers (CGM_AC0_DC1)
- Auxiliary Clock 0 Divider 2 Configuration Registers (CGM_AC0_DC2)
- Auxiliary Clock 0 Divider 3 Configuration Registers (CGM_AC0_DC3)
- Auxiliary Clock 0 Divider 4 Configuration Registers (CGM_AC0_DC4)
- Auxiliary Clock 1 Divider 0 Configuration Register (CGM_AC1_DC0)
- Auxiliary Clock 1 Divider 1 Configuration Register (CGM_AC1_DC1)
- Auxiliary Clock 2 Divider 0 Configuration Register (CGM_AC2_DC0)
- Auxiliary Clock 5 Divider 0 Configuration Register (CGM_AC5_DC0)
- Auxiliary Clock 5 Divider 1 Configuration Register (CGM_AC5_DC1)
- Auxiliary Clock 6 Divider 0 Configuration Register (CGM_AC6_DC0)
- Auxiliary Clock 8 Divider 0 Configuration Register (CGM_AC8_DC0)
- Auxiliary Clock 9 Divider 0 Configuration Register (CGM_AC9_DC0)
- Auxiliary Clock 10 Divider 0 Configuration Register (CGM_AC10_DC0)

The reset value of all counters is '1'. If a divider has its **DE** bit in the respective configuration register set to '0' (the divider is disabled), any value in its **DIV** field is ignored.

25.4.3.1 Punch-Out Divider Details

All system clock dividers FXBAR, SXBAR, and AIPS are punch-out dividers. Punch-out dividers are integer dividers and are based on a counter/comparator pair. An enable signal generated by this pair is asserted when the counter reaches the value '0' and is deasserted when its value is non-zero. The consequence of this design is that the duty cycle of output clock generated by this pair is not 50%.

A punch-out divided clock can be considered synchronous to other punch-out and 50% duty cycle divided clocks which are sourced by the same clock source selector, but only the system clock dividers are aligned.

Table 25-54. MC_CGM Punch-Out Divider Characteristics

Characteristic	Value
min frequency	$F_{\text{"divider source clock"}} \div$ "division factor"
max frequency	$F_{\text{"divider source clock"}} \div$ "division factor"
duty cycle	"high period of divider source clock" ÷ ("period of divider source clock" × "division factor")

Table continues on the next page...

Table 25-54. MC_CGM Punch-Out Divider Characteristics (continued)

Characteristic	Value
jitter due to divider	0

25.4.3.2 50% Duty Cycle Divider Details

All auxiliary dividers are 50% duty cycle dividers except the one for DSPIM1 and M2. 50% duty cycle dividers generate a real divided clock. The division factor is always an integer but is not restricted to even numbers. The rising edge of the divided clock is always synchronous to the rising edge of the divider source clock, but the falling edge is synchronous to the rising edge or the falling edge depending on whether the division factor is even or odd, respectively.

A 50% duty cycle divided clock can be considered synchronous to other punch-out and 50% duty cycle divided clocks which are sourced by the same clock source selector, but only the system clock dividers are aligned.

Table 25-55. MC_CGM 50% Duty Cycle Divider Characteristics

Characteristic	Value
min frequency	$F_{\text{"divider source clock"}} \div$ "division factor"
max frequency	$F_{\text{"divider source clock"}} \div$ "division factor"
duty cycle	50%[1]
jitter due to divider	0

1. for a division factor of 1, the duty cycle = the duty cycle of the divider source clock

For system clock dividers, $i = 16 \times m$, where m = the divider number. For auxiliary clock dividers, $i = 96 \times n + 16 \times m$, where n = the auxiliary clock number and m = the divider number.

25.4.3.3 Fractional Divider Details

Fractional dividers generate a real divided clock of AUX0_3 (DSPIM0, M2). The division factor is always an integer divided by either 1, 10, 100, or 1,000. Both the rising and the falling edges of the divider source clock are used to generate the rising and falling edges of the divided clock. Therefore, the rising edge of the divided clock may sometimes be synchronous to the rising edge and at other times to the falling edge. The same is true for the divided clock's falling edge.

A fractionally divided clock must be considered asynchronous to all other clocks.

Table 25-56. MC_CGM Fractional Divider Characteristics

Characteristic	Value
min frequency	$F_{\text{"divider source clock"}}$ ÷ "division factor rounded up to the nearest .0 or .5"[1]
max frequency	$F_{\text{"divider source clock"}}$ ÷ "division factor rounded down to the nearest .0 or .5"[1]
duty cycle	50% ± 0.25 × "period of divider source clock" [1]
jitter due to divider	≤ 0.25 × "period of divider source clock"[1]

1. assuming that the duty cycle of the divider source clock = 50%

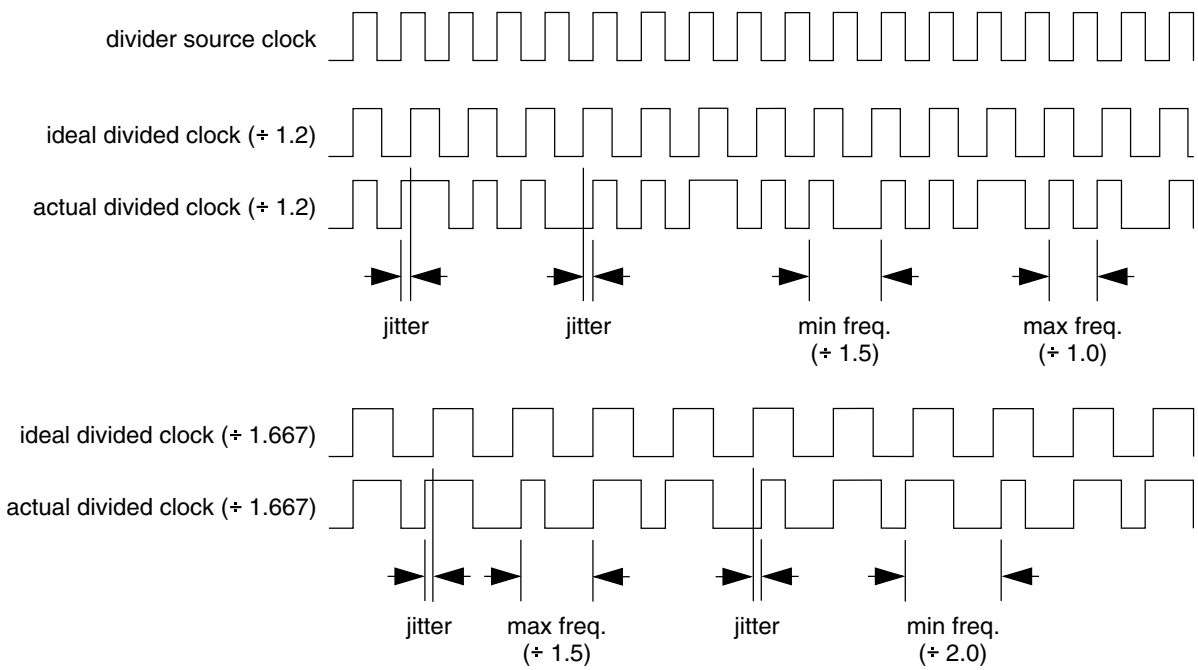


Figure 25-66. MC_CGM Fractional Clock Division Examples

25.4.3.4 Divider Update Synchronization

For the system clock and all the auxiliary clocks, all the associated dividers of the punch-out and 50% duty cycle types are synchronized to each other such that the rising edges of the lower frequency clocks are aligned with those of the higher frequency clocks. This, however, imposes limitations on the division factors which can be used. In order for the synchronization to work properly, each division factor must be selected such that its value is an integer multiple of each division factor that has a lower value.

The following table shows an example of a correct divider configuration.

Table 25-57. MC_CGM Example Clock Division Values Compatible with Divider Synchronization

Divider	Division Factor	Register Value
0	1	**CGM_xC_DC0[DIV]** = 0
1	2	**CGM_xC_DC1[DIV]** = 1
2	6	**CGM_xC_DC2[DIV]** = 5

In this case, clock divider 1 has a division factor greater than that of clock divider 0. Since 2 is an integer multiple of 1, its configuration is correct. Also, clock divider 3 has a division factor that is greater than those of both clock dividers 0 and 1. Since 6 is an integer multiple of 1 and 6 is an integer multiple of 2, its configuration is also correct.

The following table shows an example of an incorrect divider configuration.

Table 25-58. MC_CGM Example Clock Division Values Incompatible with Divider Synchronization

Divider	Division Factor	Register Value
0	1	**CGM_xC_DC0[DIV]** = 0
1	4	**CGM_xC_DC1[DIV]** = 3
2	6	**CGM_xC_DC2[DIV]** = 5

In this case, clock divider 1 has a division factor greater than that of clock divider 0. Since 4 is an integer multiple of 1, its configuration is correct. Also, clock divider 3 has a division factor that is greater than those of both clock dividers 0 and 1. Since 6 is not an integer multiple of 4, its configuration is incorrect.

NOTE

Configuring the clock dividers incorrectly will cause the divided clocks to become unsynchronized. This may lead to lost communication between the clock domains and/or the inability to further change the clock divider division factors without first disabling them or performing a reset.

25.4.3.5 Divider Update Triggering

There are two methods for updating the dividers associated with the system clock and each auxiliary clock: immediate and register triggered. Which method is used for which set of dividers is controlled by the **CGM_UPD_TYPE** register.

Regardless of which method is selected for a given group of dividers, once an update of one of the dividers of that group has been triggered, the associated bit in the **CGM_UPD_STAT** register is set. This bit is then cleared when all the dividers of that group have completed the update process and are running with the current configuration.

25.4.3.5.1 Immediate Divider Update

If a divider group has been configured to be updated immediately (i.e., the corresponding **CGM_UPD_TYPE** register bit = '0'), the configuration of a divider in that group will be applied immediately on the write to that divider's configuration register. The actual update will take some time to actually reach the divider output due to the clock domain crossing and edge alignment mechanisms. Timing examples are shown in the following figures.

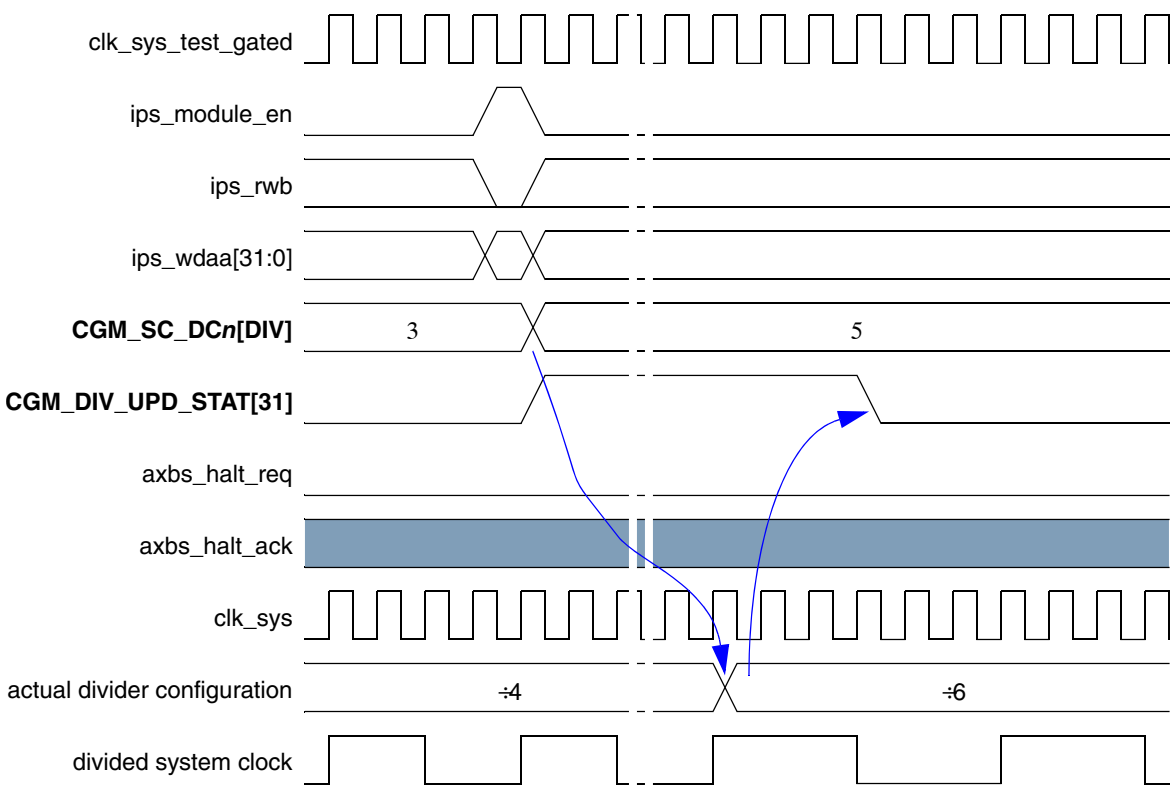


Figure 25-67. MC_CGM Immediate System Clock Divider Update Example

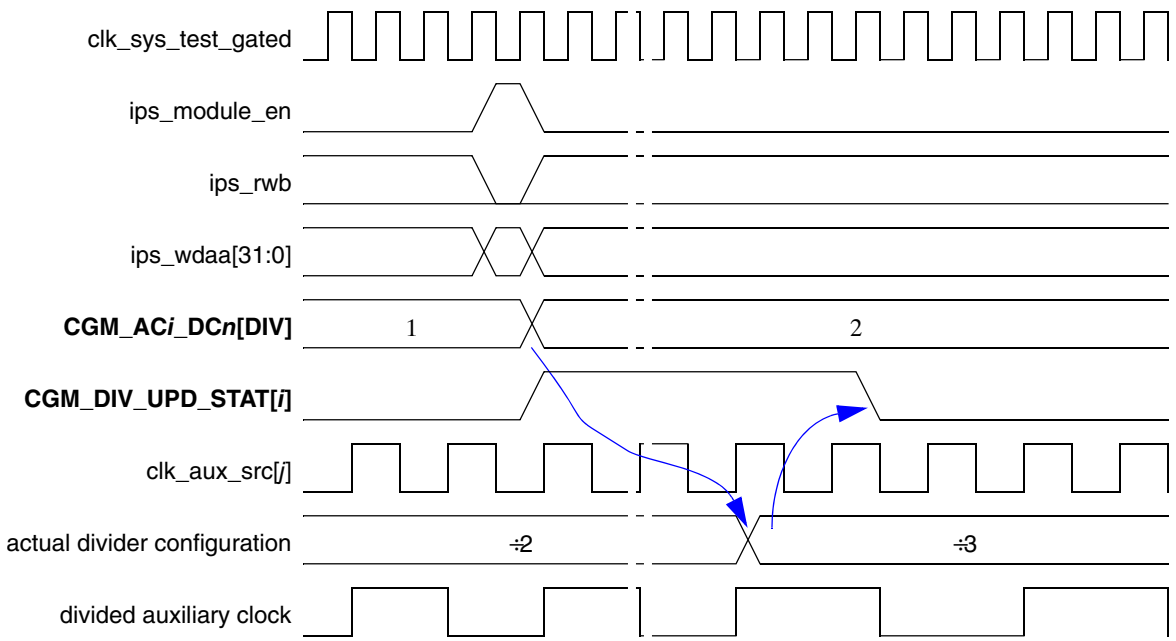


Figure 25-68. MC_CGM Immediate System Clock Divider Update Example

25.4.3.5.2 Software-Triggered Divider Update

If a divider group has been configured to be updated via a software trigger (i.e., the corresponding **CGM_UPD_TYPE** register bit = '1'), the configuration of a divider in that group will not be applied immediately on the write to that divider's configuration register. Instead, the configuratoin in the corresponding configuration register will be applied only when software writes any value to the **CGM_UPD_TRIG** register. The actual update will take some time to actually reach the divider output due to the clock domain crossing and edge alignement mechanisms.

In the case of the system clock dividers, if the SYS_DIV_RATIO_CHNG bit in the CGM_SC_DIV_RC register = `1', the crossbar switch is requested to halt its ongoing transactions, and only after the crossbar switch has acknowledged the request does the update of the system clock dividers take place. Once these updates have completed, the halt request to the crossbar is deasserted.

Timing examples are shown in the following figures.

NOTE

If halting the crossbar can cause a bus master to starve and thus lead to lost data or other problems, it is recommended to vary the system clock divider configurations without changing the ratios and to set CGM_SC_DIV_RC[SYS_DIV_RATIO_CHNG] to `0'.

NOTE

In order to ensure a clean clock divider configuration update, software must check that all ongoing updates have already completed (i.e., all relevant bits in the CGM_DIV_UPD_STAT register are `0') before triggering a new update via the CGM_DIV_UPD_TRIG register.

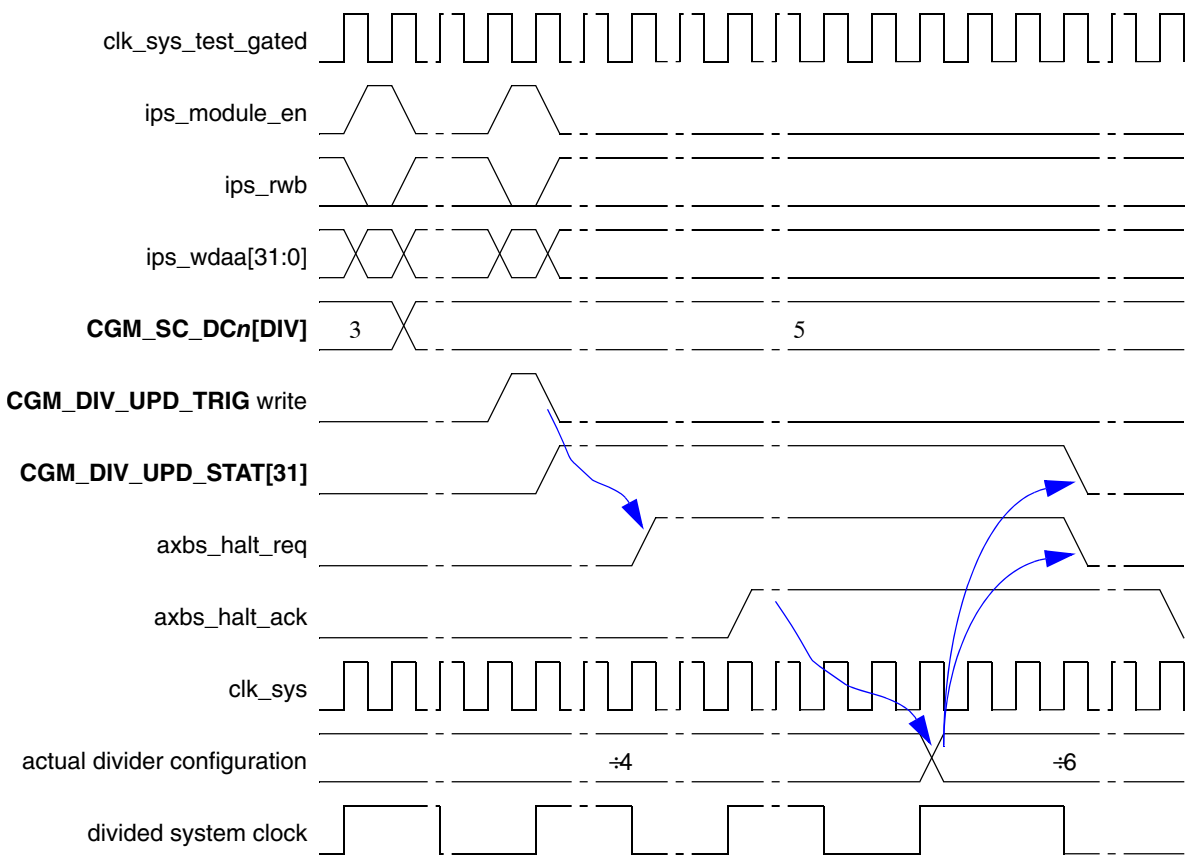


Figure 25-69. MC_CGM Software-Triggered System Clock Divider Update Example (ratio change)

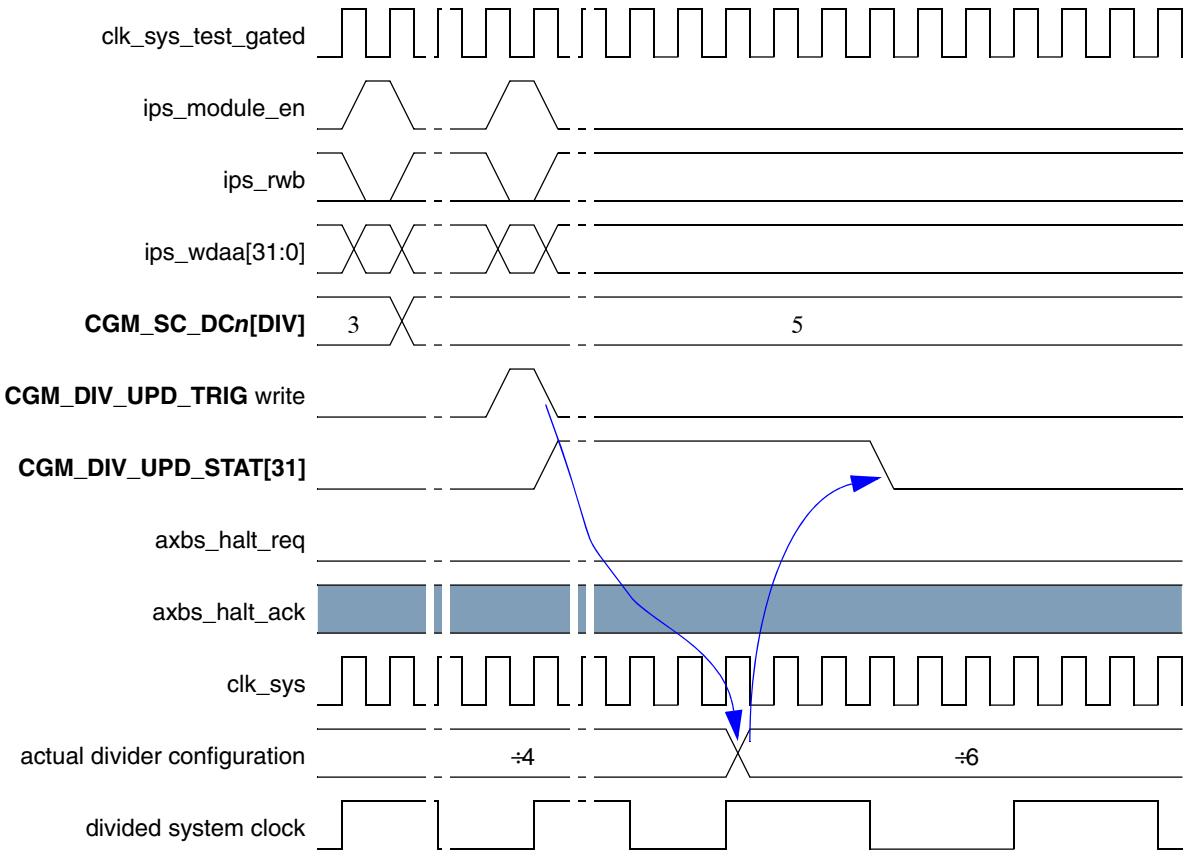


Figure 25-70. MC_CGM Software-Triggered System Clock Divider Update Example (no ratio change)

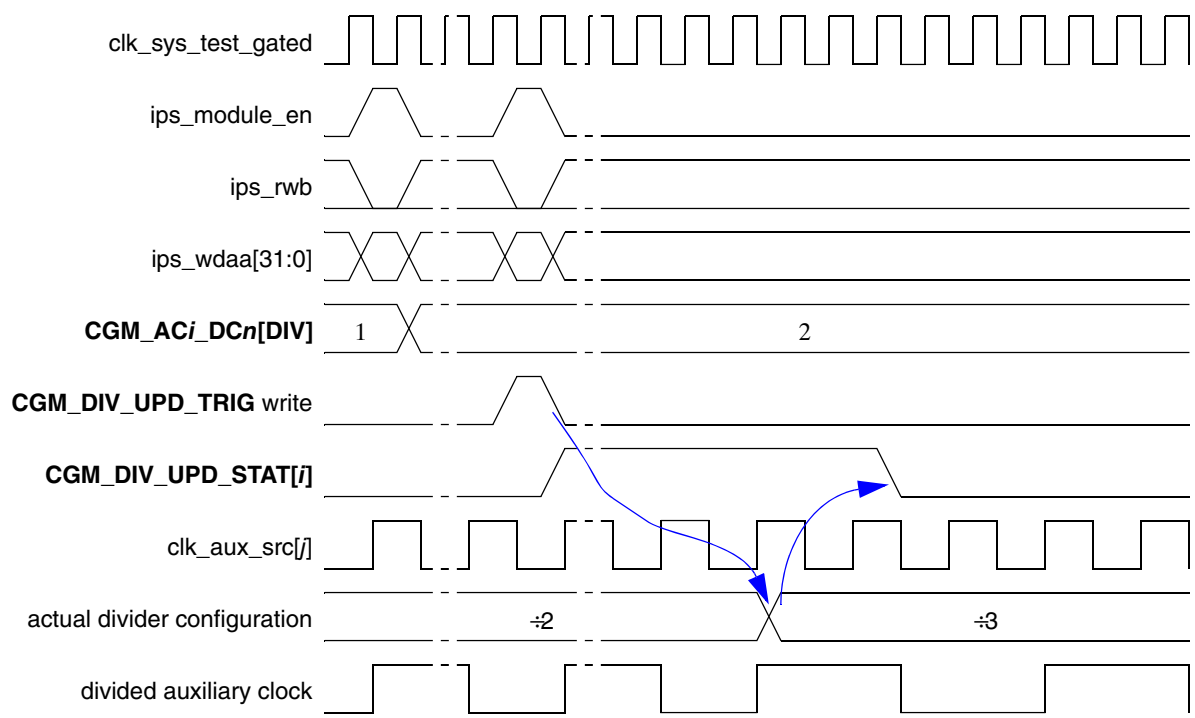


Figure 25-71. MC_CGM Software-Triggered Auxiliary Clock Divider Update Example

25.4.3.5.3 Reset-Triggered Divider Update

When a reset occurs as indicated by the assertion of the ipg_hard_async_reset_b input, all clock dividers are reconfigured to their default values, regardless of any other pending or ongoing divider configuration update.

In the system clock case, in order to prevent frequency overshoot and to allow the highest possible frequency during the device configuration part of the reset sequence, the dividers are updated in three steps as shown in the following figure.

1. safe_clock_busy deassertion edge after ipg_hard_async_reset_b assertion edge causes system clock dividers to switch to division factor = 1
2. sscm_done assertion causes system clock dividers to switch to the default division factor 0 (DIV1)
3. clk_sys_div_def_val_done is asserted when all system clock dividers have completed update to default division factors

The clk_sys_div_def_val_done output is used to indicate to the MC_RGM that the reset sequence can exit reset PHASE3, which ensures that ipg_hard_async_reset_b will not be deasserted until after the system clock dividers have completed updating to their default configurations.

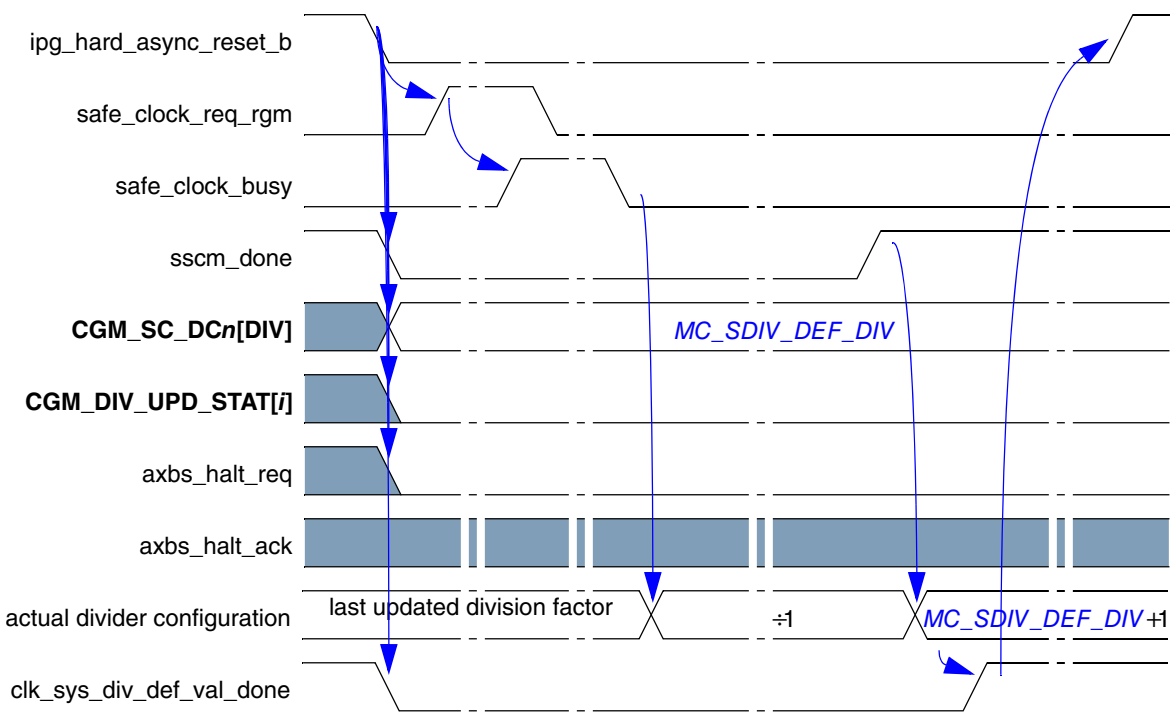


Figure 25-72. MC_CGM Reset-Triggered System Clock Divider Update Example

In the auxiliary clock case, this update takes place immediately on ipg_hard_async_reset_b assertion as shown in the following figure.

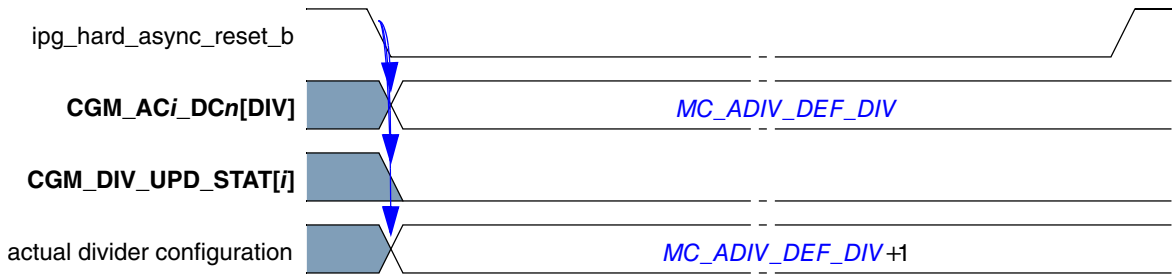


Figure 25-73. MC_CGM Reset-Triggered Auxiliary Clock Divider Update Example

Chapter 26
OSC Digital Interface (XOSC)

26.1 Introduction

The XOSC digital interface is used to control the on-chip oscillator (XOSC) and provide the register interface for the programmable features. Selection of the XOSC as a source clock is controlled by the Clock Generation Module (MC_CGM). The XOSC is enabled and disabled by configuring the Mode Entry Module (MC_ME).

The programmable features of the XOSC digital interface are as follows:

- Oscillator power-down control and status
- Oscillator startup delay
- Oscillator clock available interrupt
- Oscillator bypass mode control

26.2 Functional description

The XOSC digital interface provides control of the on-chip oscillator and the register interface for the mode and startup delay control. The oscillator provides an output clock that is used as an input to the PLL and can be selected as the reference clock for many of the peripherals on the device.

The table below shows the possible configurations of the XOSC.

Table 26-1. XOSC configurations

OSC EN (from MC_ME)	XOSC_CTL[OSCBYP]	XTAL	EXTAL	XOSC Output	XOSC MODE
0	0	Crystal, or left opened	Crystal, or left opened	0	Power down, IDDQ
0	1	x	external clock	Undefined	Reserved, OSC Disabled
1	1	x	external clock	XTALOUT	Bypass, OSC Enabled
1	0	crystal	crystal	XTALOUT	Normal, OSC Enabled

26.2.1 Oscillator power-down control and status

Enabling and disabling of the oscillator is done in the Mode Entry (MC_ME) module (see "Mode Entry Module (MC_ME)" chapter). The ME_<*mode*>_MC[XOSCON] controls the power down of oscillator based on the current device mode (see "MC_ME Memory Map" table for available *modes*). The status of the XOSC is shown in the MC_ME_GS[S_XOSC].

26.2.2 Oscillator startup delay

If enabled, the analog portion of the oscillator is powered-up during Phase3 of the device reset sequence (pre-self-test sequence). In order to allow the oscillator to reach full amplitude before use, the internal counter (OSCCNT) is started when the device exits reset. The counter is driven by the oscillator output clock. When the counter reaches XOSC_CTL[EOCV] × 512, a signal is sent to the MC_ME module, allowing a mode switch to the oscillator (when used as a system clock or an input to the PLL).

The default value for this field after reset depends on implementation and can be found in the XOSC Device Configuration. After reset, software can clear or change the EOCV field in order to shorten or lengthen the delay until the oscillator is ready.

Note

It is advised that if the EOCV value is modified it should be done as early as possible in user code. This will ensure that the new value will be used before OSCCNT reaches the count value.

26.2.3 Oscillator clock available interrupt

An interrupt can be generated when the count value is reached (OSCCNT = XOSC_CTL[EOCV] × 512). The XOSC_CTL[I_OSC] will be set and an interrupt will be generated if the interrupt mask is set (XOSC_CTL[M_OSC] = 1).

26.2.4 Oscillator bypass mode

The oscillator circuit can be bypassed by setting XOSC_CTL[OSCBYP]. This field can only be set by software and is only cleared by a system reset. In bypass mode, the oscillator is disabled and the input clock on the EXTAL pins is level-shifted and driven to the logic on the device. The bypass configuration is independent of the power-down mode of the oscillator.

NOTE

The external oscillator must be running before the XOSC is turned off. If XOSC is being turned on, the status will be updated only after the clock starts toggling.

NOTE

Power down from the MC_ME will have no effect on the oscillator. The bypass clock will continue running in the device. It is ensured that mode transitions from the MC_ME complete successfully by providing bypass information to the MC_ME.

26.3 Memory map and register definition

XOSC memory map

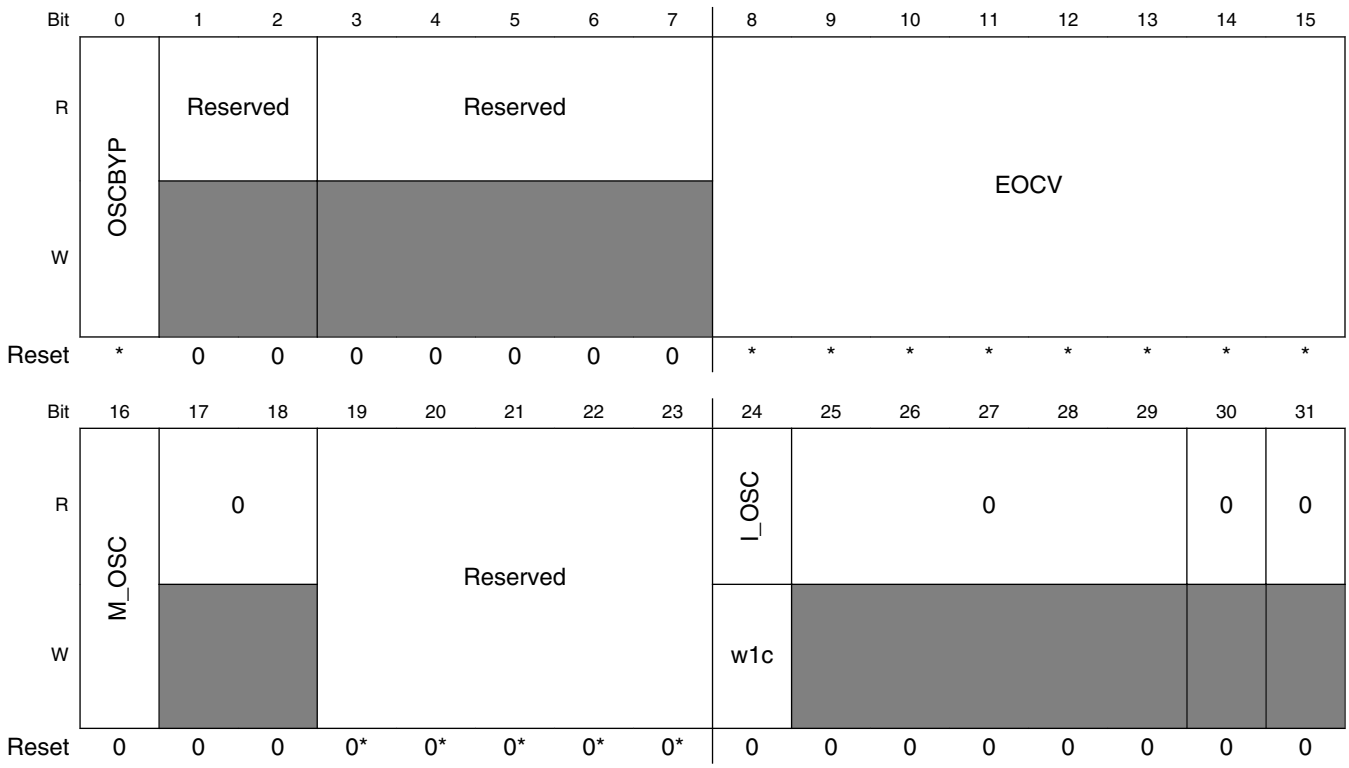
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	XOSC Control Register (XOSC_CTL)	32	R/W	See section	26.3.1/772

26.3.1 XOSC Control Register (XOSC_CTL)

NOTE

The XOSC_CTL is only writable in supervisor mode.

Address: 0h base + 0h offset = 0h



* Notes:

- Reserved field: This field can be written at any time, but writes have no effect. Reads return the previously written value.
- M_OSC field: See Clocking chapter for reset value.
- EOCV field: See Clocking chapter for reset value.
- OSCBYP field: See Clocking chapter for reset value.

XOSC_CTL field descriptions

Field	Description
0 OSCBYP	Crystal Oscillator bypass This bit specifies whether the oscillator should be bypassed or not. Software can only set this bit. System reset is needed to reset this bit. 0 Oscillator output is used as root clock. 1 EXTAL is used as root clock.
1–2 Reserved	This field is reserved.

Table continues on the next page...

XOSC_CTL field descriptions (continued)

Field	Description
3–7 Reserved	This field is reserved.
8–15 EOCV	End of Count Value These bits specify the end of count value to be used for comparison by the oscillator stabilization counter OSCCNT after reset or whenever it is switched on from the off state. This counting period ensures that external oscillator clock signal is stable before it can be selected by the system. When oscillator counter reaches the value EOCV × 512, oscillator available interrupt request is generated. The OSCCNT counter is kept under reset if oscillator bypass mode is selected.
16 M_OSC	Crystal oscillator clock interrupt mask This bit masks the I_OSC interrupt bit. 0 Crystal oscillator clock interrupt is masked. 1 Crystal oscillator clock interrupt is enabled.
17–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 Reserved	This field is reserved. This field can be written at any time, but writes have no effect. Reads return the previously written value.
24 I_OSC	Crystal oscillator clock interrupt This bit is set by hardware when OSCCNT counter reaches the count value EOCV x 512. It is cleared by software by writing 1. 0 No oscillator clock interrupt occurred. 1 Oscillator clock interrupt pending.
25–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Chapter 27
IRCOSC Digital Interface

27.1 Introduction

The Internal RC Oscillator Digital Interface (IRCOSC) controls the internal 16 MHz RC oscillator system. This oscillator system includes the main internal RC oscillator (MRC), as well as an internal temperature sensor and an internal voltage regulator used to compensate external variations in temperature and voltage. It also provides access to trimming information used for safety implementations.

27.2 Functional description

The IRCOSC provides a high frequency (f_{MRC}) clock with a nominal frequency of 16 MHz. After deassertion of reset, the output clock is available after a short time (see the Data Sheet for timing details) which is required for stabilization. The IRCOSC status is updated in the MC_ME_GS[S_IRC] field (see the "Mode Entry Module MC_ME" chapter).

During a power on reset sequence the following are initialized from TEST flash memory:

- Native trimming bits
- Fine trimming bits
- TSENS code bits

The IRCOSC output frequency can be trimmed by configuring IRCOSC_CTL[USER_TRIM[4:0]]. The USER_TRIM bits can be programmed to modify internal capacitor/resistor values to match output clock frequency with δf_{var_SW}, as defined in the Data Sheet.

27.3 Memory map and register definition

IRCOSC memory map

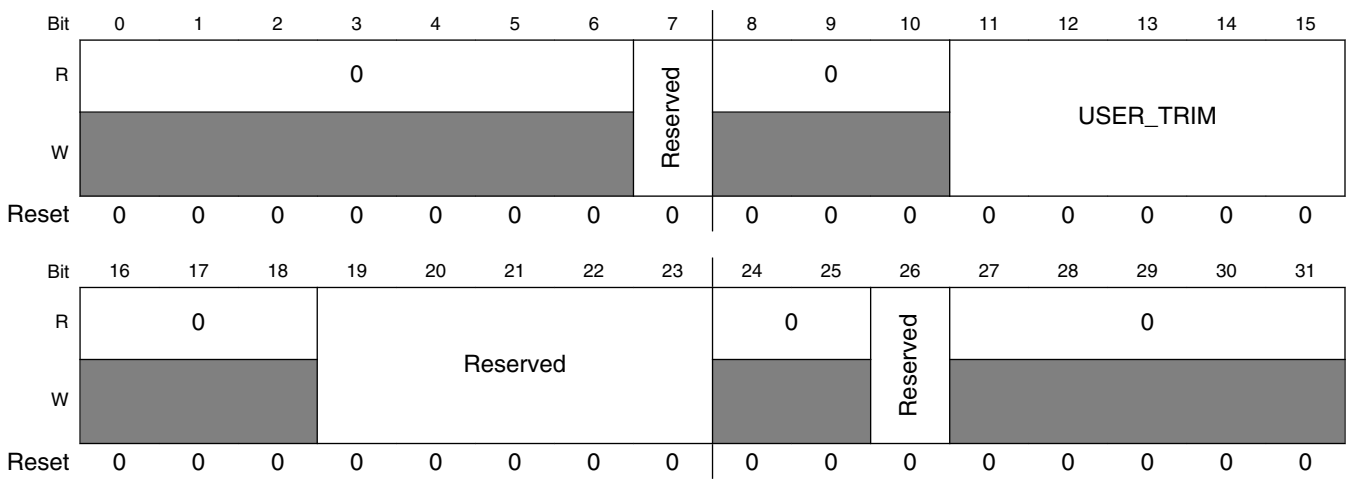
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	IRCOSC Control Register (IRCOSC_CTL)	32	R/W	See section	27.3.1/776

27.3.1 IRCOSC Control Register (IRCOSC_CTL)

The IRCOSC_CTL contains user programmable parameters.

IRCOSC_CTL is writable only in supervisor mode.

Address: 0h base + 0h offset = 0h



IRCOSC_CTL field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This bit can be written with any value, but writes are ignored. A read returns last written value.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 USER_TRIM	User trimming bits with respect to nominal factory frequency The MSB of the USER_TRIM bits is used to determine whether the frequency will be increased or decreased. If MSB = 0 then a change in USER_TRIM[3:0] will decrease the frequency, and likewise, if MSB = 1 then a change in USER_TRIM[3:0] will increase the frequency. Frequency trimming calculation shows how RCOSC_CTL[USER_TRIM] field is used to trim the IRCOSC frequency.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

IRCOSC_CTL field descriptions (continued)

Field	Description
19–23 Reserved	This field is reserved. This field can be written at any time, but writes have no effect. Reads return the previously written value.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 Reserved	This field is reserved. This field may sometimes have the value of 1, and can be cleared by writing a 1 to this bit. However, this field has no effect on the IRCOSC or chip behavior.
27–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

27.3.2 Frequency trimming calculation

Table 27-3. IRCOSC_CTL[USER_TRIM] frequency trimming calculation

USER_TRIM[4:0] value	Frequency
....	....
10011	$f + (3 \cdot \delta f_{TRIM})$
10010	$f + (2 \cdot \delta f_{TRIM})$
10001	$f + \delta f_{TRIM}$
10000	f
00000	f
00001	$f - \delta f_{TRIM}$
00010	$f - (2 \cdot \delta f_{TRIM})$
00011	$f - (3 \cdot \delta f_{TRIM})$
....	....

NOTE

See the data sheet for δf_{TRIM} value and min/max frequency variations.

Chapter 28
RAM Controller (PRAMC)

28.1 Introduction

This section provides an overview of the Platform RAM Controller. The RAM controller acts as an interface between the system bus (AHB-Lite 2.v6) and the integrated system RAM. It converts the protocols between the system bus and the dedicated RAM array interface.

The RAM controller supports two 64-bit AHB interfaces and a 64-bit RAM array interface. The primary AHB port provides a connection to the platform crossbar for direct RAM access from the various crossbar masters. The backdoor AHB port provides a connection from the flash controller to facilitate calibration overlay access. Shown below in Figure 28-1 is a simplified block diagram depicting the position of the RAM controller within a typical platform architecture.

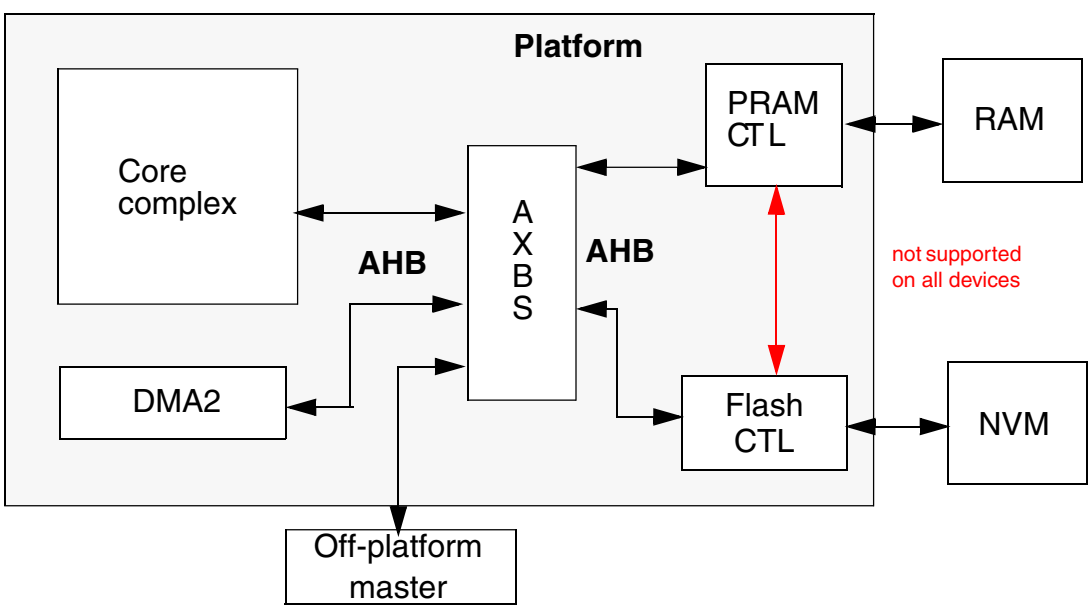


Figure 28-1. Simplified platform block diagram

The following list summarizes the key features of the RAM controller:

- System bus supports 64-bit data + 8-bit ECC AHB interfaces
- Array interface supports a 64-bit data + 8-bit ECC interface
- Late-write support via 64-bit data + 8-bit ECC storage buffer to support single-cycle write accesses
- Configurable read access timing (zero or one wait-state programmable) allowing use in large range of frequency targets
- Read-modify-write operation to support array write size less than a doubleword

The address path of the RAM controller contains a mux that chooses among the addresses presented on the system bus for a read request, either the address from the temporary holding register or the address from the late-write buffer. The temporary holding register contains the address which was presented during the AHB address phase of the write request. The request is stalled from being presented to the RAM by one cycle in order to present simultaneously the address and associated write data, which is not valid until the AHB dataphase. The late-write buffer holds write requests which are delayed to facilitate single-cycle response on the system bus.

The read datapath contains a mux that chooses the source of the read data to be presented on the system bus on a read request. In the event that a read request matches the contents of the late-write buffer, a RAM access is not required and the late-write buffer contents are selected onto the read databus. Otherwise, the read data is a result of a RAM access. Along with the read and write data, the corresponding ECC codewords traverse the entire datapath of the RAM controller, including the late-write storage buffer, to support end-to-end ECC (e2eECC) coverage.

The write datapath contains the mux that chooses the source of the write data as well as the control logic for generating read-modify-write operations on writes that are less than 64 bits in size. A write operation can be performed to empty the contents of the late-write buffer. In the case of back-to-back writes, the write may be forced to bypass the late-write buffer and instead be stored directly, precisely in the RAM.

28.2 SRAM controller memory map and register definition

The RAM controller module provides an IPS programming model mapped to a standard on-platform peripheral slot. The programming model can only be referenced using a 32-bit (word) access. Attempted references using different access sizes or to undefined (reserved) addresses generate an IPS error termination.

NOTE

Attempted updates to the PRAMC programming model while a PRAMC operation is in progress will result in non-deterministic behavior. Software must be architected to avoid this scenario by ensuring that PRAMC configuration changes are made only during system boot or when only one master is enabled. In multi-core devices, update the PRAMC configuration only when one core is active and no other masters, e.g., DMA or communications modules, are enabled. Move any instruction execution or memory references outside the system RAM while updating the PRAMC configuration, e.g., to the core local memory space.

PRAMC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Platform RAM Configuration Register 1 (PRAMC_PRCR1)	32	R/W	0000_0200h	28.2.1/781

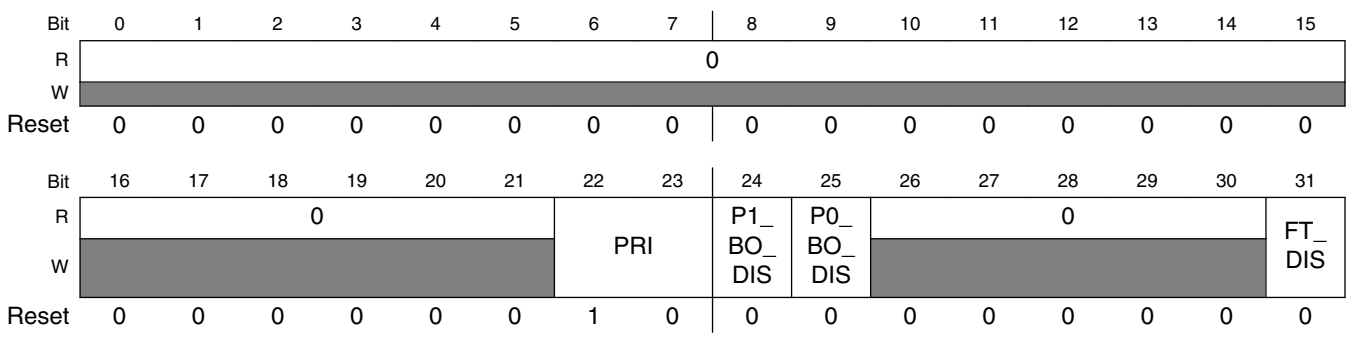
28.2.1 Platform RAM Configuration Register 1 (PRAMC_PRCR1)

The Platform RAM Configuration Register 1 (PRCR1) is used to specify operation of the RAM controller.

NOTE

This register is accessible only in supervisor mode. Accesses in user mode return a transfer error.

Address: 0h base + 0h offset = 0h



PRAMC_PRCR1 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PRAMC_PRCR1 field descriptions (continued)

Field	Description
22–23 PRI	AHB port arbitration mode. Use this register to select the AHB port arbitration mode. 00 Round Robin arbitration is selected. 01 Port p0 has priority over port p1. 10 Port p1 has priority over port p0. 11 Reserved.
24 P1_BO_DIS	Port p1 read burst optimization disable. **NOTE:** The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to the PRAMC controller clock frequency. 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response when PRCR1[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response when PRCR1[FT_DIS]=1
25 P0_BO_DIS	Port p0 read burst optimization disable. **NOTE:** The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to the PRAMC controller clock frequency. 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response when PRCR1[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response when PRCR1[FT_DIS]=1
26–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 FT_DIS	Flow through disabled. This field defines the AHB response on reads. The state of this field has no impact on the response latency on writes. This bit is cleared by hardware reset. **NOTE:** Do not change the FT_DIS bit value while accessing system RAM. Relocate code programming the FT_DIS bit to another memory area, e.g., local core memory. 0 RAM read data is passed directly to the system bus, incurring no additional latency 1 RAM read data is registered prior to returning on the system bus, incurring 1 extra cycle of latency

28.3 Functional description

This section describes the functions of the RAM controller.

28.3.1 Read and Write operations

The RAM controller processes read and write requests to on-chip RAM and provides a register interface for access to performance tuning functions.

NOTE

The RAM controller register interface is accessible only in supervisor mode. Accesses in user mode return a transfer error.

28.3.1.1 Reads

Read transfers, of any size, can be configured to complete with a zero or one additional wait state response on the system bus.

28.3.1.2 Optional read wait-state

The RAM controller can be optionally programmed to register RAM read data prior to returning it on the system bus. Setting the PRCRx[FT_DIS] field inserts a register in the read data path for use when operating the controller at high frequencies. The state of PRCRx[FT_DIS] field has no effect on writes.

A random, initial access will take 2 clock cycles (2T) to complete if PRCRx[FT_DIS]=0, and a WRAP4 burst will have an access time of 2-2-2-2. A random, initial access will take 3 clock cycles (3T) to complete if PRCRx[FT_DIS]=1, and a WRAP4 burst will have an access time of 3-2-2-2.

NOTE

The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles

depending on RAM speed relative to the PRAMC controller clock frequency.

28.3.2 Writes

This section discusses various write operations of the RAM controller.

28.3.2.1 64-bit writes

Aligned 64-bit writes execute in a single AHB data phase cycle, allowing for zero wait states on back-to-back writes of these sizes. If, during the data phase of a write, a read is requested on the AHB, the write is registered in the late-write buffer, allowing the read to take place without a wait state. The valid buffered or late-write data is stored on the next available array address phase.

Back to back 64-bit writes execute slightly differently whereby the first write bypasses the late-write buffer. Rather, the write data is stored directly to the array in the same cycle in which it is valid on the AHB.

28.3.2.2 Less than 64-bit writes

Writes of size less than 64 bits incur a read-modify-write action as a consequence of the ECC coding scheme's 64-bit granularity. In the case of a read-modify-write action, the RAM controller performs SEC/DED on the read data. The write data is merged into the appropriate byte lanes along with the potentially corrected read data. A new codeword is generated based on the newly formed doubleword. The newly formed doubleword and its associated checkbits are subsequently written to the RAM. Figure 28-3 provides details on the read-modify-write datapath. For details on the ECC coding scheme, refer to Reliability considerations.

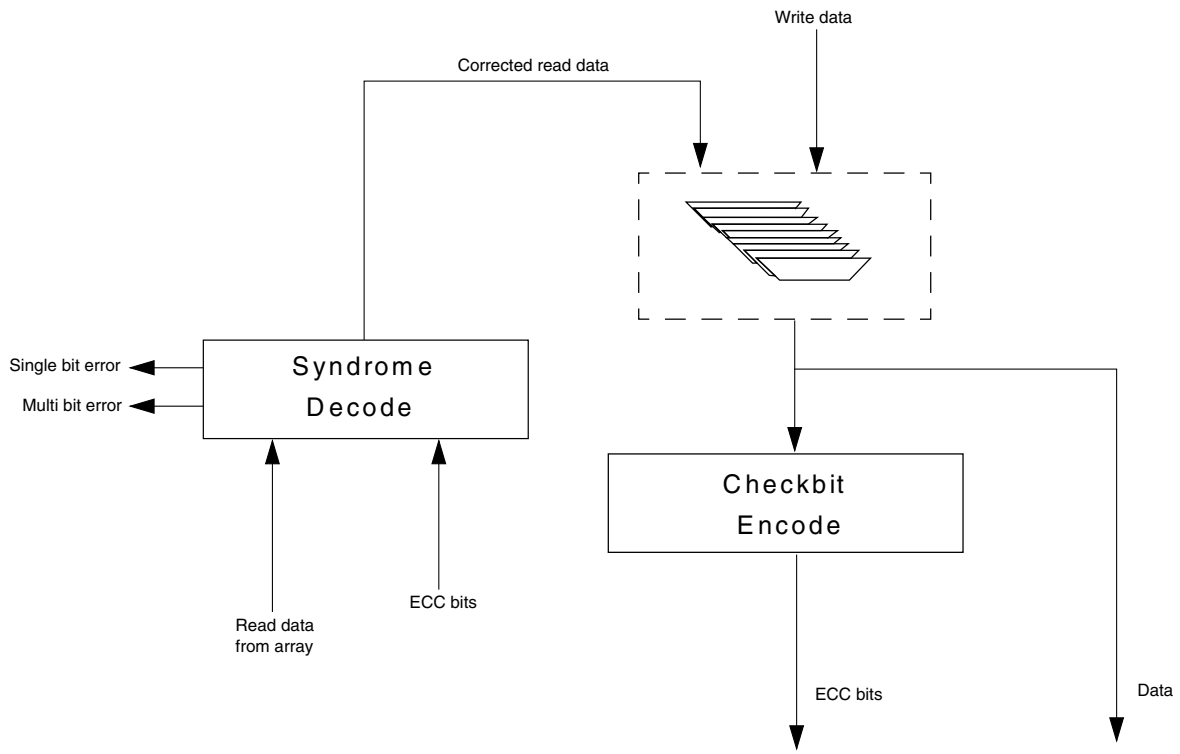


Figure 28-3. Read-modify-write datapath

On a read-modify-write operation, the array read data is registered after it is decoded and before it is merged with the AHB write data. Therefore, writes of size less than 64 bits require the insertion of one wait state before the data phase can be completed.

28.3.2.3 Unaligned writes

The RAM controller is compliant with the AMBA-AHB2.v6 Extensions specification with regard to byte strobes. The size of the transfer is sufficient to cover all bytes being written and covers more bytes in the case of an unaligned transfer. The address of the transfer is rounded down to the nearest boundary of the size of the transaction.

NOTE

Unaligned writes always generate read-modify-write operations in the RAM controller in order to preserve the validity of the ECC codeword.

28.4 Initialization/application information

It is essential that each memory address be written to a known value before it is read, to initialize the ECC. This includes reads generated from the read-modify-write operation which occurs when a write transfer of less than 64 bits or an unaligned write is requested. Without writing an address to a known value first, a read from this address will most likely generate an uncorrectable ECC event.

One possibility for initializing PRAMC memory space is to use a stored 64-bit word instruction such as Store Multiple Word (e_stmw) in Power Architecture VLE.

28.5 Reliability considerations

This section discusses reliability considerations of the RAM controller.

28.5.1 Hsiao ECC algorithm

The e2eECC scheme implements a single-error correction, double-error detection (SECDED) code using the so-called Hsiao odd-weight column criteria. These codes are named for M.Y. Hsiao, an IBM researcher who published extensively in the early 1970s on SECDED codes better suited for implementation in protecting (mainframe) computer memories than traditional Hamming codes.

The Hsiao codes are Hamming distance 4 implementations which provide the SECDED capabilities. The minimum odd-weight constraints defined by Hsiao are relatively simple in the resulting implementation of the parity check H matrix which defines the association between the data (and address) bits and the checkbits. They are:

1. There are no all-zeroes columns.
2. Every column is distinct.
3. Every column contains an odd number of ones, and hence is "odd weight".

In defining the H-matrix for this family of devices, these requirements from Hsiao were applied. Additionally, there are a variety of ECC codeword requirements associated with specific functional requirements associated with the system RAM that further dictated the specific column definitions. In any case, the resulting ECC is organized based on 64 data bits plus 29 address bits (the upper bits of the 32-bit address field minus the 3 bits which select the byte within the 64-bit (8-byte) data field.

The basic H-matrix for this (101, 93) code (93 is the total number of data bits, 101 is the total number of data bits (93) plus 8 checkbits) is shown in the following table. A '*' in the table indicates the corresponding data or address bit is XOR'ed to form the final checkbit value on the left. For 64-bit data writes, the table sections corresponding to D[63:32], D[31:0], and A[31:3] are logically summed (output of each table section is XOR'ed) together to the final value driven on the hwchkbit[7:0] outputs. Note that this table uses *the AHB bit numbering convention where bit[0] is the least significant bit*.

Table 28-3. RAM controller basic H-matrix definition

		XOR sum	Data Bit[1, 1]																																					
			Byte 7								Byte 6								Byte 5								Byte 4							
			63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Checkbits [7:0]	7		*				*					*			*		*	*		*	*	*	*				*	*				*	*	
	6				*	*			*	*					*		*		*				*				*			*	*			*
	5		*	*	*	*					*				*	*		*		*	*				*	*	*				*			
	4		*		*			*		*					*		*		*			*			*	*						*		
	3										*	*	*	*	*		*				*		*				*	*	*	*					
	2						*		*					*		*			*	*				*	*							*		*
	1			*		*		*		*	*	*	*	*						*		*		*		*						*		*
	0			*			*	*	*				*			*	*	*		*				*				*		*	*		*		
		XOR sum	Byte 3								Byte 2								Byte 1								Byte 0							
			31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Checkbits [7:0]	7						*		*				*		*				*	*			*	*			*				*	*		*
	6		*	*		*	*				*				*				*	*		*	*					*	*				*	*
	5				*	*			*	*	*			*	*	*					*		*	*		*								
	4		*		*	*				*		*	*				*	*	*			*			*	*				*		*		
	3		*	*	*			*						*	*		*	*			*				*		*	*	*			*		
	2			*		*	*	*	*	*	*	*	*			*					*					*		*		*	*	*		
	1			*						*		*			*	*	*			*				*				*			*	*	*	
	0			*		*		*		*				*				*				*				*					*	*		*	*
		XOR sum	Address Bit[2, 2]																																							
			31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Checkbits [7:0]	7			*		*		*		*	*	*		*	*	*		*	*	*		*		*		*	*	*		*		Unused	Unused	Unused
	6			*		*	*	*	*			*	*			*	*	*	*		*		*	*		*	*	*		*				
	5			*	*	*			*	*			*	*		*		*		*	*			*	*	*			*	*	*			
	4		*	*	*		*	*		*		*	*	*	*		*					*	*		*	*	*		*	*				
	3		*		*		*	*	*		*		*	*	*			*		*		*	*		*		*	*	*		*			
	2		*	*			*		*	*	*	*				*	*	*	*	*	*	*		*	*		*		*		*			
	1		*		*	*		*	*		*	*		*	*		*		*	*	*		*	*	*					*	*			
	0		*		*	*	*			*	*		*		*	*	*		*		*	*	*			*		*	*		*			

1. Bit numbering is AHB convention: bit 0 is LSB. D[7:0] corresponds to byte at address 0. D[63:56] corresponds to byte at address 7.

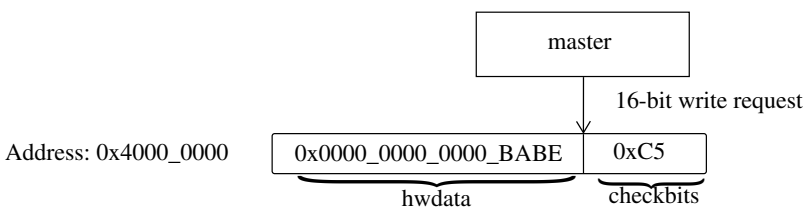
2. Bit numbering is AHB convention: bit 0 is LSB. D[7:0] corresponds to byte at address 0. D[63:56] corresponds to byte at address 7.

28.5.1.1 e2eECC considerations on less-than-64-bit write transactions

Recall writes of size less than 64-bit incur a read-modify-write action as a consequence of the ECC coding scheme 64-bit granularity. In the case of a read-modify-write action, the RAM controller performs SEC/DED on the read data. The RAM controller uses the following checkbit manipulation technique to ensure any faults in the PRAMC control logic associated with performing the read-modify-write action will ultimately be detectable by e2eECC:

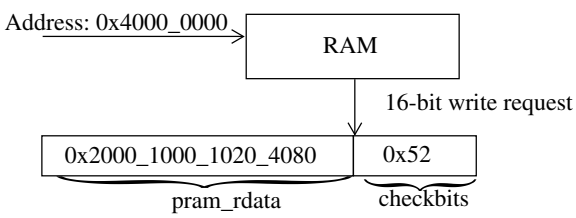
- On an 8-, 16-, or 32-bit write request, the write data is presented by the master on the AHB bus in the correct byte lanes, with the non-pertinent byte lanes zeroed out, and the associated checkbit is based on this "zero-padded" write data.

  Consider the following example whereby the master presents a 16-bit write request at address 0x40000000:

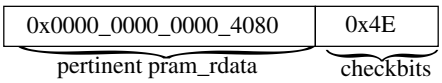


- The RAM controller detects that the request is a less-than-64-bit write request and initiates a read to address 0x40000000. An ECC check and potential single-bit correction is performed on the data returned from the RAM. In addition, the ECC event is reported to the Memory Error Management Unit (MEMU).

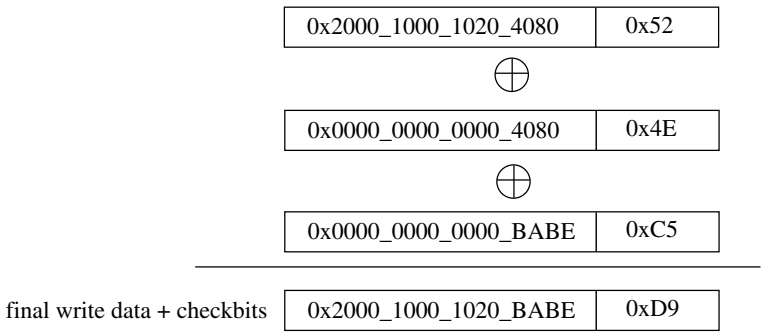
  If a non-correctable ECC event is detected, the AHB request is terminated with error on the system bus. In addition, the ECC event is reported to the MEMU.



- The RAM controller isolates the pertinent byte lanes of the read data by zero-padding the non-pertinent byte lanes and calculating the checkbit contribution associated with the read data in the byte lanes to be updated as a result of the write request.



- Merge the non-pertinent read data bytes with the write data in the appropriate byte lanes. Take the checkbit value that was stored in the RAM and remove the checkbit contribution associated with the read data in the byte lanes to be updated. Then introduce the checkbits associated with the write data. The result is ultimately a checkbit value associated with the complete 64-bit data to be written to the array.



This technique calculates the ultimate checkbit value on a read-modify-write transaction through a series of data manipulations, taking care to avoid discarding the original checkbits provided by the requesting master, such that faults in the crossbar (XBAR) and RAM controller datapath and control logic are covered by e2eECC.

Malfunction of the ECC logic described above may result in the corruption of the SEC/DED event reporting. The RAM controller performs EDC after ECC check on all read-modify-write transactions. If a mismatch is detected, indicating a failure in the ECC logic, the event is reported to the FCCU module.

Chapter 29
Flash memory controller

29.1 Flash memory controller (PFLASH) chip-specific module information

29.1.1 PFCR1[RWSC]

Although the PFLASH_PFCR1[RWSC] field is 5 bits wide, the MSB of this field has no effect on this chip.

29.2 Introduction

The flash memory controller has these functions:

- It acts as an interface between the system bus (AHB-Lite 2.v6) and the flash memory array.
- It serves as the interface to the on-chip overlay RAM.
- It serves as the interface to on-chip System RAM that can be used as overlay RAM.
- It serves as the interface to the off-chip buddy device.[1]

The flash memory controller supports two 64-bit AHB buses and a 256-bit read data interface from each flash memory array. Each AHB port contains a 4-entry, 2-way set-associative mini-cache as well as an associated controller that prefetches sequential lines of data from the flash arrays into the mini-cache. This buffer mechanism serves to deliver flash read data with zero-wait state response on lines that reside in the cache. AHB requests that miss the cache generate the needed flash array access and are forwarded to the AHB upon completion. Each mini-cache entry is 256 bits in size, matching the flash array page size and providing 256 bytes of total buffered storage. Along with the read and

1. The term "buddy device" refers to an optional companion chip with additional memory and debug features that is paired with a production chip during system development. It is not used in production devices.

write data, the corresponding ECC codewords traverse the entire datapath of the flash memory controller, including the mini-cache, to support end-to-end ECC (e2eECC) coverage.

29.3 Features

The following list summarizes the key features of the flash memory controller:

- Two 64-bit AHB interface ports (p0, p1) allowing simultaneous access to dedicated prefetch mini-cache per slave port.
- 256-bit read data bus + 64-bit write data bus
- Configurable read buffering and line prefetching support via 4-entry, 2-way set-associative mini-cache plus prefetch controller per AHB port to provide single-cycle "buffer hit" read response.
- Configurable access control based on read/write and AHB master ID attributes.
- Configurable access timing (wait-state programmable) allowing use in a wide range of frequency targets.
- Optional address pipelining capability to maximize throughput.
- Support for reporting of single- and multi-bit flash ECC events on a 64-bit doubleword boundary.
- Configurable overlay remapping of logical flash accesses to on-chip calibration RAM, extended off-chip calibration RAM, on-chip system RAM
- Nexus trace stream interface supports message dumping to the on-chip overlay RAM

29.4 Block diagrams

Figure 29-1 provides a block diagram showing the flash memory controller and the attached flash array.

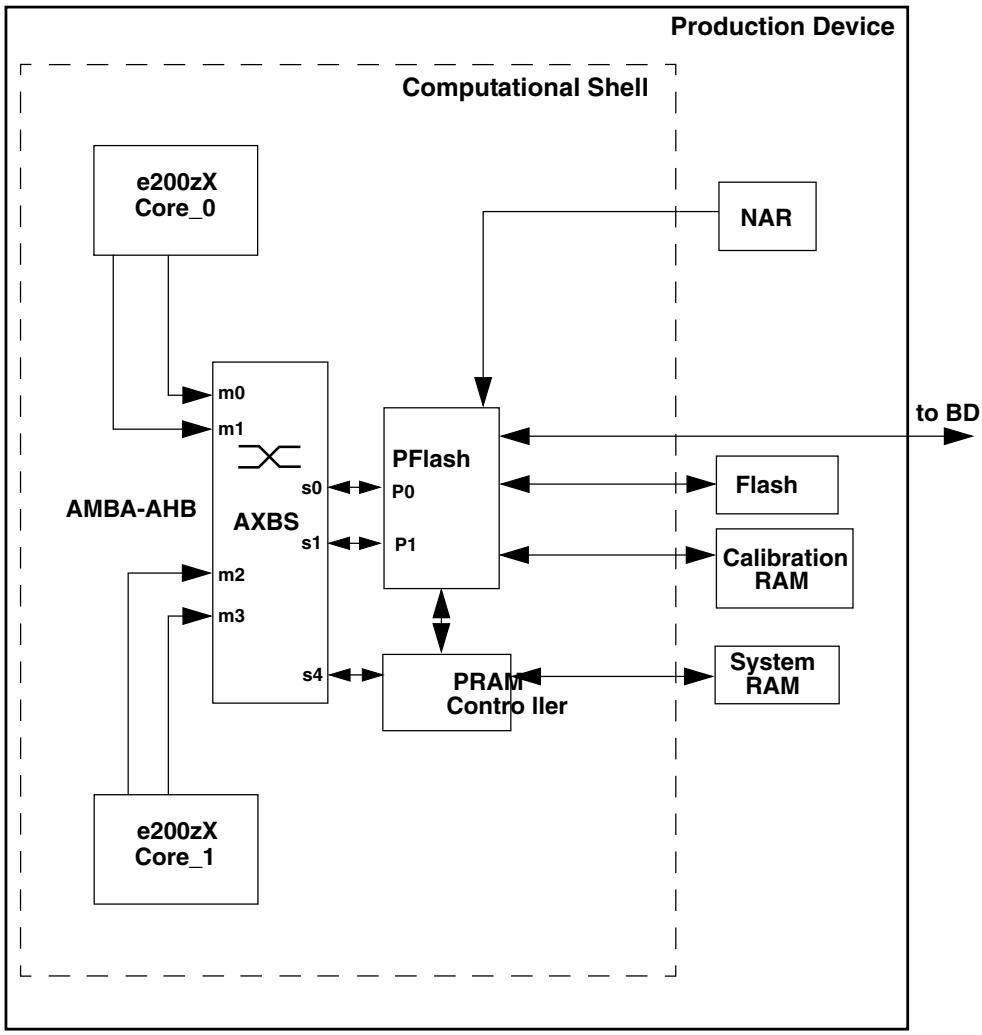


Figure 29-1. Platform-centric simple block diagram with flash memory ontroller

29.5 Flash memory controller memory map

The flash memory controller module provides an IPS programming model mapped to a standard 16 KB on-platform peripheral slot. The programming model consists of flash memory access configurationand overlay remapping configuration.

The programming model can only be referenced using a 32-bit (word) access. Attempted references using different access sizes or to undefined (reserved) addresses or in user mode generates an IPS error termination. The flash controller allows access to the programming model by all system bus masters.

The programming model can only be accessed in supervisor mode.

Attempted updates to the programming model while the flash memory controller module is in the midst of an operation will result in non-deterministic behavior. Software must be architected to avoid this scenario. There is no idle indicator for the flash controller. The recommended flow for multi-core devices is to start only one core and execute initialization code to completion before starting the remaining cores. If the user needs to reconfigure the flash, code execution must be temporarily moved to system RAM.

PFLASH memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Platform Flash Configuration Register 1 (PFLASH_PFCR1)	32	R/W	0000_0601h	29.5.1/798
4	Platform Flash Configuration Register 2 (PFLASH_PFCR2)	32	R/W	0000_0001h	29.5.2/802
8	Platform Flash Configuration Register 3 (PFLASH_PFCR3)	32	R/W	0000_0000h	29.5.3/805
C	Platform Flash Access Protection Register (PFLASH_PFAPR)	32	R/W	FFFF_FFFFh	29.5.4/807
10	Platform Flash Remap Control Register (PFLASH_PFCRCR)	32	R/W	0000_0000h	29.5.5/810
14	Platform Flash Remap Descriptor Enable Register (PFLASH_PFCRDE)	32	R/W	0000_0000h	29.5.6/812
100	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD0_Word0)	32	R/W	0000_0000h	29.5.7/817
104	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD0_Word1)	32	R/W	0000_0000h	29.5.8/818
108	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD0_Word2)	32	R/W	0000_0000h	29.5.9/819
110	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD1_Word0)	32	R/W	0000_0000h	29.5.7/817
114	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD1_Word1)	32	R/W	0000_0000h	29.5.8/818
118	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD1_Word2)	32	R/W	0000_0000h	29.5.9/819
120	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD2_Word0)	32	R/W	0000_0000h	29.5.7/817
124	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD2_Word1)	32	R/W	0000_0000h	29.5.8/818
128	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD2_Word2)	32	R/W	0000_0000h	29.5.9/819
130	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD3_Word0)	32	R/W	0000_0000h	29.5.7/817
134	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD3_Word1)	32	R/W	0000_0000h	29.5.8/818
138	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD3_Word2)	32	R/W	0000_0000h	29.5.9/819
140	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD4_Word0)	32	R/W	0000_0000h	29.5.7/817
144	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD4_Word1)	32	R/W	0000_0000h	29.5.8/818

Table continues on the next page...

PFLASH memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
148	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD4_Word2)	32	R/W	0000_0000h	29.5.9/819
150	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD5_Word0)	32	R/W	0000_0000h	29.5.7/817
154	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD5_Word1)	32	R/W	0000_0000h	29.5.8/818
158	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD5_Word2)	32	R/W	0000_0000h	29.5.9/819
160	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD6_Word0)	32	R/W	0000_0000h	29.5.7/817
164	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD6_Word1)	32	R/W	0000_0000h	29.5.8/818
168	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD6_Word2)	32	R/W	0000_0000h	29.5.9/819
170	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD7_Word0)	32	R/W	0000_0000h	29.5.7/817
174	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD7_Word1)	32	R/W	0000_0000h	29.5.8/818
178	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD7_Word2)	32	R/W	0000_0000h	29.5.9/819
180	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD8_Word0)	32	R/W	0000_0000h	29.5.7/817
184	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD8_Word1)	32	R/W	0000_0000h	29.5.8/818
188	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD8_Word2)	32	R/W	0000_0000h	29.5.9/819
190	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD9_Word0)	32	R/W	0000_0000h	29.5.7/817
194	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD9_Word1)	32	R/W	0000_0000h	29.5.8/818
198	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD9_Word2)	32	R/W	0000_0000h	29.5.9/819
1A0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD10_Word0)	32	R/W	0000_0000h	29.5.7/817
1A4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD10_Word1)	32	R/W	0000_0000h	29.5.8/818
1A8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD10_Word2)	32	R/W	0000_0000h	29.5.9/819
1B0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD11_Word0)	32	R/W	0000_0000h	29.5.7/817
1B4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD11_Word1)	32	R/W	0000_0000h	29.5.8/818
1B8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD11_Word2)	32	R/W	0000_0000h	29.5.9/819

Table continues on the next page...

PFLASH memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1C0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD12_Word0)	32	R/W	0000_0000h	29.5.7/817
1C4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD12_Word1)	32	R/W	0000_0000h	29.5.8/818
1C8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD12_Word2)	32	R/W	0000_0000h	29.5.9/819
1D0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD13_Word0)	32	R/W	0000_0000h	29.5.7/817
1D4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD13_Word1)	32	R/W	0000_0000h	29.5.8/818
1D8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD13_Word2)	32	R/W	0000_0000h	29.5.9/819
1E0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD14_Word0)	32	R/W	0000_0000h	29.5.7/817
1E4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD14_Word1)	32	R/W	0000_0000h	29.5.8/818
1E8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD14_Word2)	32	R/W	0000_0000h	29.5.9/819
1F0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD15_Word0)	32	R/W	0000_0000h	29.5.7/817
1F4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD15_Word1)	32	R/W	0000_0000h	29.5.8/818
1F8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD15_Word2)	32	R/W	0000_0000h	29.5.9/819
200	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD16_Word0)	32	R/W	0000_0000h	29.5.7/817
204	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD16_Word1)	32	R/W	0000_0000h	29.5.8/818
208	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD16_Word2)	32	R/W	0000_0000h	29.5.9/819
210	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD17_Word0)	32	R/W	0000_0000h	29.5.7/817
214	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD17_Word1)	32	R/W	0000_0000h	29.5.8/818
218	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD17_Word2)	32	R/W	0000_0000h	29.5.9/819
220	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD18_Word0)	32	R/W	0000_0000h	29.5.7/817
224	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD18_Word1)	32	R/W	0000_0000h	29.5.8/818
228	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD18_Word2)	32	R/W	0000_0000h	29.5.9/819
230	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD19_Word0)	32	R/W	0000_0000h	29.5.7/817

Table continues on the next page...

PFLASH memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
234	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD19_Word1)	32	R/W	0000_0000h	29.5.8/818
238	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD19_Word2)	32	R/W	0000_0000h	29.5.9/819
240	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD20_Word0)	32	R/W	0000_0000h	29.5.7/817
244	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD20_Word1)	32	R/W	0000_0000h	29.5.8/818
248	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD20_Word2)	32	R/W	0000_0000h	29.5.9/819
250	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD21_Word0)	32	R/W	0000_0000h	29.5.7/817
254	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD21_Word1)	32	R/W	0000_0000h	29.5.8/818
258	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD21_Word2)	32	R/W	0000_0000h	29.5.9/819
260	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD22_Word0)	32	R/W	0000_0000h	29.5.7/817
264	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD22_Word1)	32	R/W	0000_0000h	29.5.8/818
268	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD22_Word2)	32	R/W	0000_0000h	29.5.9/819
270	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD23_Word0)	32	R/W	0000_0000h	29.5.7/817
274	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD23_Word1)	32	R/W	0000_0000h	29.5.8/818
278	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD23_Word2)	32	R/W	0000_0000h	29.5.9/819
280	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD24_Word0)	32	R/W	0000_0000h	29.5.7/817
284	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD24_Word1)	32	R/W	0000_0000h	29.5.8/818
288	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD24_Word2)	32	R/W	0000_0000h	29.5.9/819
290	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD25_Word0)	32	R/W	0000_0000h	29.5.7/817
294	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD25_Word1)	32	R/W	0000_0000h	29.5.8/818
298	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD25_Word2)	32	R/W	0000_0000h	29.5.9/819
2A0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD26_Word0)	32	R/W	0000_0000h	29.5.7/817
2A4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD26_Word1)	32	R/W	0000_0000h	29.5.8/818

Table continues on the next page...

PFLASH memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2A8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD26_Word2)	32	R/W	0000_0000h	29.5.9/819
2B0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD27_Word0)	32	R/W	0000_0000h	29.5.7/817
2B4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD27_Word1)	32	R/W	0000_0000h	29.5.8/818
2B8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD27_Word2)	32	R/W	0000_0000h	29.5.9/819
2C0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD28_Word0)	32	R/W	0000_0000h	29.5.7/817
2C4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD28_Word1)	32	R/W	0000_0000h	29.5.8/818
2C8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD28_Word2)	32	R/W	0000_0000h	29.5.9/819
2D0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD29_Word0)	32	R/W	0000_0000h	29.5.7/817
2D4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD29_Word1)	32	R/W	0000_0000h	29.5.8/818
2D8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD29_Word2)	32	R/W	0000_0000h	29.5.9/819
2E0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD30_Word0)	32	R/W	0000_0000h	29.5.7/817
2E4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD30_Word1)	32	R/W	0000_0000h	29.5.8/818
2E8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD30_Word2)	32	R/W	0000_0000h	29.5.9/819
2F0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD31_Word0)	32	R/W	0000_0000h	29.5.7/817
2F4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD31_Word1)	32	R/W	0000_0000h	29.5.8/818
2F8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD31_Word2)	32	R/W	0000_0000h	29.5.9/819

29.5.1 Platform Flash Configuration Register 1 (PFLASH_PFCR1)

The PFlash Configuration Register 1 (PFCR1) controls the operation of Port p0 of the PFLASH_C55FM flash memory controller.

NOTE

See the chip-specific platform flash controller information for information on the masters.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	P0_M15PFE	P0_M14PFE	P0_M13PFE	P0_M12PFE	P0_M11PFE	P0_M10PFE	P0_M9PFE	P0_M8PFE	P0_M7PFE	P0_M6PFE	P0_M5PFE	P0_M4PFE	P0_M3PFE	P0_M2PFE	P0_M1PFE	P0_M0PFE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	APC			RWSC					0	P0_DPFEN	0	P0_IPFEN	0	P0_PFLIM		P0_BFEN
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1

PFLASH_PFCR1 field descriptions

Field	Description
0 P0_M15PFE	Port0 Master 15 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 15. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
1 P0_M14PFE	Port0 Master 14 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 14. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
2 P0_M13PFE	Port0 Master 13 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 13. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
3 P0_M12PFE	Port0 Master 12 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 12. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
4 P0_M11PFE	Port0 Master 11 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 11. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
5 P0_M10PFE	Port0 Master 10 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 10. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
6 P0_M9PFE	Port0 Master 9 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 9. This bit is cleared by hardware reset.

Table continues on the next page...

PFLASH_PFCR1 field descriptions (continued)

Field	Description
	0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
7 P0_M8PFE	Port0 Master 8 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 8. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
8 P0_M7PFE	Port0 Master 7 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 7. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
9 P0_M6PFE	Port0 Master 6 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 6. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
10 P0_M5PFE	Port0 Master 5 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 5. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
11 P0_M4PFE	Port0 Master 4 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 4. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
12 P0_M3PFE	Port0 Master 3 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 3. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
13 P0_M2PFE	Port0 Master 2 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 2. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
14 P0_M1PFE	Port0 Master 1 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 1. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
15 P0_M0PFE	Port0 Master 0 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 0. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
16–18 APC	Address Pipeline Control. This field controls the number of cycles that a subsequent flash read from the opposite AHB port can be initiated prior to the previous read data being valid. This field applies to the configuration of Port0 and Port1.

Table continues on the next page...

PFLASH_PFCR1 field descriptions (continued)

Field	Description
	For valid RWSC and APC combinations please refer to the device data sheet. **NOTE:** 1. A value of '1' in the most significant bit causes the flash controller to function in retrograde/ legacy mode, in which there is no pipelining between the sampling of flash read data and the presentation of the next address for flash lookup. 2. Flash operation is not guaranteed for RWSC/APC combinations other than those specified in the data sheet. 000 Pipelined access to the flash disabled. 001 A pipelined access can be initiated 1 cycle before the previous data is valid 010 A pipelined access can be initiated 2 cycles before the previous data is valid 011 A pipelined access can be initiated 3 cycles before the previous data is valid 1xx Pipelined access to the flash is disabled and one wait state is inserted before a subsequent access can be initiated
19–23 RWSC	Read Wait State Control. This field controls the number of wait-states to be added to the best-case flash array access time for reads. The best-case flash array access time for reads is one cycle. This field must be set to a value corresponding to the operating frequency of the flash memory controller and the actual read access time of the flash memory controller. For valid RWSC and APC combinations please refer to the device data sheet. **NOTE:** 1. Higher operating frequencies require non-zero settings for this field for proper flash operation. 2. Flash operation is not guaranteed for RWSC/APC combinations other than those specified in the device data sheet. This field applies to the configuration of Port0 and Port1. 00000 No additional wait-states are added 00001 One additional wait-state is added ... 11111 Thirty-one additional wait-states are added
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 P0_DPFEN	Port0 Data Prefetch Enable. This field enables or disables prefetching initiated by a data read access. This field is cleared by hardware reset. 0 No prefetching is triggered by a data read access 1 Prefetching may be triggered by any data read access
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 P0_IPFEN	Port0 Instruction Prefetch Enable. This bit enables or disables prefetching initiated by an instruction read access. This field is cleared by hardware reset. 0 No prefetching is triggered by an instruction read access 1 Prefetching may be triggered by any instruction read access
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 P0_PFLIM	Port0 PFlash Prefetch Limit. This field controls the prefetch algorithm used by the prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit.

Table continues on the next page...

PFLASH_PFCR1 field descriptions (continued)

Field	Description
	This field is cleared by hardware reset. 00 No prefetching or buffering is performed. 01 The referenced line is prefetched on a buffer miss, that is, prefetch on miss. 10 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit. 11 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit.
31 P0_BFEN	Port0 PFlash Line Read Buffers Enable. This bit enables or disables line read buffer hits. It is also used to invalidate the buffers. This bit can only be updated while PFCR3[BAF_DIS]=0. After execution of the boot code, as indicated by PFCR3[BAF_DIS]=1, this bit becomes read-only until the next hardware reset. 0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.

29.5.2 Platform Flash Configuration Register 2 (PFLASH_PFCR2)

The PFlash Configuration Register 2 (PFCR2) controls the operation of Port1 of the PFLASH_C55FM flash memory controller.

NOTE

See the chip-specific platform flash controller information for details about the masters.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	P1_M15PFE	P1_M14PFE	P1_M13PFE	P1_M12PFE	P1_M11PFE	P1_M10PFE	P1_M9PFE	P1_M8PFE	P1_M7PFE	P1_M6PFE	P1_M5PFE	P1_M4PFE	P1_M3PFE	P1_M2PFE	P1_M1PFE	P1_M0PFE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										0		0			
W										P1_DPFEN		P1_IPFEN		P1_PFLIM		P1_BFEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

PFLASH_PFCR2 field descriptions

Field	Description
0 P1_M15PFE	Port1 Master 15 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 15. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
1 P1_M14PFE	Port1 Master 14 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 14. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
2 P1_M13PFE	Port1 Master 13 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 13. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
3 P1_M12PFE	Port1 Master 12 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 12. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
4 P1_M11PFE	Port1 Master 11 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 11. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
5 P1_M10PFE	Port1 Master 10 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 10. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
6 P1_M9PFE	Port1 Master 9 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 9. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
7 P1_M8PFE	Port1 Master 8 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 8. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
8 P1_M7PFE	Port1 Master 7 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 7. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
9 P1_M6PFE	Port1 Master 6 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 6. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master

Table continues on the next page...

PFLASH_PFCR2 field descriptions (continued)

Field	Description
10 P1_M5PFE	Port1 Master 5 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 5. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
11 P1_M4PFE	Port1 Master 4 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 4. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
12 P1_M3PFE	Port1 Master 3 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 3. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
13 P1_M2PFE	Port1 Master 2 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 2. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
14 P1_M1PFE	Port1 Master 1 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 1. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
15 P1_M0PFE	Port1 Master 0 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 0. This bit is cleared by hardware reset. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 P1_DPFEN	Port1 Data Prefetch Enable. This field enables or disables prefetching initiated by a data read access. This field is cleared by hardware reset. 0 No prefetching is triggered by a data read access 1 Prefetching may be triggered by any data read access
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 P1_IPFEN	Port1 Instruction Prefetch Enable. This bit enables or disables prefetching initiated by an instruction read access. This field is cleared by hardware reset. 0 No prefetching is triggered by an instruction read access 1 Prefetching may be triggered by any instruction read access
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 P1_PFLIM	Port1 PFlash Prefetch Limit. This field controls the prefetch algorithm used by the prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. This field is cleared by hardware reset.

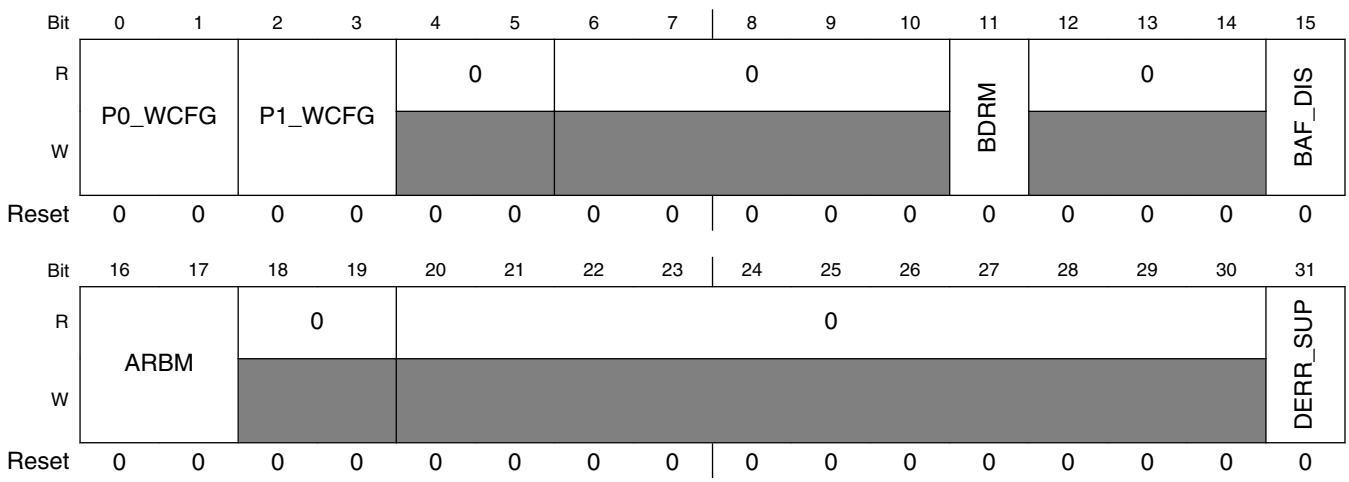
Table continues on the next page...

PFLASH_PFCR2 field descriptions (continued)

Field	Description
	00 No prefetching or buffering is performed. 01 The referenced line is prefetched on a buffer miss, that is, prefetch on miss. 10 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit. 11 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit
31 P1_BFEN	Port1 PFlash Line Read Buffers Enable. This bit enables or disables line read buffer hits. It is also used to invalidate the buffers. This bit can only be updated while PFCR3[BAF_DIS]=0. After execution of the boot code, as indicated by PFCR3[BAF_DIS] = 1, this bit becomes read-only until the next hardware reset. 0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.

29.5.3 Platform Flash Configuration Register 3 (PFLASH_PFCR3)

Address: 0h base + 8h offset = 8h



PFLASH_PFCR3 field descriptions

Field	Description
0–1 P0_WCFG	Port0 Way Configuration. This field controls the configuration of the line buffers for a given set acrosstwo ways in the controller cache. The indexed set can be organized as a "pool" of available resources, or with a fixed partition between instruction and data buffers. In all cases, when a buffer miss occurs, the flash page is assigned a location within the controller cache. Within the indexed set, the way is selected using a least-recently-used replacement policy, and the entry is then marked as most-recently-used for that set. If the flash access is for the next-sequential line (prefetch), the buffer is not marked as most-recently-used until the given address produces a buffer hit. This field is initialized by hardware reset. 00 Both buffers in an indexed set are available for any flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved

Table continues on the next page...

PFLASH_PFCR3 field descriptions (continued)

Field	Description
	10 The buffers in an indexed set are partitioned into two groups with way 0 allocated for instruction fetches and way 1 for data accesses. 11 Reserved
2–3 P1_WCFG	Port1 Way Configuration. This field controls the configuration of the line buffers for a given set across two ways in the controller cache. The indexed set can be organized as a "pool" of available resources, or with a fixed partition between instruction and data buffers. In all cases, when a buffer miss occurs, the flash page is assigned a location within the controller cache. Within the indexed set, the way is selected using a least-recently-used replacement policy, and the entry is then marked as most-recently-used for that set. If the flash access is for the next-sequential line (prefetch), the buffer is not marked as most-recently-used until the given address produces a buffer hit. This field is initialized by hardware reset. 00 Both buffers are available for any flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved 10 The buffers are partitioned into two groups with way 0 allocated for instruction fetches and way 1 for data accesses. 11 Reserved
4–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 BDRM	Buddy Device Read Mode - This field controls the PFLASH_C55FM bus response when the master presents a read request targeting the extended overlay RAM on the Buddy Device(BD) using the direct, non-remapped address. This field is ignored on production ECUs where the Buddy Device is not present. On production ECUs, this memory map region is provided for the "OLDA" (on-line data acquisition) function. Read requests on production ECUs targeting the non-existent extended overlay RAM using the direct, non-remapped address are terminated with error on the system bus. This field is ignored on write requests targeting the extended overlay RAM using the direct, non-remapped address. Write requests on production ECUs targeting the non-existent extended overlay RAM using the direct, non-remapped address are ignored and terminate cleanly on the system bus. Write requests on engineering ECUs targeting the extended overlay RAM using the direct, non-remapped address are executed. This field is initialized by hardware reset. 0 Read requests targeting the extended overlay RAM on the Buddy Device(BD) using the direct, non-remapped address are aborted, and the request is terminated with error on the system bus. 1 Read requests targeting the extended overlay RAM on the Buddy Device(BD) using the direct, non-remapped address are allowed. The PFLASH_C55FM returns the requested data from the extended overlay RAM, which resides on the BD. Any system bus errors returned by the BD are passed onto the master.
12–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 BAF_DIS	BAF Disable. This field controls executable access to the BAF (Boot Assist Flash) region of the flash. Once this field is set, attempted instruction accesses targeting the BAF region are aborted and terminated with a system bus error. The affect of this field applies to system bus transfers through both, Port0 and Port1. Data-type accesses to the BAF region are not affected by this field.

Table continues on the next page...

PFLASH_PFCR3 field descriptions (continued)

Field	Description
	Once this field is set, it becomes a read-only field and can only be cleared by hardware reset. Once this field is set, any subsequent write attempts to modify this field are ignored with an error-free data transfer termination. This field is initialized by hardware reset. 0 Executable access to the BAF flash region is allowed. 1 Executable access to the BAF flash region is prohibited.
16–17 ARBM	Arbitration Mode. This 2-bit field controls the arbitration of concurrent flash access requests from the two AHB ports of the flash memory controller. In both fixed priority or round-robin modes, write requests are prioritized higher than read requests, and read requests are prioritized higher than speculative prefetch requests whenever both ports issue concurrent requests. This field is initialized by hardware reset. 00 Fixed priority arbitration with AHB p0>p1 01 Fixed priority arbitration with AHB p1>p0 10 Round-robin arbitration 11 Round-robin arbitration
18–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 DERR_SUP	Data Error Suppression. Refer to the Embedded Flash Memory configuration information for which Flash blocks are affected by this field. 0 ECC events on data flash accesses are reported. 1 ECC events on data flash accesses are suppressed. Single-bit and multi-bit ECC errors are not reported. On multibit ECC events, the corrupted read data is replaced with a fixed, ECC-clean illegal opcode value of 0x15551555.

29.5.4 Platform Flash Access Protection Register (PFLASH_PFAPR)

The PFlash Access Protection Register (PFAPR) is used to control read and write accesses to the flash array.

NOTE

See the chip-specific platform flash controller information for details about the actual masters available on the device.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	M0AP		M1AP		M2AP		M3AP		M4AP		M5AP		M6AP		M7AP	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	M8AP		M9AP		M10AP		M11AP		M12AP		M13AP		M14AP		M15AP	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

PFLASH_PFAPR field descriptions

Field	Description
0–1 M0AP	Master 0 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
2–3 M1AP	Master 1 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
4–5 M2AP	Master 2 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
6–7 M3AP	Master 3 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
8–9 M4AP	Master 4 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
10–11 M5AP	Master 5 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
12–13 M6AP	Master 6 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

Table continues on the next page...

PFLASH_PFAPR field descriptions (continued)

Field	Description
14–15 M7AP	Master 7 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
16–17 M8AP	Master 8 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
18–19 M9AP	Master 9 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
20–21 M10AP	Master 10 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
22–23 M11AP	Master 11 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
24–25 M12AP	Master 12 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
26–27 M13AP	Master 13 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

Table continues on the next page...

PFLASH_PFAPR field descriptions (continued)

Field	Description
28–29 M14AP	Master 14 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
30–31 M15AP	Master 15 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

29.5.5 Platform Flash Remap Control Register (PFLASH_PFCRCR)

The PFCRCR is used to globally enable/disable the calibration remap function.

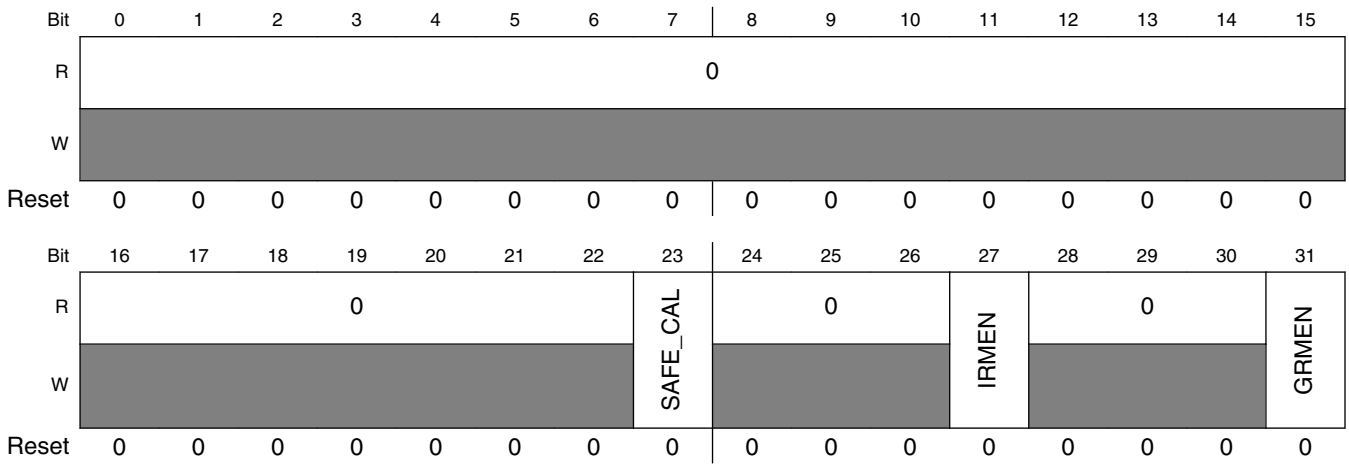
CAUTION

Overlay remap functions, when used with line read buffers enabled (PFLASH_PFCRx[BFy_EN] field(s)='1') have a potential to cause data coherency issues, i.e., prefetched data in a line read buffer can become out-of-sync with data contained at the associated flash address range. To prevent this issue, always clear the flash controller line read buffers prior to enabling any remap region.

To clear the flash controller line read buffers, issue an interlock write command to a valid flash memory address. There will not be an actual flash program operation performed, but the flash controller mini-cache will be cleared as a side effect of executing the command. The interlock write must be issued from a core having appropriate access to flash. The sequence is as follows:

1. Set the C55FMC_MCR[PGM] field to '1'
2. Issue a 64-bit write to a valid flash address on a 64-bit boundary, i.e., the least significant 5 bits of the address are '0'
3. Clear C55FMC_MCR[PGM]

Address: 0h base + 10h offset = 10h



PFLASH_PFCRCR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SAFE_CAL	Safe Calibration. This bit facilitates the use of safety-critical calibration data. When this bit is enabled, the flash memory controller is configured to perform redundancy checking on the calibration remap function and reduce the total number of potential calibration regions from 32 to 16. This field is initialized by hardware reset. 1 Established calibration overlay regions are considered safety-critical. 0 Established calibration overlay regions are not considered safety-critical.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 IRMEN	Instruction Remap. This bit enables calibration remapping evaluation on instruction fetches. When PFCRCR[GRMEN] is disabled, this bit is ignored. This field is initialized by hardware reset. 0 Calibration remap evaluation is performed on any incoming data fetch requests only. 1 Calibration remap evaluation is performed on all incoming flash access requests - data and instruction fetches.
28–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 GRMEN	Global Remap Enable. This bit globally enables or disables the calibration remapping evaluation on system flash accesses. This field is initialized by hardware reset. 0 Calibration remap evaluation is not performed on any incoming system flash access requests. 1 Calibration remap evaluation is performed on all incoming system flash access requests.

29.5.6 Platform Flash Remap Descriptor Enable Register (PFLASH_PFCRDE)

The PFlash Calibration Remap Descriptor Enable Register (PFCRDE) is used to enable or disable up to 32 calibration remap descriptors. Note there is also a global remap enable (PFCRCR[GRMEN]) that also must be asserted in conjunction with the individual CRDnEN flags to enable a given remap descriptor.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CRD0EN	CRD1EN	CRD2EN	CRD3EN	CRD4EN	CRD5EN	CRD6EN	CRD7EN	CRD8EN	CRD9EN	CRD10EN	CRD11EN	CRD12EN	CRD13EN	CRD14EN	CRD15EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CRD16EN	CRD17EN	CRD18EN	CRD19EN	CRD20EN	CRD21EN	CRD22EN	CRD23EN	CRD24EN	CRD25EN	CRD26EN	CRD27EN	CRD28EN	CRD29EN	CRD30EN	CRD31EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PFLASH_PFCRDE field descriptions

Field	Description
0 CRD0EN	Calibration Remap Descriptor 0 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 0 invalid 1 Calibration Remap Descriptor 0 valid
1 CRD1EN	Calibration Remap Descriptor 1 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 1 invalid 1 Calibration Remap Descriptor 1 valid
2 CRD2EN	Calibration Remap Descriptor 2 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding

Table continues on the next page...

PFLASH_PFCRDE field descriptions (continued)

Field	Description
	PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 2 invalid 1 Calibration Remap Descriptor 2 valid
3 CRD3EN	Calibration Remap Descriptor 3 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 3 invalid 1 Calibration Remap Descriptor 3 valid
4 CRD4EN	Calibration Remap Descriptor 4 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 4 invalid 1 Calibration Remap Descriptor 4 valid
5 CRD5EN	Calibration Remap Descriptor 5 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 5 invalid 1 Calibration Remap Descriptor 5 valid
6 CRD6EN	Calibration Remap Descriptor 6 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 6 invalid 1 Calibration Remap Descriptor 6 valid
7 CRD7EN	Calibration Remap Descriptor 7 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 7 invalid 1 Calibration Remap Descriptor 7 valid
8 CRD8EN	Calibration Remap Descriptor 8 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset.

Table continues on the next page...

PFLASH_PFCRDE field descriptions (continued)

Field	Description
	0 Calibration Remap Descriptor 8 invalid 1 Calibration Remap Descriptor 8 valid
9 CRD9EN	Calibration Remap Descriptor 9 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 9 invalid 1 Calibration Remap Descriptor 9 valid
10 CRD10EN	Calibration Remap Descriptor 10 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 10 invalid 1 Calibration Remap Descriptor 10 valid
11 CRD11EN	Calibration Remap Descriptor 11 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 11 invalid 1 Calibration Remap Descriptor 11 valid
12 CRD12EN	Calibration Remap Descriptor 12 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 12 invalid 1 Calibration Remap Descriptor 12 valid
13 CRD13EN	Calibration Remap Descriptor 13 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 13 invalid 1 Calibration Remap Descriptor 13 valid
14 CRD14EN	Calibration Remap Descriptor 14 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 14 invalid 1 Calibration Remap Descriptor 14 valid
15 CRD15EN	Calibration Remap Descriptor 15 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to

Table continues on the next page...

PFLASH_PFCRDE field descriptions (continued)

Field	Description
	enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 15 invalid 1 Calibration Remap Descriptor 15 valid
16 CRD16EN	Calibration Remap Descriptor 16 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 16 invalid 1 Calibration Remap Descriptor 16 valid
17 CRD17EN	Calibration Remap Descriptor 17 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 17 invalid 1 Calibration Remap Descriptor 17 valid
18 CRD18EN	Calibration Remap Descriptor 18 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 18 invalid 1 Calibration Remap Descriptor 18 valid
19 CRD19EN	Calibration Remap Descriptor 19 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 19 invalid 1 Calibration Remap Descriptor 19 valid
20 CRD20EN	Calibration Remap Descriptor 20 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 20 invalid 1 Calibration Remap Descriptor 20 valid
21 CRD21EN	Calibration Remap Descriptor 21 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset.

Table continues on the next page...

PFLASH_PFCRDE field descriptions (continued)

Field	Description
	0 Calibration Remap Descriptor 21 invalid 1 Calibration Remap Descriptor 21 valid
22 CRD22EN	Calibration Remap Descriptor 22 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 22n invalid 1 Calibration Remap Descriptor 22 valid
23 CRD23EN	Calibration Remap Descriptor 23 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 23 invalid 1 Calibration Remap Descriptor 23 valid
24 CRD24EN	Calibration Remap Descriptor 24 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 24 invalid 1 Calibration Remap Descriptor 24 valid
25 CRD25EN	Calibration Remap Descriptor 25 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 25 invalid 1 Calibration Remap Descriptor 25 valid
26 CRD26EN	Calibration Remap Descriptor 26 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 26 invalid 1 Calibration Remap Descriptor 26 valid
27 CRD27EN	Calibration Remap Descriptor 27 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 27 invalid 1 Calibration Remap Descriptor 27 valid
28 CRD28EN	Calibration Remap Descriptor 28 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to

Table continues on the next page...

PFLASH_PFCRDE field descriptions (continued)

Field	Description
	enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 28 invalid 1 Calibration Remap Descriptor 28 valid
29 CRD29EN	Calibration Remap Descriptor 29 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 29 invalid 1 Calibration Remap Descriptor 29 valid
30 CRD30EN	Calibration Remap Descriptor 30 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 30 invalid 1 Calibration Remap Descriptor 30 valid
31 CRD31EN	Calibration Remap Descriptor 31 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 31 invalid 1 Calibration Remap Descriptor 31 valid

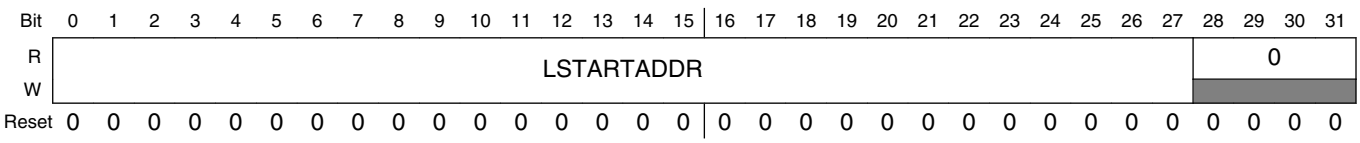
29.5.7 Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD*n*_Word0)

Each 96-bit (12-byte) region descriptor specifies an overlay region where a flash access can be remapped during calibration and debug. The calibration remap descriptors are organized sequentially as 128-bit (16-byte) structures in the Platform Flash Controller's programming model. Each of the three 32-bit words that define a single calibration region are detailed in the subsequent sections; the fourth word is unused.

The first word of the flash memory controller overlay region descriptor defines the 0-modulo-size logical start (byte) address of the calibration remap region. It is software's responsibility to guarantee the low-order bits of the address, as defined by the remap descriptor size, are zeroed to enable the remap descriptor hit logic to function correctly.

The hardware performs basic checks on the validity of the logical start address when this word is written; if a non-supported logical start address is defined, the register write is terminated with an error and the register left unchanged and the valid bit cleared. Successful writes to this word also clear the calibration remap descriptor's valid bit.

Address: 0h base + 100h offset + (16d × i), where i=0d to 31d



PFLASH_PFCRD*n*_Word0 field descriptions

Field	Description
0–27 LSTARTADDR	Logical Start Address. This field defines the most significant bits of the 0-modulo-size logical start address of the overlay remap region. It corresponds to the logical system address which maps to the flash memory space. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid.
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

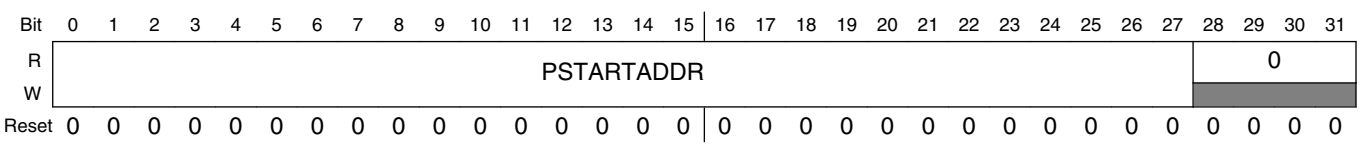
29.5.8 Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD*n*_Word1)

The second word of the flash memory controller calibration region descriptor defines the 0-modulo-size byte physical start (byte) address of the calibration remap region.

The contents of this word define the targeted destination overlay memory. It is software's responsibility to guarantee the low-order bits of the address, as defined by the remap descriptor size, are zeroed to enable the remap descriptor hit logic to function correctly.

Attempted writes to this register with illegal values, as defined by the sizes and locations of the overlay memories, are terminated with an error and clear the valid bit. Successful writes to this word also clear the calibration remap descriptor's valid bit.

Address: 0h base + 104h offset + (16d × i), where i=0d to 31d



PFLASH_PFCRD*n*_Word1 field descriptions

Field	Description
0–27 PSTARTADDR	Calibration Remap Descriptor *n* Physical Start Address - This field defines the most significant bits of the 0-modulo-size physical start byte address of the calibration remap descriptor. This address corresponds to

Table continues on the next page...

PFLASH_PFCRD*n*_Word1 field descriptions (continued)

Field	Description
	the physical address which maps to the destination calibration overlay memory. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid.
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

29.5.9 Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD*n*_Word2)

The third word of the calibration region descriptor defines a per-master calibration remap enable and the remap region size. For cacheable spaces being remapped, the minimum region size is 32 bytes to match the flash page and cache line sizes. Writes to this word clear the calibration remap descriptor's valid bit.

Address: 0h base + 108h offset + (16d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	M0EN	M1EN	M2EN	M3EN	M4EN	M5EN	M6EN	M7EN	M8EN	M9EN	M10EN	M11EN	M12EN	M13EN	M14EN	M15EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											CRDSize				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PFLASH_PFCRD*n*_Word2 field descriptions

Field	Description
0 M0EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash.

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	0 Calibration remap evaluation is ignored on flash access requests from Master 0 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 0 as defined by this region descriptor.
1 M1EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 1 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 1 as defined by this region descriptor.
2 M2EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 2 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 2 as defined by this region descriptor.
3 M3EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 3 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 3 as defined by this region descriptor.
4 M4EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash.

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	0 Calibration remap evaluation is ignored on flash access requests from Master 4 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 4 as defined by this region descriptor.
5 M5EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 5 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 5 as defined by this region descriptor.
6 M6EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 6 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 6 as defined by this region descriptor.
7 M7EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 7 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 7 as defined by this region descriptor.
8 M8EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash.

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	0 Calibration remap evaluation is ignored on flash access requests from Master 8 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 8 as defined by this region descriptor.
9 M9EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 9 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 9 as defined by this region descriptor.
10 M10EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 10 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 10 as defined by this region descriptor.
11 M11EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 11 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 11 as defined by this region descriptor.
12 M12EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash.

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	0 Calibration remap evaluation is ignored on flash access requests from Master 12 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 12 as defined by this region descriptor.
13 M13EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 13 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 13 as defined by this region descriptor.
14 M14EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 14 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 14 as defined by this region descriptor.
15 M15EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 15 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 15 as defined by this region descriptor.
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 CRDSize	Calibration Remap Descriptor *n* Size - This field specifies the size of the calibration remap region. 00000 Reserved 00001 Reserved 00010 Reserved 00011 Reserved

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	00100 Reserved 00101 Region size is 32 bytes. Low-order 5 bits of LSTARTADDR and PSTARTADDR must be zero 00110 Region size is 64 bytes. Low-order 6 bits of LSTARTADDR and PSTARTADDR must be zero 00111 Region size is 128 bytes. Low-order 7 bits of LSTARTADDR and PSTARTADDR must be zero 01000 Region size is 256 bytes. Low-order 8 bits of LSTARTADDR and PSTARTADDR must be zero 01001 Region size is 512 bytes. Low-order 9 bits of LSTARTADDR and PSTARTADDR must be zero 01010 Region size is 1 KB. Low-order 10 bits of LSTARTADDR and PSTARTADDR must be zero 01011 Region size is 2 KB. Low-order 11 bits of LSTARTADDR and PSTARTADDR must be zero 01100 Region size is 4 KB. Low-order 12 bits of LSTARTADDR and PSTARTADDR must be zero 01101 Region size is 8 KB. Low-order 13 bits of LSTARTADDR and PSTARTADDR must be zero 01110 Region size is 16 KB. Low-order 14 bits of LSTARTADDR and PSTARTADDR must be zero 01111 Region size is 32 KB. Low-order 15 bits of LSTARTADDR and PSTARTADDR must be zero 10000 Region size is 64 KB. Low-order 16 bits of LSTARTADDR and PSTARTADDR must be zero 10001 Region size is 128 KB. Low-order 17 bits of LSTARTADDR and PSTARTADDR must be zero 10010 Region size is 256 KB. Low-order 18 bits of LSTARTADDR and PSTARTADDR must be zero 10011 Region size is 512 KB. Low-order 19 bits of LSTARTADDR and PSTARTADDR must be zero 10100 Region size is 1 MB. Low-order 20 bits of LSTARTADDR and PSTARTADDR must be zero 10101 Region size is 2 MB. Low-order 21 bits of LSTARTADDR and PSTARTADDR must be zero 10110 Region size is 4 MB. Low-order 22 bits of LSTARTADDR and PSTARTADDR must be zero 10111 Region size is 8 MB. Low-order 23 bits of LSTARTADDR and PSTARTADDR must be zero 11000 Reserved 11001 Reserved 11010 Reserved 11011 Reserved 11100 Reserved 11101 Reserved 11110 Reserved 11111 Reserved

29.6 Functional description

The flash memory controller interfaces between:

- The AHB system bus ports
- Nexus trace
- Flash memory array
- Overlay RAM
- Off-chip Buddy Device
- System RAM

For accesses targeting the flash array, the flash memory controller generates read and write enables, block selects, array address, write size and write data as inputs to the flash array. The flash memory controller captures read data from the flash array and drives it onto the AHB system bus. Up to four pages of data (256-bit page size) may be buffered

in each of two ways of the flash memory controller mini-cache. Lines may be prefetched in advance of being requested, allowing single-cycle (zero AHB wait-states) read data responses on buffer hits.

Several prefetch control algorithms are available for controlling line read buffer fills. Prefetch triggering may be restricted to instruction accesses only, data accesses only, or may be unrestricted. Prefetch triggering may also be controlled on a per-master basis.

Buffers may also be selectively enabled or disabled for allocation by instruction and data prefetch.

Access protections may be applied on a per-master basis for both reads and writes to support security and privilege mechanisms.

29.6.1 Basic interface protocol

Read accesses are terminated under control of the appropriate wait state settings. Thus, the access time of the operation is determined by the setting of the PFCR1[RWSC] field. Access timing can be varied to account for the operating conditions of the SoC (frequency, voltage, temp) by appropriately setting the PFCR1[RWSC] field.

29.6.2 Access protections

On-chip Flash memory accesses are subject to restrictions based on chip security implementation and/or requirements based on functional restrictions of the flash memory, e.g., a read access to a flash block while it is currently being written is not allowed. When a prohibited access is attempted, a system bus error termination results.

The flash memory controller may invoke a system bus error termination in the following scenarios:

- Attempted access by an AHB master whose corresponding read access control or write access control settings do not allow the access, thus causing a protection violation. See Platform Flash Access Protection Register (PFLASH_PFAPR) for more detail. In this case the flash memory controller does not initiate the requested flash array access.
- Attempted instruction fetch by an AHB master to the BAF flash region when the PFCR3[BAF_DISABLE] field in Platform Flash Configuration Register 3 (PFCR3) is set. The assertion of PFCR3[BAF_DISABLE] indicates that code stored in the BAF region should not be executed. See Platform Flash Configuration Register 3

(PFLASH_PFCR3) for more detail. In this case, the flash memory controller does not initiate a flash array access.
- Attempted access by an AHB master to a flash region where the corresponding censorship control is asserted.
- The flash or overlay RAM returns an error response on an attempted access to a partition that is unavailable. (see Flash error response operation or Embedded Flash Memory).
- Attempted access by an AHB master to a reserved region in the flash memory map.

The flash array may also signal an error response to terminate a requested access due to improper sequencing during program/erase operations, improper sequencing during array integrity testing. When an error response is received the flash memory controller does not update or validate a page read buffer. An error response may be signaled on a read or interlock write operation. For more information on the specifics related to signaling of flash errors, including flash ECC events, array integrity testing and read-while-write events, refer to the flash memory chapter.

29.6.2.1 Censorship

The flash space is defined by regions:

- Code flash (instruction and constant data)
- Data flash (data for EEPROM emulation)
- UTEST flash

Each section can be independently censored, and the flash memory controller provides independent censorship control inputs for each region on a read vs. write basis.

29.6.2.2 PFAPR - Platform Flash Access Protection Register (PFAPR)

The flash memory controller provides programmable, configurable access protections for both read and write cycles on a per-master basis via the PFlash Access Protection Register (PFAPR). It allows restriction of read and write requests on a per-master basis. This functionality is described in Platform Flash Access Protection Register (PFLASH_PFAPR). Detection of a protection violation results in an error response from the flash memory controller on the AHB transfer.

29.6.2.3 BAF execution disable

The flash memory controller provides a mechanism to disable instruction execution of the BAF (Boot Assist Flash) code. Setting the BAF Disable field (BAF_DIS) in the PFlash Configuration Register 3 (PFCR3) prevents execution of the BAF instructions. When this field is set, all incoming requests from the system bus on Port0 or Port1 are evaluated. If the flash memory controller detects an attempted instruction fetch access to the BAF region of the flash, the flash memory controller aborts the access and returns a system bus error. Data accesses to the BAF region are not affected by the state of PFCR3[BAF_DIS].

29.6.3 Access pipelining

Accesses to the flash array may be pipelined by driving a subsequent access address and control signals while waiting for the current access to complete. Pipelined access requests are always run to completion and are not aborted by the flash memory controller. Flash access pipelining allows for improved performance by reducing the access latency seen by the AHB master. Access pipelining may be applied only to read cycles targeting the flash array. Address pipelining is enabled by setting the PFCR1[APC] field.

Access pipelining is only supported for normal flash access and is not supported for calibration overlay RAM fetches. On calibration reads, the setting of PFCR1[APC] is ignored.

NOTE

Not all combinations of PFCR1[APC] (Address Pipeline Control) and PFCR1[RWSC] (Read Wait State Control) settings are recommended or supported. Please refer to the device data sheet for guidance.

29.6.4 Line read buffers and prefetch operation

The AHB ports of the flash memory controller each contain a two-way set-associative mini cache, where each way contains four page buffers which are used to hold data read from the flash arrays. Each 256-bit buffer operates independently, and is filled using a single array access. The buffers are used for both prefetch and normal demand fetches.

Prefetch triggering is controllable on a per-master and access-type basis (see Platform Flash Configuration Register 1 (PFLASH_PFCR1)). Bus masters may be enabled or disabled from triggering prefetches, and triggering may be further restricted based on whether a read access is for instruction or data. A read access by the flash memory controller may trigger a prefetch to the next sequential line of array data on the cycle

following the request. The access address is incremented by 32-bytes, and a subsequent flash access is initiated. A flash array prefetch is initiated if the data is not already resident in a line read buffer. Prefetched data is always loaded into the least-recently-used buffer.

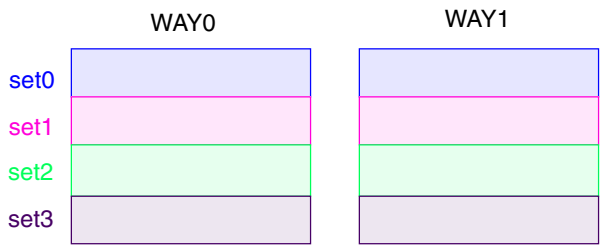


Figure 29-107. Flash memory controller 4-entry, 2-way mini-cache organization

Once the candidate line buffer has been selected, the flash array is accessed and read data loaded into the buffer. If the buffer load was in response to a miss, the buffer which was loaded is immediately marked as most-recently-used. If the buffer load was in response to a speculative fetch to the next-sequential line address after a buffer hit, *the recently-used status is not changed*. Rather, it is marked as most-recently-used only after a subsequent buffer hit.

This policy maximizes performance based on reference patterns of flash accesses and allows for prefetched data to remain valid when non-prefetch enabled bus masters are granted flash access.

Several algorithms are available for prefetch control which trade off performance for power. They are described in Platform Flash Configuration Register 1 (PFLASH_PFCR1). More aggressive prefetching may increase power due to the number of potentially discarded prefetches, but may increase performance by lowering average read latency.

For prefetching to occur, all of the following must apply:

1. PFCR{1,2}[P{0,1}_BFEN] must be set to ‘1’,
2. PFCR{1,2}[P{0,1}_PFLIM] must be non-zero, and
3. Either PFCR{1,2}[P{0,1}_IPFEN] or PFCR{1,2}[P{0,1}_DPFEN] must be asserted.

Refer to Platform Flash Configuration Register 1 (PFLASH_PFCR1) for a description of these controls.

29.6.5 Instruction/Data prefetch triggering

Port0 prefetch triggering may be enabled for instruction reads via the PFCR1[IPFEN] control field, while Port0 prefetching for data reads is enabled via the PFCR1[DPFEN] control field. Additionally, the PFCR1[PFLIM] must also be set to enable prefetching on Port0. Refer to Platform Flash Configuration Register 1 (PFLASH_PFCR1) for a description of these controls. Prefetches are never triggered by write cycles.

Port1 prefetch triggering may be enabled for instruction reads via the PFCR2[IPFEN] control field, while Port1 prefetching for data reads is enabled via the PFCR2[DPFEN] control field. Additionally, the PFCR2[PFLIM] must also be set to enable prefetching on Port1. Refer to Platform Flash Configuration Register 2 (PFLASH_PFCR2) for a description of these controls. Prefetches are never triggered by write cycles.

29.6.6 Per-Master prefetch triggering

Prefetch triggering may be controlled for individual bus masters. Refer to Platform Flash Configuration Register 1 (PFLASH_PFCR1) for a description of these controls.

29.6.7 Buffer allocation

Allocation of the line read buffers is controlled via the PFCR3 control register, specifically the line buffer configuration ({P0,P1}_WCFG) field. Refer to Platform Flash Configuration Register 3 (PFLASH_PFCR3). The buffers can be organized as a "pool" of available resources (with both ways within a given set) or with a fixed partition between ways allocated to instruction or data accesses. For the fixed partitions, way 0 is allocated for instruction fetches and way 1 for data accesses.

29.6.8 PFlash calibration remap support

The flash memory controller supports calibration development by providing a remapping function to route flash accesses to overlay RAM and on-chip System RAM that can be used as overlay RAM.

Recall this family of devices includes a comprehensive set of features including dedicated overlay RAM arrays to aid calibration and debug. The overlay function supports the following features:

- Can be mapped over internal flash memory

    - Allows calibration of constant data without requirement for additional external RAMs and calibration memory interfaces
- Can be used to hold device debug trace stream
    - Allows limited trace based debug without requirement for complex debug tool hardware and access to high speed trace port
- Protected with error detection and correction mechanism
- Accesses to all overlay RAMs can achieve the same timing as accesses to internal flash
    - Timing for calibration accesses to a particular overlay RAM is only guaranteed if it is not being used as a destination for trace streaming at the same time
    - Timing for calibration accesses to overlay RAM is only guaranteed if RWSC is set sufficiently high to cover the intrinsic burst access incurred when fetching a full page of calibration data
- Support for up to 32 distinct calibration remap regions.
- Protection against accidental enable and reconfiguration of calibration remap function :
    - Two stage mechanism to enable remap, using both global and individual calibration region descriptor enable bits
    - Optional configuration of calibration region descriptors as redundant pairs, allowing detection of accidental region reconfiguration.
- Support for overlay remapping to unique calibration regions on a per-master basis.

The calibration remap function supports the following overlay targets:

- An internal overlay RAM is included on all standard production devices.
- An extended overlay RAM is an additional memory that is only available on special ED devices.
- A portion of the system RAM can be used for overlay.

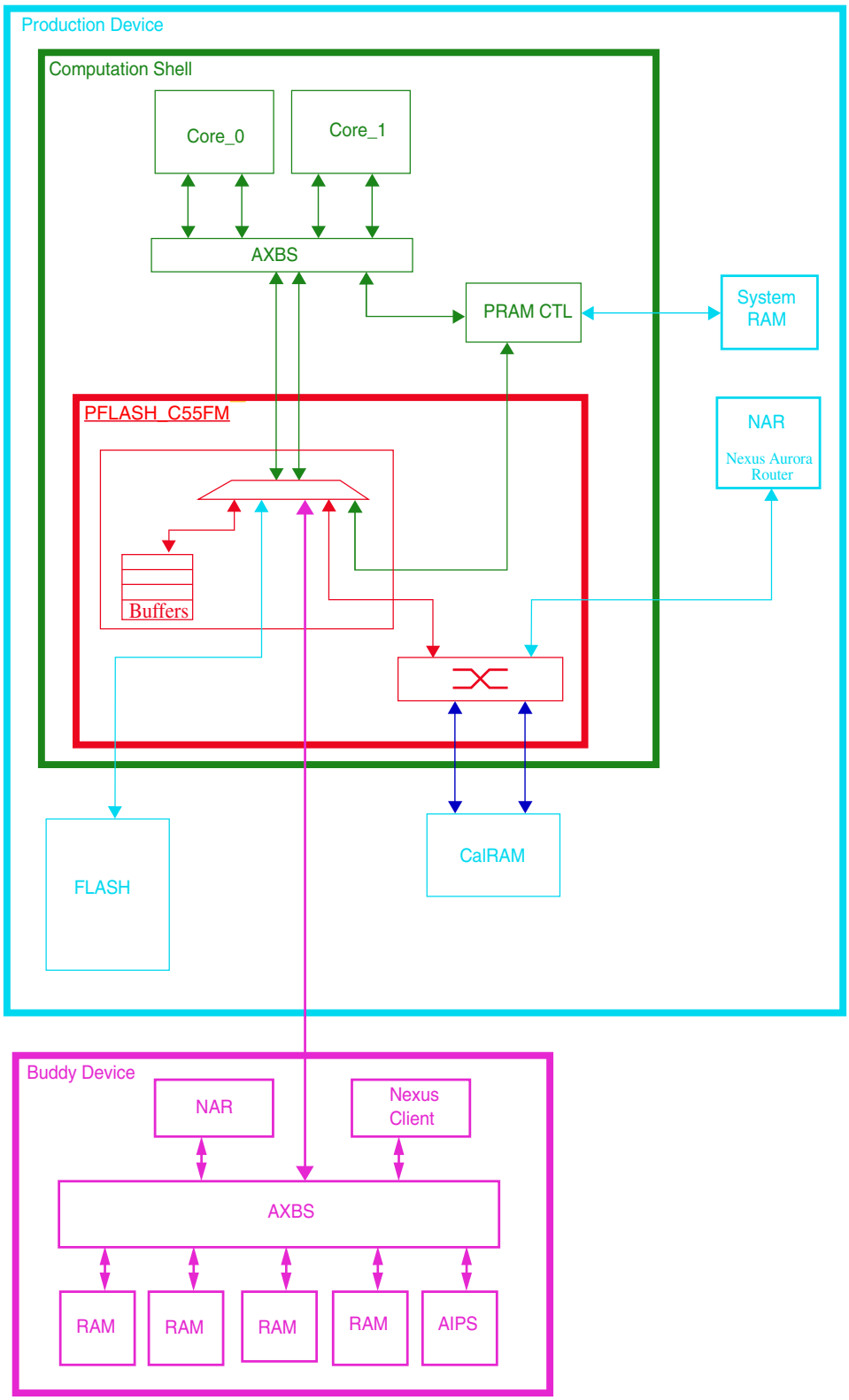


Figure 29-108. Flash memory controller and associated calibration remap targets

29.6.8.1 PFlash calibration remap to RAM

The overlay RAM is selected based on the translated physical address.

To better understand the functionality of the calibration remap, consider more detailed block diagrams, one showing the overall structure of the calibration remap function and another showing more detail on the actual remap logic. Recall the flash memory controller implements a hardware interconnection matrix connecting the three input ports and four destination memories. See Figure 29-109.

The following figure illustrates the calibration remap connection matrix.

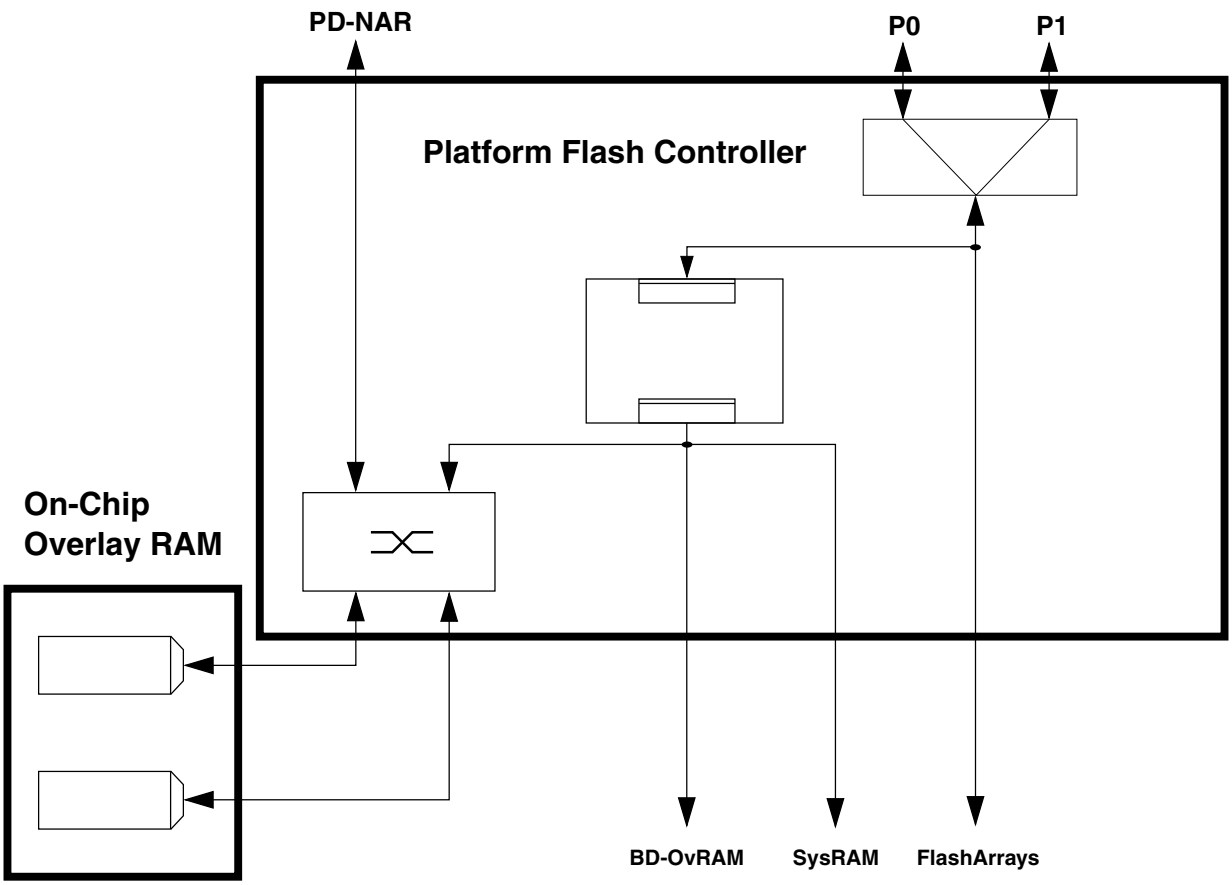


Figure 29-109. PFlash calibration remap connection matrix

Note the 64-bit on-chip overlay RAM is split into 2 equal-sized memories to allow concurrent accesses for calibration data from the P0, P1 (where these are the crossbar slave input ports) plus trace data writes from the Production Device Nexus Aurora Router (PD-NAR).

The translated physical address defines the destination memory. Table 29-107 shows the base addresses for the calibration data memories as defined by the system memory map.

Table 29-107. Calibration data memory base addresses

Calibration Memory	Base Address	Possible Access Sources
Flash Arrays	0x0{8,9,A}--_----	P0, P1
System RAM	0x40{0,1}-_----	P0, P1
On-chip Overlay RAM	0x0D00_----	P0, P1, PD-NAR
Buddy Device (Extended) Overlay RAM	0x0C{0,1}-_----	P0, P1

Note the trace data from the Nexus Aurora Router (PD-NAR) can only be routed to the on-chip overlay RAM; there is a different connection path in the Buddy Device from the BD-NAR into the extended overlay RAM.

The system RAM connection from the calibration remap logic is implemented via a private 64-bit connection to the platform RAM controller.

Accesses to the flash arrays always use the logical, non-remapped system address. Calibration data store references to the system RAM, on-chip or buddy device (extended) overlay RAM always use the physical, non-remapped system address. Conversely, calibration data load references can be accessed using a translated physical address or, if provided by the requesting master, the physical, non-remapped system address. Byte-, halfword-, word-, and doubleword-sized data store references to the on-chip overlay RAM are supported. Byte-, halfword-, word-, and doubleword-sized store references to the buddy device (extended) overlay RAM are supported.

The read data interface of system RAM, on-chip overlay RAM, and buddy (extended) device is 64 bits wide for each, whereas the flash read data interface is 256 bits wide. As a result, a remapped calibration load access initiates a four-beat burst sequence to mimic the throughput of a single flash access. There is a direct relationship between the operating frequency range and the required number of wait states to correctly sample flash read data. By contrast, a remapped calibration load access has a fixed, intrinsic 4-beat latency, where the access time of each individual beat depends on the overlay target - system RAM, on-chip overlay RAM or buddy device. The access time when fetching data from the buddy device presents the longest latency. Therefore, in order for an initial, random calibration load access to mimic the flash access time, PFCR1[RWSC] read wait state setting must be programmed to a value which sufficiently covers the intrinsic access time to complete a four-beat burst sequence from the slowest available overlay target at the specified device operating frequency. Expressed mathematically:

$$(\mathrm{RWSC}+2) \geq \left(\frac{\mathrm{IntrinsicFlashAccessTime}}{\mathrm{OperatingFrequencyClockPeriod}}+2\right) \geq (\mathrm{OverlayRAMTransferBurstLatency})$$

Figure 29-110. Access time

Notice the overall latency on a random, initial flash access that does not hit in the prefetch mini-buffer incurs an additional cycle latency beyond the intrinsic flash access time. This cycle accounts for pipeline stalls incurred by the flash controller state machine. The formula above is based on specified intrinsic flash access times of the C55FMC array at 150 °C. Calibration data that is loaded in the flash controller's mini-cache is returned with single-cycle latency.

The memory-mapped calibration remap (region) descriptor (CRD*n*) registers define the required address translation. Each calibration remap descriptor includes the logical base address, the translated physical base address and a region size (plus other control bits). The calibration remap evaluation is initiated when the flash controller is presented with a flash access request from any system bus master to a location in the mirrored flash address space. Conversely, references using the associated non-mirrored flash system address are not evaluated for calibration remapping. Generally, not all areas of the flash memory map are available for calibration remapping. EEPROM, BAF and UTEST are usually not available for calibration remapping. Flash access requests which present an address targeting EEPROM, BAF or UTEST will bypass calibration remap evaluation and return flash data.

The address translation mechanism requires the logical and physical base addresses of the region be aligned on 0-modulo-size boundaries. See Figure 29-111.

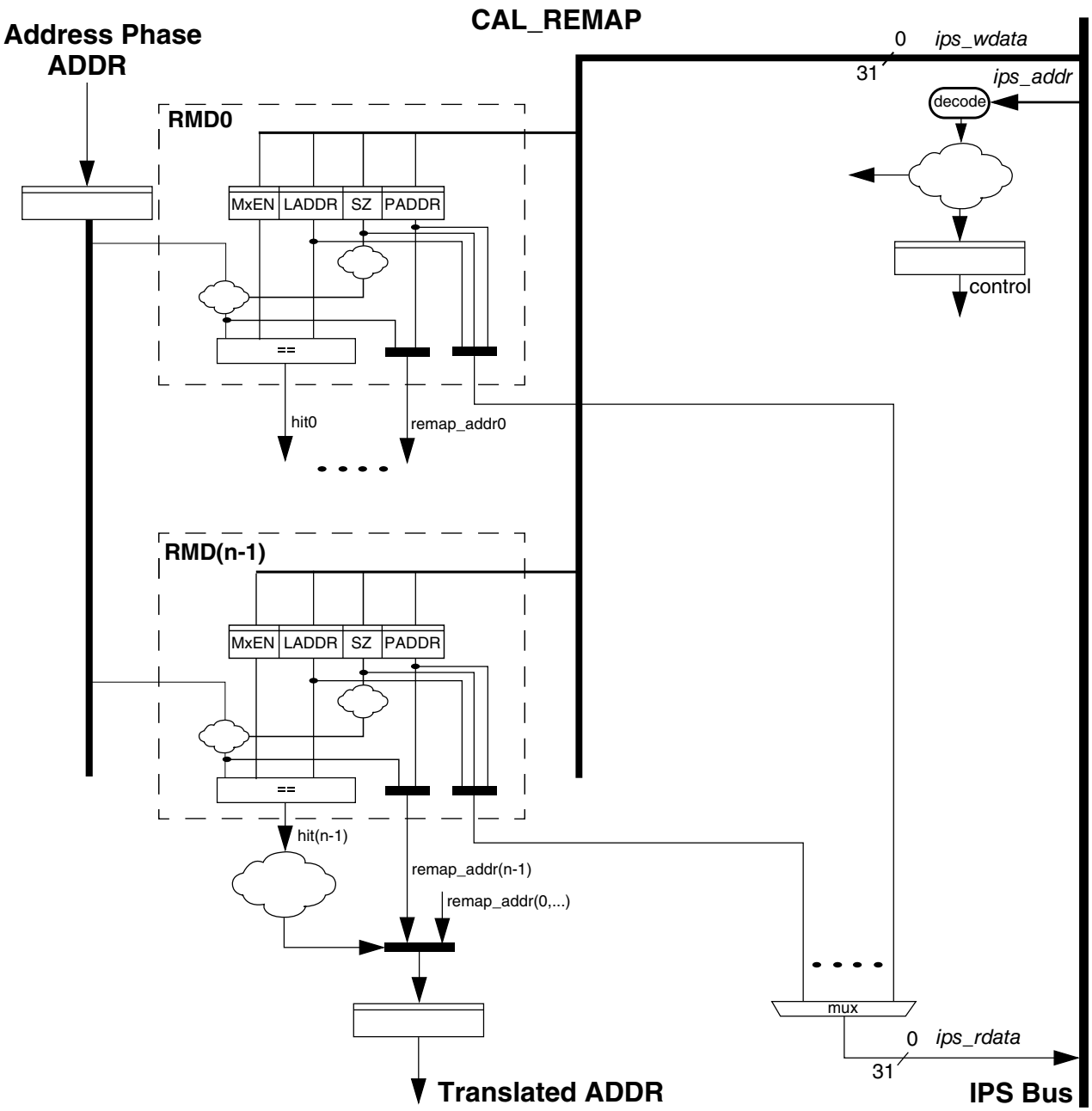


Figure 29-111. PFlash calibration remap logic detail

Each instantiation of the basic calibration remap function {RMD0,…, RMD(*n*-1)} in Figure 29-111 includes the three descriptor registers containing the logical start address (LADDR in the figure), the physical start address (PADDR), region size (SZ), and per-master enable bits (MxEN). The region size is decoded to create the appropriate address bit enables which are then applied to the registered logical flash address. An equality comparator then compares the adjusted logical flash address against the logical address from the calibration remap descriptor to determine the region "hit". The hit determination is further qualified based on the bus master number that initiated the flash access. At the same time, the logical flash address and the physical address defined in the calibration

remap descriptor are merged to form the translated address to be used in the event of a calibration region hit. Aside from the individual instantiations of the calibration remap descriptors, this logic examines all the descriptor hit indicators to select the appropriate translated address.

In the absence of a region "hit" data is returned from the flash at the provided logical address.

It should be noted in the event of *overlapping remap regions*, the calibration remap descriptor with the largest number is used for the address translation, that is, if a logical flash address hits in multiple descriptors, say CRDx and CRDy, the translated address from CRDy is used, given y > x.

In general, the calibration remap logic operates only on flash *data* accesses. However, flash instruction references can optionally remapped while in debug modes of operation to support the use of software breakpoints when PFCRC[IRMEN] is set.

29.6.9 Reliability considerations

This section summarizes mechanism to enhance the reliability of flash by correcting and detecting faults.

29.6.9.1 e2eECC End-to-End ECC

29.6.9.1.1 e2eECC and flash accesses

There are multiple complications associated with the use of the standard e2eECC algorithm with flash memory. The basic issues involve: the default state of a flash block that is erased (all ones) and the fact that programming an erased flash involves changing the state of memory bit from a logical 1 to a logical 0. These factors require that the system level e2eECC must be modified on references to the flash memory in three ways.

First, the all-ones codeword, since it reflects the state of an erased flash location, is treated as a valid error free encoding; in particular, it is required that the checkbits associated with the all all-ones value and the all zeroes data value are required to be the same with a checkbit value of 0xFF. Second, since an erased flash block generates an all-ones data for all locations, the address field cannot be included in the ECC code used by the flash. The flash array implements special address re-encoding logic based on the decoded array address to protect against address faults. Third, there are additional ECC implications associated with the flash memory and its support for EEPROM emulation. Lastly, there are ECC implications associated with the storage and retrieval of overlaid calibration data.

In summary, the basic operation of an NVM bit cell impacts the e2eECC algorithm used in the flash memory in multiple ways. The required ECC adjustments are handled by the platform flash controller before data is written to the flash array(s) or applied to read data accessed from the array(s). These adjustments are two-fold. Consider a flash write during a programming event:

1. Remove the address field from the ECC codeword by XOR'ing the address calculation of the H-matrix from the checkbits.
2. Invert the resulting 8-bit ECC checkbit vector. This step is required to support the all-ones erased state.

On a flash read operation, a similar but "reversed" set of steps are performed as the data is driven into the system bus interconnect:

1. Invert the 8-bit ECC checkbit vector read from the flash.
2. Factor the address field into the ECC checkbits by XOR'ing the address calculation of the H-matrix.

Consider the following flash ECC example. Let the flash block containing address 0x00345678 be erased. The following is the sequence of operations:

1. Read address 0x00345678. Flash array returns fl_rdata = 0xFFFFFFFF_FFFFFFFF, fl_cdata = 0xFF. The flash memory controller calculates the addr_chkbit = 0xC4 using the H-matrix; it inverts the fl_cdata vector to 0x00 and factors in the addr_chkbit = 0xC4 to produce the following valid e2eECC codeword:

   addr = 0x00345678, rdata = 0xFFFFFFFF_FFFFFFFF, rchkbit = 0xC4 (0x00 ^ 0xC4)

2. Program address 0x00345678 with data = 0xBABEFACE_DEADBEEF. The initiating bus master calculates the e2eECC codeword as:

   addr = 0x00345678, wdata = 0xBABEFACE_DEADBEEF, wchkbit = 0xE3

   Before performing the interlock write to the flash array, the flash memory controller adjusts the chkbit value as fl_wchkbit = 0xD8 (0xE3 ^ 0xff ^ 0xC4) by toggling the original write checkbits and then factoring in the addr_ecc. Accordingly, the codeword stored in the flash array is:

   fl_data = 0xBABEFACE_DEADBEEF, fl_checkbit = 0xD8

3. Read address 0x00345678. The flash array returns the stored codeword, fl_rdata = 0xBABEFACE_DEADBEEF, fl_rchkbit = 0xD8. The flash memory controller inverts the checkbit vector and factors in the address checkbits (0xD8 ^ 0xFF ^ 0xC4) to produce:

   addr = 0x00345678, rdata = 0xBABEFACE_DEADBEEF, rchkbit = 0xE3

This read codeword matches the original e2eECC codeword generated by the write.

Malfunction of the ECC logic described above may result in the corruption of the SEC/DED event reporting. The flash memory controller performs EDC after ECC check on all flash transactions. If a mismatch is detected, indicating a failure in the ECC logic, the event is reported to the device fault collection module.

29.6.9.2 Flash address generation check

Fault detection coverage of the address and data are handled by ECC performed within the flash and e2eECC performed at the master. Fault detection coverage of the address path and control within the flash memory controller rely on a feedback path between the flash controller and flash. Recall on a requested access to flash, the flash memory controller must decode the system AHB bus signals to generate the corresponding flash interface signals to invoke a flash lookup. In addition to providing the requested read data, the flash also provides output sidebands reflecting the encoded address and block selects used to perform the actual row lookup.

This sideband information is used by the flash memory controller to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the flash memory controller or the transmission path between the flash memory controller and flash array, the event is forwarded to the device fault collection module and the corresponding buffer is invalidated.

29.6.9.2.1 On-Chip Overlay RAM feedback check

The integrity of the address and data on an overlay RAM transaction is covered by e2eECC check performed by the requesting master. Fault detection coverage of the address path and control within the flash memory controller is handled by a transaction monitor which verifies the integrity of the transactions between the flash memory controller and the on-chip overlay RAM. The function of the transaction monitor relies on a feedback path between the flash memory controller and the on-chip overlay RAM, wherein the RAM provides latched address and control feedback outputs as an indication of received inputs when a RAM access is initiated. This feedback information is used by the flash memory controller transaction monitor to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the flash memory controller or the transmission path between the flash memory controller and on-chip overlay RAM array, the event is forwarded to the Fault Collection and Control Unit (FCCU).

29.6.9.2.2 Reliability considerations on overlay accesses

Because calibration write accesses are always presented with the physical, non-remapped system address, the write data checkbits presented by the master are calculated based on the physical overlay RAM system address. On a subsequent overlay read, the ECC checkbits must be manipulated to replace the physical overlay RAM address contribution with the logical flash address contribution. This is required to satisfy the e2eECC check at the originating master.

In addition, the flash memory controller can be configured to treat calibration RAM space as a safety-critical component. By programming PFCRCR[SAFE_CAL]=1, the flash memory controller treats the set of available calibration region descriptors as a replicated pair, with each set containing half the total number of available calibration region descriptors.

If there are 32 calibration region descriptors, CRD0 – CRD31, and PFCRCR[SAFE_CAL]=1, the flash memory controller treats CRD0 – CRD15 and CRD16 – 32 as replicated sets of 16 calibration region descriptors, where:

- CRD0 and CRD16 are configured identically
- CRD1 and CRD17 are configured identically
- CRD2 and CRD18 are configured identically
- ...
- and CRD15 and CRD31 are configured identically

Expressed more generally, the contents of CRD[n] must be identical to the contents of CRD[(N/2) + n], where N is the total number of region descriptors and $0 \le n \le (N/2) - 1$.

Note

It is the user's responsibility to establish redundant calibration region descriptors by programming the associated pairs of CRDs.

Incoming flash read requests are compared simultaneously against the calibration regions defined in CRD0 to CRD[(N/2) – 1] and the calibration regions defined in CRD[N/2] to CRD[N – 1]. Due to the enabled redundancy, the parallel overlay remap hardware structures should evaluate to symmetric results. If a mismatch is detected, the event is reported to the FCCU and calibration remapping is not performed. Instead, data is returned from the flash.

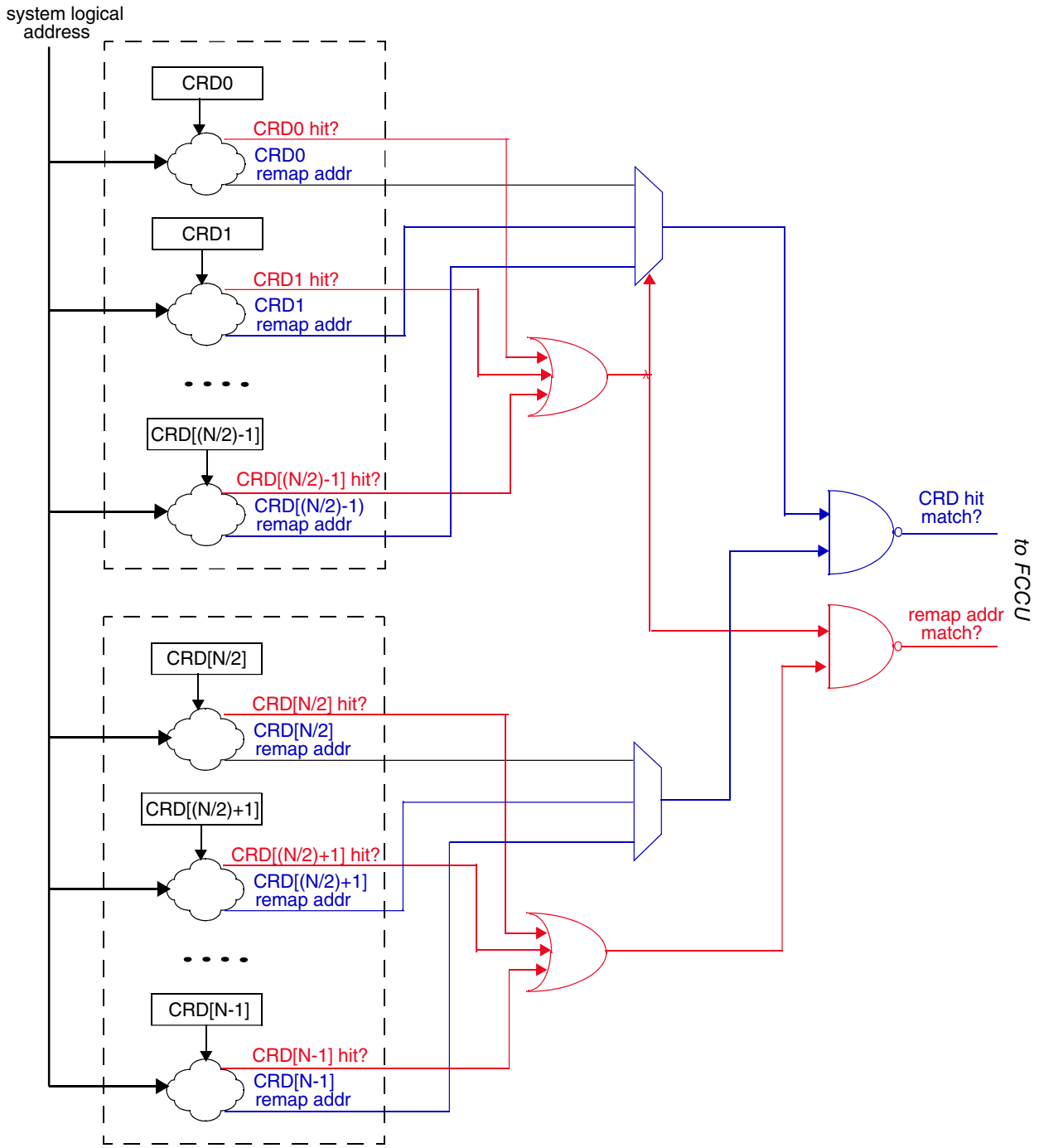


Figure 29-112. Safety-critical calibration remap datapath with redundancy

29.6.10 ECC on data flash accesses

When PFCR3[DERR_SUP] is enabled, ECC errors on data flash blocks are handled specially.

ECC events detected on accesses to data flash blocks are suppressed from being reported to the Memory Error Management Unit (MEMU).

In the event of a single-bit correction, the corrected data and checkbits are returned to the requesting master, and the single-bit correction event is suppressed from being reported to the MEMU.

In the event of a non-correctable error detection, a fixed, illegal opcode value is returned to the requesting master along with the associated ECC checkbits as determined by the requesting address, and the non-correctable error event is suppressed from being reported to the MEMU. For non-correctable error detection, the flash memory controller returns a value of 1555_1555h to the requesting master.

NOTE

EEPROM should be avoided for storage of executable code.

29.6.11 Array integrity considerations

During an array integrity sequence, the flash memory array ignores any incoming read requests. When a flash array integrity check is in progress, the flash memory controller terminates all flash access requests with an error. More specifically, it aborts the incoming flash access requests and terminates the system bus transfer with an error.

Chapter 30
Embedded Flash Memory (c55fmc)

30.1 Embedded Flash Memory (c55fmc) chip-specific module information

This section describes the configuration of this module specific to the MPC5746R.

30.2 C55FMC_LOCK*n* register bit mapping

See the attached Flash Block Assignment spreadsheet for the C55FMC_LOCK*n* register bit mapping.

30.3 C55FMC_SEL*n* register bit mapping

See the attached Flash Block Assignment spreadsheet for the C55FMC_SEL*n* register bit mapping..

30.4 Embedded Flash Memory (c55fmc) register reset values

Table 30-1 lists chip-specific register reset values specific to the MPC5746R. Any registers not listed have the reset values shown in their respective sections of this chapter.

Table 30-1. Chip specific c55fmc flash memory reset values for the MPC5746R

Register	Reset Value
Module Configuration Register (C55FMC_MCR)	0000_0600b
Extended Module Configuration Register (C55FMC_MCRE)	0700_244Bb

Table continues on the next page...

Table 30-1. Chip specific c55fmc flash memory reset values for the MPC5746R (continued)

Register	Reset Value
Over-Program Protection 0 register (C55FMC_OPP0)	Dependent on customer settings[1, 1]
Over-Program Protection 1 register (C55FMC_OPP1)	Dependent on customer settings[1]
Over-Program Protection 2 register (C55FMC_OPP2)	Dependent on customer settings[1]
Over-Program Protection 3 register (C55FMC_OPP3)	Dependent on customer settings[1]

1. The reset value of the C55FMC_OPP*n* registers, for each flash memory block mapped to a bit within each register, reflects whether the block has been marked as One Time Programmable (OTP). The C55FMC_OPP*n* registers have the same bit-to-block mapping as the C55FMC_LOCK*n* registers. A 1 indicates a block has been marked as OTP. The reset values of the C55FMC_OPP*n* are similar to the reset values of the TDM_OTPEN*n* registers in the Tamper Detection Module (TDM).

30.5 Introduction

The primary function of the embedded flash memory is to serve as electrically programmable and erasable non-volatile memory (NVM) that may be used for instruction or data storage.

The following figure shows the top-level diagram and functional organization of the flash memory unit.

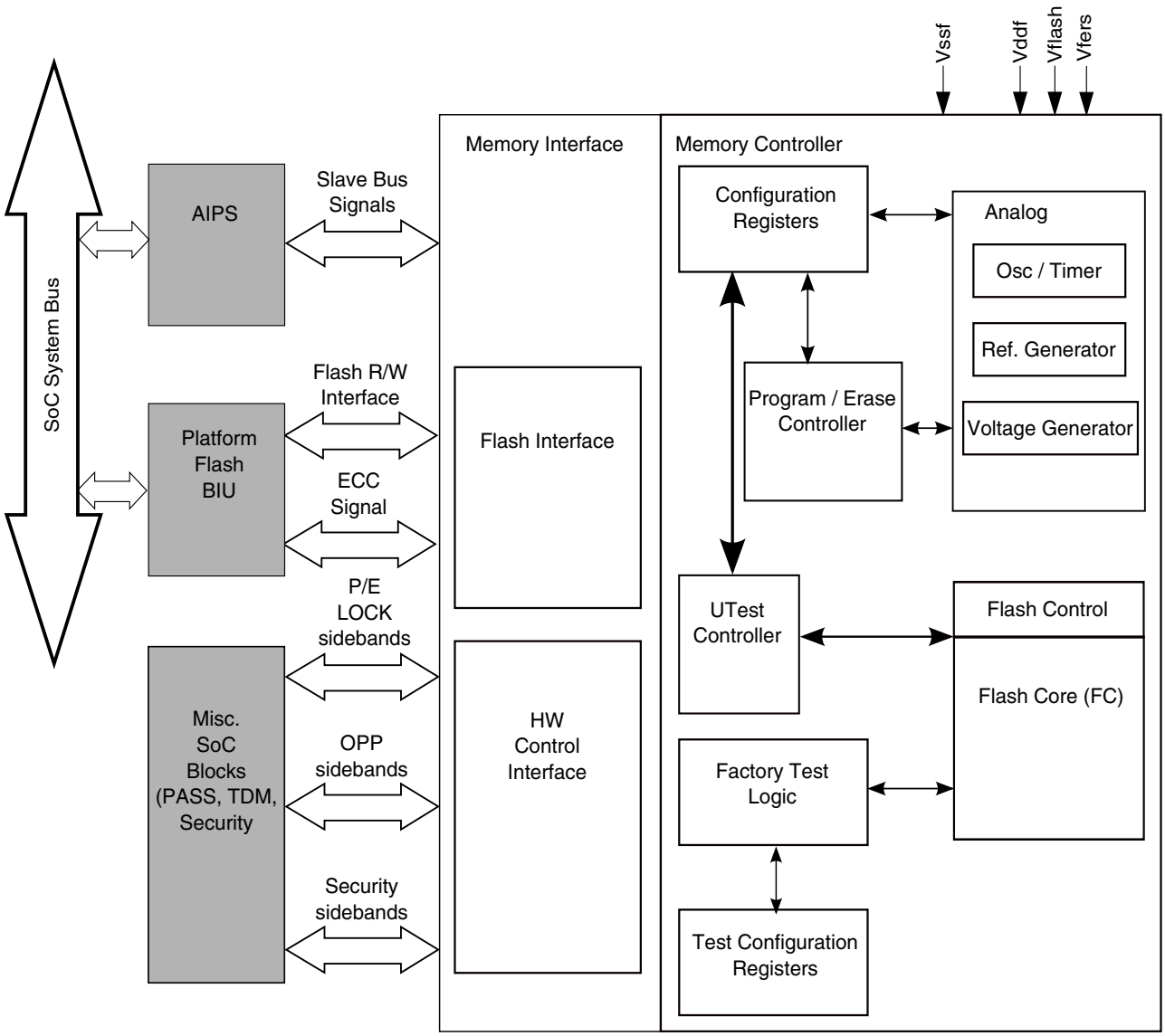


Figure 30-1. c55fmc block diagram

30.5.1 Overview

The embedded flash memory is addressable by word (32 bits) or double word (64 bits) for program operation, and page (256 bits) for read operation. Multiple word or double-word writes may be done to the flash memory to fill up the program page buffer (256 bits), enabling page programming (256 bits, requiring 4 double-word writes) and quad-page programming (1024 bits, requiring 16 double-word writes). Flash memory reads always return 256 bits, although read page buffering may be performed by the Bus Interface Unit (BIU).

For more details on the embedded flash memory architecture and features, see Functional Description.

30.5.2 Features

The embedded flash memory includes these distinct features:

- Test information stored in a dedicated nonvolatile block (referred to as the UTest block)
- OTP space made available in the UTest block
- Read page size of 256 bits (8 words)
- ECC with single-bit correction, double-bit detection (all 1's valid)
- Quad Page programming (64-bit granularity)
- Software programmable block program/erase restriction control
- Erasing of selected block(s)
- Independent programming of the UTest NVM block
- Embedded hardware program and erase algorithms
- Support for reading while writing when the accesses are to different partitions
- Erase suspend, program suspend, and erase-suspended program
- UTest mode (user-accessible test modes) including Array Integrity and Margin Read
- Triple Voted Flops for flash functions requiring high reliability, e.g., internal trimming, redundancy, and mode control

30.5.3 Modes of operation

Following is a brief description of the embedded flash memory operating modes.

- *User mode* is the default operating mode of the embedded flash memory. In this mode, it is possible to read and write registers, read and interlock write the memory array, program the memory array, and erase the memory array. In this mode program and erase operations are initiated by doing array and register writes, and are controlled by an internal state machine.
- *UTest mode* is a tiered test mode strategy in which a portion of the factory test modes are made available. This mode is protected but accessible.

30.6 UTest NVM block

The UTest NVM block may be enabled by the BIU. When the UTest NVM space is enabled, all operations are mapped to the UTest NVM block. User-mode programming of the UTest NVM block is enabled only when MCR[PEAS] is high. The UTest NVM block is an OTP block (assuming Test Mode Disable Seal is written) - thus erase is not allowed.

The UTest NVM block supports RWW, and is grouped with the blocks in partition 0.

The UTest NVM block may be locked against program by using the lock registers.

Programming of the UTest NVM space has restrictions that are similar to those of the main space in terms of how ECC is calculated. Only one program operation is allowed per 64-bit ECC segment.

The UTest NVM block contains specified data that is needed for embedded flash memory or SoC features.

30.7 Test mode disable seal

Also included in the UTest NVM block is a mechanism to disable factory entry into Test mode. Extreme care must be taken when using this feature, as blocks that are selected to be protected in this method will not be able to have possible failures analyzed by factory failure analysts. Once sealed, the UTest block becomes OTP (Erase Locked, and Over Program Protection enabled). The UTest Block also is not accessible in Test mode.

Protection of this sort prevents all high-voltage operations to the flash executed by the internal state-machine, as well as reads through this state machine, and reads through the Array Integrity state machine when using Test mode interfaces.

UTest operations Margin Read and Array Integrity are also protected by preventing MISR updates on blocks selected for protection, although reads will still be done and single bit corrections and double bit detections will be logged. Protection through other interfaces is protected by normal User mode protection mechanisms, and are device-specific.

The method to disable factory entry into Test mode is to first program the Test Mode Disable Seal location to be 0x2D3C_4B5A. Once the next reset is asserted, Test mode will be disabled.

It is possible to create a password to enable factory entry into test mode. This can be programmed into the Test Mode Disable Override Passcode. The passcode may not be 0x0000_0000, 0xFFFF_FFFF, or 0x5555_5555. These are all invalid passcodes and will not be accepted to override. If it is desired by the customer that override never be possible, one of the three invalid passcodes should be put into this location. Passcode may be entered to authenticate entry (if enabled) by doing a 32-bit register write to register address 0x90. The UTest NVM block is always protected even if the Test Mode Disable Seal password is written. UTest operations Margin Read and Array Integrity are always protected even if the Test Mode Disable Seal password is written.

Only blocks selected in the Test Mode Disable Block Select field(s) are controlled by the Test Mode Disable feature. Thus it is possible for customers to selectively pick blocks that have this type of protection, and will not be eligible for factory failure analysis. Bits programmed to 0 in the Test Mode Disable Block Select field(s) will designate blocks that are controlled by the Test Mode Disable feature. If either Test Mode Disable Block Select - 0 or Test Mode Disable Block Select - 1 have a 0 programmed, that block will be protected. In order to provide two opportunities to select blocks for the Test Mode Disable, two regions are available, Test Mode Disable Block Select - 0 and Test Mode Disable Block Select - 1. The bit of these two regions are logically ANDed, to define which blocks are effectively selected for the Test Mode Disable feature. The Block Select field is organized as follows, and aligns with how blocks are defined in the LOCK registers.

Table 30-2. Test mode disable block select

Block	Bits used to select blocks
Low Block Disable	Data[13:0]
Unused	Data[15:14]
Mid Block Disable	Data[31:16]
High Block Disable	Data[47:32]
256K Block Disable	Data[95:48]

30.8 Memory map and register definition

The embedded flash memory map consists of a flash memory array (which includes main array space and UTest NVM space) and a region of registers associated with the programming model that enable flash memory array operation and modification.

The address space consists of 16 KB, 32 KB, 64 KB, and 256 KB blocks..

C55FMC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (C55FMC_MCR)	32	R/W	0000_0600h	30.8.1/850
8	Extended Module Configuration Register (C55FMC_MCRE)	32	R	See section	30.8.2/855
10	Lock 0 register (C55FMC_LOCK0)	32	R/W	BFFF_FFFFh	30.8.3/858
14	Lock 1 register (C55FMC_LOCK1)	32	R/W	0000_FFFFh	30.8.4/860
18	Lock 2 register (C55FMC_LOCK2)	32	R/W	FFFF_FFFFh	30.8.5/860
1C	Lock 3 register (C55FMC_LOCK3)	32	R/W	0000_FFFFh	30.8.6/861
38	Select 0 register (C55FMC_SEL0)	32	R/W	0000_0000h	30.8.7/862
3C	Select 1 register (C55FMC_SEL1)	32	R/W	0000_0000h	30.8.8/863
40	Select 2 register (C55FMC_SEL2)	32	R/W	0000_0000h	30.8.9/864
44	Select 3 register (C55FMC_SEL3)	32	R/W	0000_0000h	30.8.10/ 865
50	Address register (C55FMC_ADR)	32	R	0000_0000h	30.8.11/ 866
54	UTest 0 register (C55FMC_UT0)	32	R/W	0000_0001h	30.8.12/ 868
58	UMISR register (C55FMC_UM0)	32	R/W	0000_0000h	30.8.13/ 871
5C	UMISR register (C55FMC_UM1)	32	R/W	0000_0000h	30.8.13/ 871
60	UMISR register (C55FMC_UM2)	32	R/W	0000_0000h	30.8.13/ 871
64	UMISR register (C55FMC_UM3)	32	R/W	0000_0000h	30.8.13/ 871
68	UMISR register (C55FMC_UM4)	32	R/W	0000_0000h	30.8.13/ 871
6C	UMISR register (C55FMC_UM5)	32	R/W	0000_0000h	30.8.13/ 871
70	UMISR register (C55FMC_UM6)	32	R/W	0000_0000h	30.8.13/ 871
74	UMISR register (C55FMC_UM7)	32	R/W	0000_0000h	30.8.13/ 871
78	UMISR register (C55FMC_UM8)	32	R/W	0000_0000h	30.8.13/ 871
7C	UMISR register (C55FMC_UM9)	32	R/W	0000_0000h	30.8.14/ 872
80	Over-Program Protection 0 register (C55FMC_OPP0)	32	R	See section	30.8.15/ 873
84	Over-Program Protection 1 register (C55FMC_OPP1)	32	R	See section	30.8.16/ 874
88	Over-Program Protection 2 register (C55FMC_OPP2)	32	R	See section	30.8.17/ 875
8C	Over-Program Protection 3 register (C55FMC_OPP3)	32	R	See section	30.8.18/ 875
90	Test Mode Disable Password Check register (C55FMC_TMD)	32	R/W	0000_0000h	30.8.19/ 876

30.8.1 Module Configuration Register (C55FMC_MCR)

NOTE

1. A number of Module Configuration Register (MCR) bits are locked against write by other bits. These locks are discussed in relationship to each bit in this section. Simultaneously writing bits which lock each other out is also discussed in Functional Description.
2. See Functional Description for information about simultaneous MCR writes, and priority levels.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RVE	RRE	AEE	EEE	0											
W	w1c	w1c	w1c	w1c												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EER	RWE	SBC	0	PEAS	DONE	PEG	PECIE	0			PGM	PSUS	ERS	ESUS	EHV
W	w1c	w1c	w1c													
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

C55FMC_MCR field descriptions

Field	Description
0 RVE	Read Voltage Error RVE provides information on previous reads monitoring the read pump voltage. If the read voltage is detected to be out of range, this bit is set to indicate that previous reads requested may have been

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
	corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without voltage issues 1 A previous read may have been corrupted due to read voltage being out of range
1 RRE	Read Reference Error RRE provides information on previous reads monitoring the read reference. If the read reference is detected to be out of range, this bit is set to indicate that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without reference issues 1 A previous read may have been corrupted due to read reference being out of range
2 AEE	Address Encode Error AEE provides information on previous reads monitoring the address encode feature. On every read request to the flash, the incoming address is compared to an encoded address (row, column, and block) coming back from the memory array using the read data sense amplifier timing. If these two values do not match (zero selected, multiple selected, wrong selected), or the timing is incorrect, an address encode error will be recorded. If an address encode mismatch is detected, this bit is set to indicate that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without address encode mismatches 1 A previous read may be corrupted based on address encode mismatch
3 EEE	ECC after ECC Error EEE provides information on previous reads monitoring the ECC after ECC feature. On every read request to the flash, ECC is recalculated serially, and if there is a mismatch between the ECC calculations (taking into account corrections or detections) a late error will be reported. If an ECC after ECC mismatch is detected, this bit is set to indicate that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without ECC after ECC mismatches 1 A previous read may be corrupted based on ECC calculation errors
4–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 EER	ECC Event Error This bit provides information on previous reads. If a double bit detection occurred, the EER bit is set to a '1'. This bit must then be cleared, or a reset must occur before it returns to a 0 state. This bit may not be set by software. In the event of a single bit detection and correction, this bit is not set. If EER is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of EER) are correct. Since this bit is an error flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring normally. 1 An ECC Error occurred during a previous read.
17 RWE	Read-While-Write Event Error This bit provides information on previous read-while-write (RWW) reads. If an RWW error occurs, this bit is set to 1. The bit must then be cleared, or a reset must occur before it returns to a 0 state. This bit may not be written to a 1. If RWE is not set, or remains 0, this indicates that all previous RWW reads (from the last

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
	reset, or clearing of RWE) are correct. Since this bit is an error flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring normally. 1 A read-while-write error occurred during a previous read.
18 SBC	Single Bit Correction SBC provides information on previous reads, if the SBCE is set. If a single bit correction occurred, the SBC bit is set to a 1. This bit must then be cleared, or a reset must occur before this bit returns to a 0 state. If SBC is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of SBC) did not require a correction. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without corrections. 1 A Single Bit Correction occurred during a previous read.
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 PEAS	Program Access Space PEAS indicates which space is valid for program or erase operations — either main array space or UTest NVM space. It should be noted that if the flash is sealed, and the Test Mode Disable Seal is programmed, erase is not allowed on the UTest NVM space. PEAS = 0 indicates the main address space is active for all program or erase operations. PEAS = 1 indicates the UTest NVM address space is active for program or erase (if applicable) operations. The value in PEAS is captured and held when the UTest NVM block is enabled with the first interlock write done for program operations(and erase operations if the flash is unsealed). The value of PEAS is retained between sampling events (in other words, subsequent first interlock writes). The value in PEAS may be changed during erase-suspended program, and reverts back to its original state once the erase-suspended program is completed. PEAS is read only. 0 UTest NVM address space is disabled for program/erase and main address space enabled. 1 UTest NVM address space is enabled for program/erase and main address space disabled.
21 DONE	State Machine Status DONE indicates whether the embedded flash memory is performing a high voltage operation. DONE is set to a 1 on termination of the embedded flash memory reset. DONE is read only. DONE is set to a 1 at the end of program and erase high voltage sequences. **NOTE:** This bit transitions from a 0 to 1 during reset and remains at 1 after reset. 0 Flash memory is executing a high voltage operation. 1 Flash memory is not executing a high voltage operation.
22 PEG	Program/Erase Good The PEG bit indicates the completion status of the last flash memory program or erase sequence for which high voltage operations were initiated. The value of PEG is updated automatically during the program and erase high voltage operations. Aborting a program/erase high voltage operation causes PEG to be cleared, indicating the sequence failed. PEG is set to a 1 when the embedded flash memory is reset. PEG is read only. The value of PEG is valid only when PGM = 1 or ERS = 1 and after DONE transitions from 0 to 1 due to an abort or the completion of a program/erase operation. PEG is valid until PGM/ERS makes a 1 to 0 transition or EHV makes a 0 to 1 transition. The value in PEG is not valid after a 0 to 1 transition of DONE caused by PSUS or ESUS being set to logic 1. If PGM and ERS are both 1 when DONE makes a qualifying 0 to 1 transition the value of PEG indicates the completion status of the PGM sequence. This happens in an erase-suspended program operation.

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
	PEG will be 0 if an Erase-Suspended Program is attempted to a block that has been selected for Erase. PEG will also be 0 in the event of an error in the interlock write for program only. An error in the interlock write occurs when the ECC value provided to the flash memory does not match the ECC calculated within the flash memory. In the event of this, high voltage will not be applied to the array, and PEG will indicate a failed program. **NOTE:** 1. If program or erase operations are attempted on blocks that are locked, the response from embedded flash memory is PEG = 1, indicating that the operation was successful, and the contents of the block are properly protected from the program or erase operation. PEG = 1 is also true if an abort occurs during an HV request to a locked block. 2. If a program operation is attempted to a location marked as OTP, the response from the embedded flash memory is PEG = 0, indicating that the operation was not allowed, and the value interlocked was not programmed, since the desired doubleword was already programmed with a previous program operation. 0 Program or erase operation failed. 1 Program or erase operation successful.
23 PECIE	Program/Erase Complete Interrupt Enable PECIE provides a mechanism to trigger an interrupt request upon the assertion of the DONE flag due to a high voltage event (program or erase) finishing (normal, abort, suspend). If PECIE is written while not in a high voltage event, the interrupt will not immediately trigger, but will trigger after the next high voltage event is completed. 0 An interrupt request will not be generated when the DONE flag is set. 1 An interrupt request will be generated when the DONE flag is set.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 PGM	Program PGM is used to set up embedded flash memory for a program operation. A 0 to 1 transition of PGM initiates a program sequence. A 1 to 0 transition of PGM ends the program sequence. PGM can be set only under one of the following conditions: • User mode read (ERS is low and UTE is low) • Erase suspend (ERS and ESUS are 1) with EHV low PGM can be cleared only when PSUS and EHV are low and DONE is high. PGM is cleared on reset. 0 Flash memory is not executing a program sequence. 1 Flash memory is executing a program sequence.
28 PSUS	Program Suspend PSUS is used to indicate the embedded flash memory is in program suspend or in the process of entering a suspend state. The embedded flash memory is in program suspend when PSUS = 1 and DONE = 1. PSUS can be set high only when PGM and EHV are high. A 0 to 1 transition of PSUS starts the sequence which sets DONE and places the embedded flash memory in program suspend. The embedded flash memory enters suspend within Tpsus of this transition. PSUS can be cleared only when DONE and EHV are high. A 1 to 0 transition of PSUS with EHV = 1 starts the sequence which clears DONE and returns the embedded flash memory to program. The embedded flash memory cannot exit program suspend and clear DONE while EHV is low. PSUS is cleared on reset. 0 Program sequence is not suspended. 1 Program sequence is suspended.

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
29 ERS	Erase ERS is used to set up embedded flash memory for an erase operation. A 0 to 1 transition of ERS initiates an erase sequence. A 1 to 0 transition of ERS ends the erase sequence. ERS can only be set in user mode read (PGM is low and UTE is low). ERS can be cleared only when ESUS and EHV are low and DONE is high. ERS is cleared on reset. 0 Flash memory is not executing an erase sequence. 1 Flash memory is executing an erase sequence.
30 ESUS	Erase Suspend ESUS is used to indicate that the embedded flash memory is in erase suspend or in the process of entering a suspend state. The embedded flash memory is in erase suspend when ESUS = 1 and DONE = 1. ESUS can be set high only when ERS and EHV are high and PGM is low. A 0 to 1 transition of ESUS starts the sequence which sets DONE and places the flash memory in erase suspend. The embedded flash memory enters suspend within Tesus of this transition. ESUS can be cleared only when DONE and EHV are high and PGM is low. A 1 to 0 transition of ESUS with EHV = 1 starts the sequence which clears DONE and returns the embedded flash memory to erase. The embedded flash memory cannot exit erase suspend and clear DONE while EHV is low. ESUS is cleared on reset. 0 Erase sequence is not suspended. 1 Erase sequence is suspended.
31 EHV	Enable High Voltage The EHV bit enables the embedded flash memory for a high voltage program/erase operation. EHV is cleared on reset. EHV must be set after an interlock write to start a program/erase sequence. EHV may be set, initiating a program/erase, after an interlock under one of the following conditions: • Erase (ERS = 1, ESUS = 0) • Program (ERS = 0, ESUS = 0, PGM = 1, PSUS = 0) • Erase-suspended program (ERS = 1, ESUS = 1, PGM = 1, PSUS = 0) If a program operation is to be initiated while an erase is suspended, EHV must be cleared while in erase suspend before setting PGM. In normal operation, a 1 to 0 transition of EHV with DONE high, PSUS low, and ESUS low terminates the current program/erase high-voltage operation. In an erase-suspended program operation, a 1 to 0 transition of EHV with DONE high, ESUS high, PGM high, ERS high, and PSUS low terminates the program that is the current high voltage operation. When an operation is aborted, there is a 1 to 0 transition of EHV with DONE low and the suspend bit for the current program/erase sequence low. An abort causes the value of PEG to be cleared, indicating a failed program/erase; address locations being operated on by the aborted operation contain indeterminate data after an abort. EHV may not be written to a 1 after an abort is requested (EHV being cleared) and before DONE transitions high. A suspended operation cannot be aborted. EHV may be written during suspend. EHV must be high for the embedded flash memory to exit suspend. EHV may not be written after a suspend bit is set high and before DONE transitions high. EHV may not be set low after the current suspend bit is set low and before DONE transitions low. **NOTE:** Aborting a high voltage operation leaves FC addresses in an indeterminate data state. This may be recovered by executing an erase on the affected blocks. 0 Flash memory is not enabled to perform a high voltage operation. 1 Flash memory is enabled to perform a high voltage operation.

30.8.2 Extended Module Configuration Register (C55FMC_MCRE)

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HT	0		n256K					n64Kh			n32Kh		n16Kh		
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	n64Km			n32Km		n16Km			n64Kl			n32Kl		n16Kl		
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_MCRE field descriptions

Field	Description
0 HT	High Temperature Enabled. HT provides configuration information of the module. If the flash is tuned for high temperature operation (165 °C) at reduced performance (slower read at 165 °C and less and slower program and erase at 150 °C or less), and limited features at this temperature (no margin reads), the HT status bit will be a 1. Default is 150 °C as maximum operating temperature, and the HT status bit will be a 0. HT is read only. 0 150 °C maximum operating temperature. **NOTE:** 165 °C operating temperature maximum is not supported on this device. 1 Reserved
1–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 n256K	Number of 256 KB blocks. The value of this field is dependent upon the size of the embedded flash memory. This field is read only. 00000 Zero 256 KB blocks. 00001 Two 256 KB blocks (512 KB total). 00010 Four 256 KB blocks (1 MB total). 00011 Six 256 KB blocks (1.5 MB total). 00100 Eight 256 KB blocks (2 MB total).

Table continues on the next page...

C55FMC_MCRE field descriptions (continued)

Field	Description
	00101 Ten 256 KB blocks (2.5 MB total). 00110 Twelve 256 KB blocks (3 MB total). 00111 Fourteen 256 KB blocks (3.5 MB total). 01000 Sixteen 256 KB blocks (4 MB total). 01001 Eighteen 256 KB blocks (4.5 MB total). 01010 Twenty 256 KB blocks (5 MB total). 01011 Twenty-two 256 KB blocks (5.5 MB total). 01100 Twenty-four 256 KB blocks (6 MB total). 01101 Twenty-six 256 KB blocks (6.5 MB total). 01110 Twenty-eight 256 KB blocks (7 MB total). 01111 Thirty 256 KB blocks (7.5 MB total). 10000 Thirty-two 256 KB blocks (8 MB total). 10001 Reserved 10010 Reserved 10011 Reserved 10100 Reserved 10101 Reserved 10110 Reserved 10111 Reserved 11000 Forty-eight 256 KB blocks (12 MB total). 11001 Reserved. 11010 Reserved. 11011 Reserved. 11100 Reserved. 11101 Reserved. 11110 Reserved. 11111 Reserved.
8–10 n64Kh	Number of 64 KB blocks in partitions 4 and 5. This field is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Twelve 64 KB blocks.. 110 Sixteen 64 KB blocks. 111 Reserved.
11–12 n32Kh	Number of 32 KB blocks in partitions 4 and 5. This field is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.
13–15 n16Kh	Number of 16 KB blocks in partitions 4 and 5. This field is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks.

Table continues on the next page...

C55FMC_MCRE field descriptions (continued)

Field	Description
	010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
16–18 n64Km	Number of 64 KB blocks in partitions 2 and 3. This field is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
19–20 n32Km	Number of 32 KB blocks in partitions 2 and 3. This field is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.
21–23 n16Km	Number of 16 KB blocks in partitions 2 and 3. This field is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
24–26 n64Kl	Number of 64 KB blocks in partitions 0 and 1. This field is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
27–28 n32Kl	Number of 32 KB blocks in partitions 0 and 1. This field is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.

Table continues on the next page...

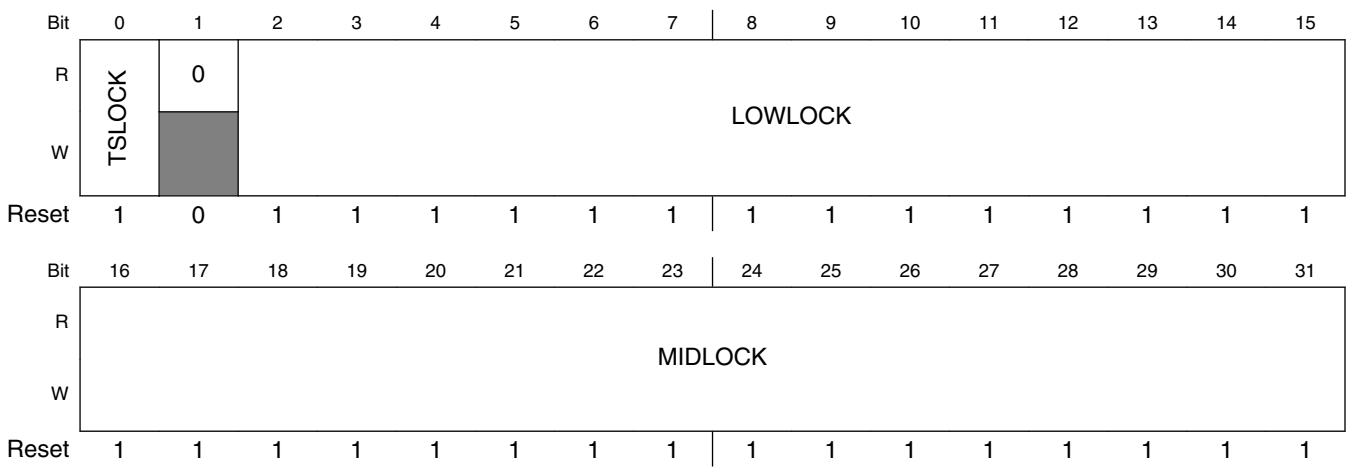
C55FMC_MCRE field descriptions (continued)

Field	Description
29–31 n16Kl	Number of 16 KB blocks in partitions 0 and 1. This field is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Reserved. 101 Reserved. 110 Reserved. 111 Reserved.

30.8.3 Lock 0 register (C55FMC_LOCK0)

The Lock 0 (LOCK0) register provides a means to protect blocks from being modified.

Address: 0h base + 10h offset = 10h



C55FMC_LOCK0 field descriptions

Field	Description
0 TSLOCK	UTest NVM Lock. This bit is used to lock the UTest NVM block from programs (erase protection not needed since UTest NVM is OTP and not erasable). A value of 1 in the TSLOCK register signifies that the UTest NVM block is locked for program. A value of 0 in the TSLOCK register signifies that the UTest NVM block is available to receive program pulses. The TSLOCK register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, TSLOCK register is not writable if a high voltage operation is suspended.

Table continues on the next page...

C55FMC_LOCK0 field descriptions (continued)

Field	Description
	0 UTest NVM block is available to be programmed. 1 UTest NVM block is locked and cannot be programmed.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 LOWLOCK	Low Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the Low Blocks starts at LOWLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Low Blocks exist in partitions 0 and 1. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations
16–31 MIDLOCK	Mid Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the Mid Blocks starts at MIDLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Mid Blocks exist in partitions 2 and 3. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

30.8.4 Lock 1 register (C55FMC_LOCK1)

The Lock 1 (LOCK1) register provides a means to protect blocks from being modified.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HIGHLOCK															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

C55FMC_LOCK1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHLOCK	High Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the High Blocks starts at HIGHLOCK[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. HIGH Blocks exist in partitions 4 and 5. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

30.8.5 Lock 2 register (C55FMC_LOCK2)

The Lock 2 (LOCK2) register provides a means to protect blocks from being modified.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A256KLOCK																															
W																																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

C55FMC_LOCK2 field descriptions

Field	Description
0–31 A256KLOCK	256 KB Block Lock 256KLOCK has the same characteristics as LOWLOCK. Please see that description for more information. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

30.8.6 Lock 3 register (C55FMC_LOCK3)

The Lock 3 (LOCK3) register provides a means to protect blocks from being modified.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																A256KLOCK															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

C55FMC_LOCK3 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 A256KLOCK	256 KB Block Lock 256KLOCK has the same characteristics as LOWLOCK. Please see that description for more information. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted.

Table continues on the next page...

C55FMC_LOCK3 field descriptions (continued)

Field	Description
	The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

30.8.7 Select 0 register (C55FMC_SEL0)

The Select 0 (SEL0) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		LOWSEL														MIDSEL															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_SEL0 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 LOWSEL	LOW Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the Low Blocks starts at LOWSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Low Blocks exist in partitions 0 and 1. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. LOWSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected 1 The block is selected for erase

Table continues on the next page...

C55FMC_SEL0 field descriptions (continued)

Field	Description
16–31 MIDSEL	Mid Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the Mid Blocks starts at MIDLSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Mid Blocks exist in partitions 2 and 3. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. MIDSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected 1 The block is selected for erase

30.8.8 Select 1 register (C55FMC_SEL1)

The Select 1 (SEL1) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HIGHSEL															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_SEL1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHSEL	High Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the High Blocks starts at HIGHSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. HIGH Blocks exist in partitions 4 and 5.

Table continues on the next page...

C55FMC_SEL1 field descriptions (continued)

Field	Description
	The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. HIGHSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected for erase 1 The block is selected for erase

30.8.9 Select 2 register (C55FMC_SEL2)

The Select 2 (SEL2) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	A256KSEL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

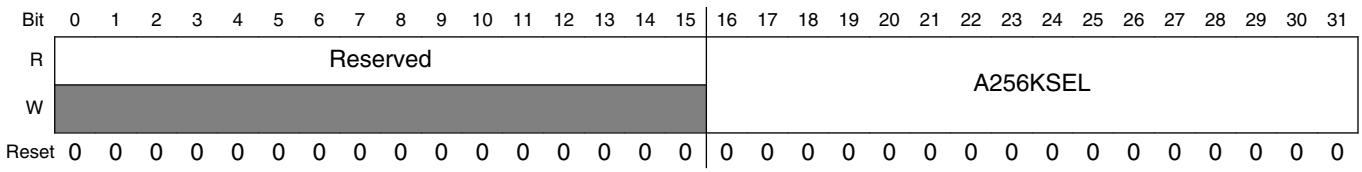
C55FMC_SEL2 field descriptions

Field	Description
0–31 A256KSEL	256 KB Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the 256 KB blocks starts with 256KSEL[0] and continues until all blocks are accounted. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. 256KSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected 1 The block is selected for erase

30.8.10 Select 3 register (C55FMC_SEL3)

The Select 3 (SEL3) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 44h offset = 44h



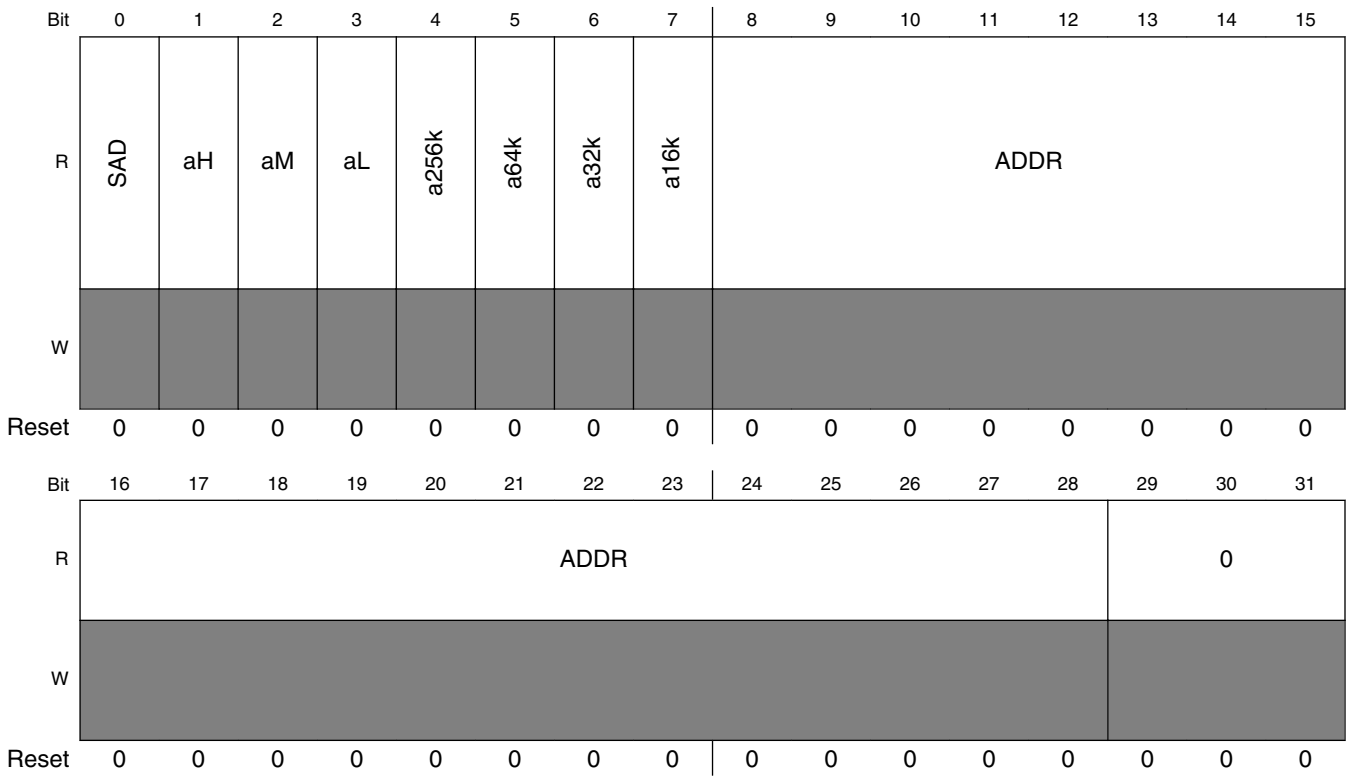
C55FMC_SEL3 field descriptions

Field	Description
0–15 Reserved	This field is reserved. Reserved, reset to 0.
16–31 A256KSEL	256 KB Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the 256 KB blocks starts with 256KSEL[0] and continues until all blocks are accounted. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. 256KSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 1 The block is selected for erase. 0 The block is not selected.

30.8.11 Address register (C55FMC_ADR)

The Address register (ADR) provides the first failing address in the event of a failure (ECC or PGM/Erase state machine), as well as single bit correction information.

Address: 0h base + 50h offset = 50h



C55FMC_ADR field descriptions

Field	Description
0 SAD	UTest NVM Address. The SAD bit qualifies the address captured during an ECC Event Error, Single Bit Correction, or State Machine operation. SAD has the same characteristics as ADDR related to capturing of addresses. The SAD field is not writable. 0 Address captured is from main array Space. 1 Address captured is from UTest NVM array space.
1 aH	Address High region. The aH bit qualifies the address field (ADDR) to the region of a16K, a32K and a64k. If aH is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from high address region. 1 Address captured or to be accessed is from high address region.
2 aM	Address Mid region. The aM bit qualifies the address field (ADDR) to the region of a16K, a32K and a64k. If aM is set, then the ADDR field maps to that region. See ADDR description for more information.

Table continues on the next page...

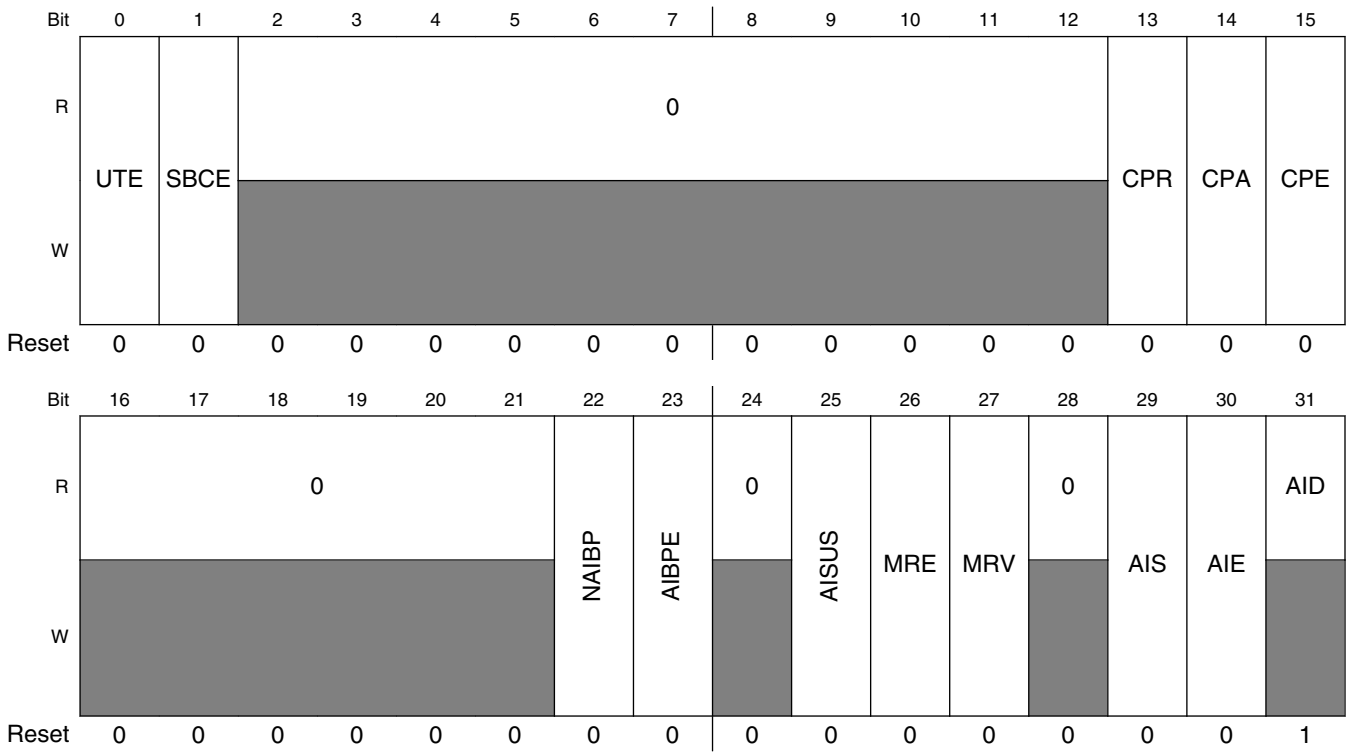
C55FMC_ADR field descriptions (continued)

Field	Description
	0 Address captured or to be accessed is not from mid address region. 1 Address captured or to be accessed is from mid address region.
3 aL	Address Low region. The aL bit qualifies the address field (ADDR) to the region of a16K, a32K and a64k. If aL is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from low address region. 1 Address captured or to be accessed is from low address region.
4 a256k	Address 256 KB region. The a256k bit qualifies the address field (ADDR) to the region. If a256k is set, then the ADDR field maps to that region. See ADDR description for more information. The a256k register is writable. 0 Address captured or to be accessed is not from 256 KB region. 1 Address captured or to be accessed is from 256 KB region.
5 a64k	Address 64 KB region. The a64k bit qualifies the address field (ADDR) to the region. If a64k is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from 64 KB region. 1 Address captured or to be accessed is from 64 KB region.
6 a32k	Address 32 KB region. The a32k bit qualifies the address field (ADDR) to the region. If a32k is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from 32 KB region. 1 Address captured or to be accessed is from 32 KB region.
7 a16k	Address 16 KB region. The a16k bit qualifies the address field (ADDR) to the region. If a16k is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from 16 KB region. 1 Address captured or to be accessed is from 16 KB region.
8–28 ADDR	Address. The ADR provides the first failing address in the event of ECC event error (EER set), Single Bit Correction (SBC set), as well as providing the address of a failure that may have occurred in a state machine operation (PEG cleared). ECC event errors take priority over Single Bit Corrections, which take priority over state machine errors. This is especially valuable in the event of an RWW operation, where the read senses an ECC error or Single Bit correction, and the state machine fails simultaneously. The failing address for ECC event error is held until EER is cleared. The failing address for Single Bit Correction is held until an ECC event error, or until SBC is cleared. State machine address is held until an ECC event error or Single Bit Correction event occurs, or until the next state machine event (PEG cleared) occurs. This address is always a doubleword address that selects 64 bits. ADDR[23:3] is an offset from a base address of 0x0 for each block size region. The aH, aM, aL, a16k, a32k, a64k, and a256k qualify the block size region the ADDR field.
29–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

30.8.12 UTest 0 register (C55FMC_UT0)

The User Test 0 (UT0) register provides a means to control UTest. UTest mode gives the ability to perform test features on the flash memory. This register is only writable when the flash memory is put into UTest mode by writing a passcode.

Address: 0h base + 54h offset = 54h



C55FMC_UT0 field descriptions

Field	Description
0 UTE	U-Test Enable. This status bit gives indication when U-Test is enabled. All bits in UT0, UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8 and UM9 are locked when this bit is 0. This bit is not writeable to a 1, but may be cleared. The reset value is 0. The method to set this bit is to provide a password, and if the password matches, the UTE bit is set to reflect the status of enabled, and is enabled until it is cleared by a register write. The UTE password will only be accepted if PGM = 0 and ERS = 0 (program and erase are not being requested). UTE can only be cleared if AID = 1, MRE and AIE = 0. While clearing UTE, writes to set AIE or MRE will be ignored. For UTE, the password 0xF9F9_9999 must be written to the UT0 register. 0 All bits in the UT0, UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8 and UM9 registers are locked 1 Bits in the UT0, UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8 and UM9 registers are unlocked
1 SBCE	Single Bit Correction Enable. SBCE enables Single Bit Correction results to be observed in SBC. ECC corrections that occur when SBCE is cleared will not be logged.

Table continues on the next page...

C55FMC_UT0 field descriptions (continued)

Field	Description
	0 Single Bit Corrections observation is disabled. 1 Single Bit Correction observation is enabled.
2–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 CPR	Customer Programmable Read Voltage and Reference Detection. This bit is used to control the error generation for the Customer Programmable Detection area in the UTest block. If this bit is set, and the Customer Programmable Detection location in UTest is read, a Read Voltage and Reference Error detection will be noted. This bit may not be set while CPE and CPA are set. If CPE or CPA are high, this bit will be locked. If attempts are made to write CPR, CPA or CPE simultaneously, none of them will be written. 0 Customer Programmable Read Voltage and Reference Detection Error is not enabled 1 Customer Programmable Read Voltage and Reference Detection Error is enabled
14 CPA	Customer Programmable Address Encode Detection. This bit is used to control the error generation for the Customer Programmable Detection area in the UTest block. If this bit is set, and the Customer Programmable Detection location in UTest is read, a Address Encode Error detection will be noted. This bit may not be set while CPE and CPR are set. If CPE or CPR are high, this bit will be locked. If attempts are made to write CPR, CPA or CPE simultaneously, none of them will be written. 0 Customer Programmable Address Encode Detection Error is not enabled 1 Customer Programmable Address Encode Detection Error is enabled
15 CPE	Customer Programmable EDC after ECC Detection. This bit is used to control the error generation for the Customer Programmable Detection area in the UTest block. If this bit is set, and the Customer Programmable Detection location in UTest is read, a EDC after ECC Error detection will be noted. This bit may not be set while CPR and CPA are set. If CPR or CPA are high, this bit will be locked. If attempts are made o write CPR, CPA or CPE simultaneously, none of them will be written. 0 Customer Programmable EDC after ECC Detection Error is not enabled 1 Customer Programmable EDC after ECC Detection Error is enabled
16–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 NAIBP	Next Array Integrity Break Point. If AIBPE is set, NAIBP will be set once a single bit correction (if enabled) or double bit detection is noted during the Array Integrity test. NAIBP is not writable to 1, but may be cleared to 0. Clearing NAIBP to 0, will enable the Array Integrity operation to be re-started after a breakpoint is encountered. If the Array Integrity operation completes without encountering another correction or detection, AID will be set with NAIBP remaining 0. 1 Array Integrity state machine is at a break point. 0 Array Integrity state machine is not currently at a break point.
23 AIBPE	Array Integrity Break Point Enable. If you want to enable breakpoints during an array integrity test, AIBPE may be set. See NAIBP description for more information about array integrity breakpoints. 1 Array Integrity breakpoints are enabled during Array Integrity Checks. 0 Array Integrity breakpoints are not enabled.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 AISUS	Array Integrity Suspend. AISUS enables a suspend of an Array Integrity Operation. Array Integrity may be suspend by Setting AISUS, and resumed by clearing AISUS. AISUS is writeable only when AID is low. AISUS is clearable only when AID is high. Attempting to write AISUS and AIE on the same clock cycle will result in only AIE getting written.

Table continues on the next page...

C55FMC_UT0 field descriptions (continued)

Field	Description
	1 Array integrity sequence is suspended. 0 Array integrity sequence not suspended.
26 MRE	Margin Read Enable. MRE combined with MRV enables user margin reads to be done. Normal user reads are not affected by MRE, although user reads while the margin read operation is ongoing are not supported. MRE is not writable if AID is low, or if AISUS is high, or if NAIBP is high. 1 Margin reads are enabled. 0 Margin reads are not enabled.
27 MRV	Margin Read Value. MRV selects the margin level that is being checked. Margin can be checked to an erased level (MRV=1) or to a programmed level (MRV=0). In order for this value to be valid, MRE must also be set. MRV is not writable if AID is low, or if AISUS is high, or if NAIBP is high. 1 One's margin reads are requested. 0 Zero's margin reads are requested.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 AIS	Array Integrity Sequence. AIS determines the address sequence to be used during array integrity checks. The default sequence (AIS = 0) is meant to replicate sequences that normal user code follows, and thoroughly checks the read propagation paths. This sequence is proprietary. The alternative sequence (AIS = 1) is logically sequential. It should be noted that the time to run a sequential sequence is shorter than the time to run the proprietary sequence. AIS is not writeable if AIE is high. 1 Array integrity sequence is sequential. 0 Array integrity sequence is proprietary sequence.
30 AIE	Array Integrity Enable. AIE set to one starts the array integrity check done on all selected blocks. The address sequence is determined by AIS, and the MISR (UM0 through UM9) can be checked after the operation is complete, to determine if a correct signature has been obtained. Once an array integrity operation is requested (AIE=1), it may be terminated by clearing AIE if the operation has finished (AID = 1) or aborted by clearing AIE if the operation is ongoing (AID = 0). AIE is not simultaneously writable to a 1 as UTE is being cleared to a 0. 0 Array integrity checks are not enabled. 1 Array integrity checks are enabled.
31 AID	Array Integrity Done. AID is cleared upon an array integrity check being enabled (to signify the operation is ongoing). Once completed, AID is set to indicate that the array integrity check is complete. At this time the MISR (UMR registers) can be checked. AID may also assert if breakpoints are enabled (AIBPE is set), an abort is requested or a suspend is requested. AID cannot be written, and is status only. 0 Array integrity check is ongoing. 1 Array integrity check is done.

30.8.13 UMISR register (C55FMC_UM*n*)

The Multiple Input Signature registers provide a means to evaluate array integrity.

Address: 0h base + 58h offset + (4d × i), where i=0d to 8d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

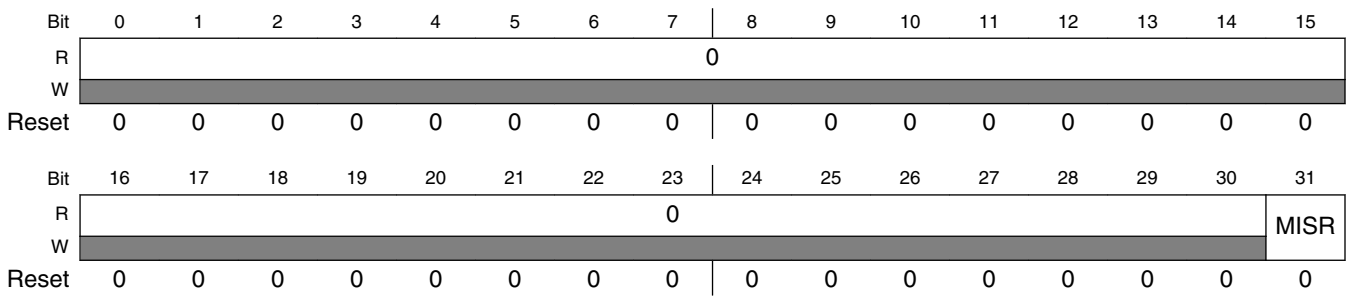
C55FMC_UM*n* field descriptions

Field	Description
0–31 MISR	MISR[31:0]. The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required). The MISR can be seeded to any value by writing the MISR registers. **NOTE:** Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way. The MISR register provides a means to calculate a MISR during array integrity operations. The MISR can be represented by the following polynomial: $x^{289} + x^{7} + x^{6} + x^{5} + x^{4} + x^{2} + 1$ The MISR is calculated by taking the previous MISR value and then "exclusive ORing" the new data. In addition the most significant bit (in this case it is MISR[288]), is then "exclusive ORed" into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2] and MISR[0]. The result of the "exclusive OR" is shifted left on each read. The MISR register is used in array integrity operations. If during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size) then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not recalculated, and the previous value is retained.

30.8.14 UMISR register (C55FMC_UM9)

The Multiple Input Signature registers provide a means to evaluate array integrity.

Address: 0h base + 7Ch offset = 7Ch



C55FMC_UM9 field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 MISR	MISR[288]. The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required). The MISR can be seeded to any value by writing the MISR registers. **NOTE:** Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way. The MISR register provides a means to calculate a MISR during array integrity operations. The MISR can be represented by the following polynomial: $x^{289} + x^{7} + x^{6} + x^{5} + x^{4} + x^{2} + 1$ The MISR is calculated by taking the previous MISR value and then "exclusive ORing" the new data. In addition the most significant bit (in this case it is MISR[288]), is then "exclusive ORed" into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2] and MISR[0]. The result of the "exclusive OR" is shifted left on each read. The MISR register is used in array integrity operations. If during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size) then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not recalculated, and the previous value is retained

30.8.15 Over-Program Protection 0 register (C55FMC_OPP0)

The Over-Program Protection 0 (OPP0) register provides a means to protect blocks from being over-programmed. This register shows the over-program protection status.

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 80h offset = 80h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		LOWOPP														MIDOPP															
W																																
Reset	0	0	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- MIDOPP field: The reset value of the MIDOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.
- LOWOPP field: The reset value of the LOWOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP0 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 LOWOPP	Low Block Over-Program Protection[13:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the Low Blocks starts at LOWOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. The LOWOPP register is not writable, and is status only. The default value of the LOWOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.
16–31 MIDOPP	Mid Block Over-Program Protection[15:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the Mid Blocks starts at MIDOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted.

Table continues on the next page...

C55FMC_OPP0 field descriptions (continued)

Field	Description
	The MIDOPP register is not writable, and is status only. The default value of the MIDOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.

30.8.16 Over-Program Protection 1 register (C55FMC_OPP1)

The Over-Program Protection 1 (OPP1) register provides a means to protect blocks from being over programmed. This register shows the over-program protection status.

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 84h offset = 84h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HIGHOPP															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- HIGHOPP field: The reset value of the HIGHOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHOPP	High Block Over-Program Protection[15:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the High Blocks starts at HIGHOPP[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. The HIGHOPP register is not writable, and is status only. The default value of the HIGHOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.

30.8.17 Over-Program Protection 2 register (C55FMC_OPP2)

The Over-Program Protection 2 (OPP2) register provides a means to protect blocks from being over programmed. This register shows the over-program protection status.

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 88h offset = 88h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A256KOPP																															
W																																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- A256KOPP field: The reset value of the A256KOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP2 field descriptions

Field	Description
0–31 A256KOPP	256K Block Over-Program Protection[31:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the 256 KB Blocks starts at A256KOPP[0] and continues until all blocks are accounted. The 256KOPP register is not writable, and is status only. The default value of the A256KOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 0 The corresponding block is available to be over-programmed.

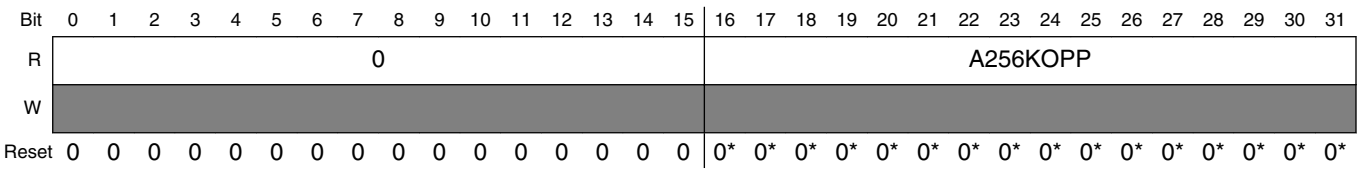
30.8.18 Over-Program Protection 3 register (C55FMC_OPP3)

The Over-Program Protection 3 (OPP3) register provides a means to protect blocks from being over programmed. This register shows the over-program protection status.

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 8Ch offset = 8Ch



* Notes:

- A256KOPP field: The reset value of the A256KOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP3 field descriptions

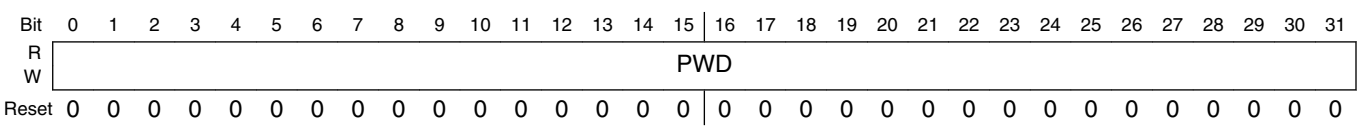
Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 A256KOPP	256K Block Over-Program Protection[47:32]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the 256 KB Blocks starts at 256KOPP[0] and continues until all blocks are accounted. The 256KOPP field is not writable, and is status only. The default value of the A256KOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.

30.8.19 Test Mode Disable Password Check register (C55FMC_TMD)

The Test Mode Disable Password Check (TMD) register provides a means to provide a challenge password to disable the Test Mode Disable Seal block select.

Writes to the register have no effect except for a password challenge. Reads to this register always return 0.

Address: 0h base + 90h offset = 90h



C55FMC_TMD field descriptions

Field	Description
0–31 PWD	Password challenge

30.9 Functional Description

The embedded flash memory consists of address spaces in four groupings:

- Low address space
- Mid address space
- High address space
- 256 KB address space

The main address space is divided into partitions. Each address space mentioned above consists of a partition pair. Partitions are used to determine locations for valid read-while-write (RWW) operations. While the embedded flash memory is performing a write (program or erase) to a given partition, it can simultaneously perform a read from any other partition. For program operations, only the address specified by an interlock write determines the partition being written (block locking and block select registers do not determine the RWW partitions being written). For erase operations, only blocks that are selected and unlocked determine the RWW partitions being written.

The main address space is also divided into blocks to implement independent erase and program protection. The UTest NVM block also exists as a block, and has independent program protection. The UTest NVM block is included to support systems that require nonvolatile memory (NVM) for security or to store system initialization information.

A number of MCR bits are protected against write when another bit, or set of bits, is in a specific state. These write locks are covered on a bit-by-bit basis in Module Configuration Register (C55FMC_MCR). The write locks do not consider the effects of trying to write two or more bits simultaneously. The embedded flash memory does not allow bits to be written simultaneously which put the device into an illegal state. This is implemented through a priority mechanism among the bits. The bit changing priorities are detailed in the following table.

Table 30-32. MCR bit set/clear priority levels

Priority level	MCR Bit(s)
1	ERS
2	PGM
3	EHV
4	ESUS, PSUS

If two or more MCR bits are written simultaneously only the bit with the lowest priority number is accepted for modification. Setting two bits with the same priority number is prevented by existing write locks or do not put the flash memory in an illegal state.

For example, setting ERS and PGM simultaneously results in only ERS being set. Attempting to clear EHV while setting PSUS results in EHV being cleared, while PSUS is unaffected.

Each read of the embedded flash memory retrieves a page, or eight consecutive words (256 bits), of information. The address for each word retrieved within a page differs from the other addresses in the page only by address bits [4:2]. The flash memory page read architecture easily supports both cache and burst mode at the BIU level for high-speed read applications.

The embedded flash memory supports fault tolerance through error correction code (ECC) and error detection. ECC implemented within the embedded flash memory corrects single-bit failures and detects double-bit failures.

A software mechanism is provided to independently lock and unlock each block.

Program and erase of the embedded flash memory requires multiple system clock cycles to complete. The program and erase sequence may be suspended or aborted.

The embedded flash memory may reside in various modes. The modes that are available include:

- User mode
- UTest mode

Each of these modes is discussed in detail in the following sections.

30.9.1 User Mode

In user mode the embedded flash memory may be read and written (register writes and interlock writes), programmed or erased. The following sub-sections define all actions that may be performed in user mode.

30.9.1.1 Read and write

The default state of the embedded flash memory is read. The main and UTest NVM address space can be read only in the read state. The MCR is always available for read. The embedded flash memory enters the read state on reset. The embedded flash memory is in the read state under four sets of conditions:

- The read state is active when the embedded flash memory is enabled (User Mode Read).
- The read state is active when MCR[PGM] or MCR[ERS] are high and high-voltage operation is ongoing (read-while-write).

NOTE

Reads done to the partition(s) being operated on (either erased or programmed) result in an error and the MCR[RWE] bit is set.

- The read state is active when MCR[PGM] and MCR[PSUS] are high (program suspend).
- The read state is active when MCR[ERS] and MCR[ESUS] are high and MCR[PGM] is low (erase suspend).

In embedded flash memory, flash core reads return 256 bits (1 page). Register reads return 32 bits of data.

NOTE

Flash core reads are done through the BIU. In many cases the BIU does read page buffering to allow sequential reads to be done with higher performance. This could provide a data coherency issue that must be handled with software. Data coherency may be an issue after a program or erase operation, as well as UTest NVM block operations.

In user mode, registers may be written. Array may be written to do interlock writes.

Register reads to unmapped register address space return all zeroes.

Register writes to unmapped register address space have no effect.

Interlock writes that are attempted during a high-voltage operation (MCR[EHV] = 1 or MCR[DONE] =0) will result in the interlock write being ignored, and address and data will not be updated.

NOTE

Care must be taken when doing 32-bit interlock writes. Performing two 32-bit interlock writes to the same word location is not supported. In addition, mixing 32-bit interlock writes with 64-bit interlock writes to the same double-word location is not supported. If these combinations are done, and a program operation is executed, the result will be MCR[PEG] = 0. See the Module Configuration Register (MCR) section for more information.

30.9.1.2 Program

A flash memory program sequence operates on any page within the flash core. Within a page, up to eight words may be altered in a single program operation. Also, up to four pages can be altered in a single program operation. Whenever the array is programmed, the ECC bits also get programmed. ECC is handled on a 64-bit boundary. Thus, if only one word in any given 64-bit ECC segment is programmed, the adjoining word (in that segment) should not be programmed, since ECC calculation has already completed for that 64-bit segment. Attempts to program the adjoining word will probably result in an operation failure. It is recommended that all programming operations be from 64 bits to 1024 bits, and be 64-bit aligned. The programming operation should completely fill selected ECC segments within the page. Only one program is allowed per 64-bit ECC segment between erases.

Warning

In rare cases over-programming of a 64-bit ECC segment may be done (EEPROM emulation).

Programming changes the value stored in an array bit from logic 1 to logic 0 only. Programming cannot change a stored logic 0 to a logic 1.

NOTE

If a logic 0 is attempted to be over-programmed by a logic 1, the resulting operation will fail (MCR[PEG] = 0), and the 0's that are interlocked will be merged (ORed) with 0's that are already present in the 64-bit ECC segment, unless the block is designated as an Over-Program Protected block.

Addresses in locked/disabled blocks cannot be programmed. Values may be programmed in any or all of eight words, within a page, with a single program sequence. Page-bound words have addresses which differ only in address bits [4:2]. Up to four pages can be programmed at once on a quad-page boundary, which differ only in address bits [6:5]. The program operation consists of the following sequence of events:

1. Change the value in the MCR[PGM] bit from a 0 to a 1.

   NOTE

   Ensure the block that contains the address to be programmed is unlocked.

2. Write the first address to be programmed with the program data. The embedded flash memory latches address bits [22:7] and SoC-specific UTest NVM enable at this time. The embedded flash memory latches data written as well. This write is referred to as

a program data interlock write. An interlock write may be done as a 64-bit transaction or a 32-bit transaction. Refer to the Flash Memory Controller chapter for information on the size of writes that are allowed.

3. If more than one word or doubleword is to be programmed, write each additional address in the page with data to be programmed. This is referred to as a program data write. The embedded flash memory ignores address bits [22:7] and SoC-specific UTest NVM enable for program data writes. All unwritten data words default to 0xFFFF_FFFF.
4. Write a logic 1 to the MCR[EHV] bit to start the internal program sequence or skip to step 9 to terminate.
5. Wait until the MCR[DONE] bit goes high.

NOTE

Since MCR[DONE] clears with MCR[EHV] being set, it may not be possible for software to read MCR[DONE] as a 0 prior to this step, depending on the operation selected.

6. Confirm MCR[PEG] = 1.
7. Write a logic 0 to the MCR[EHV] bit.
8. If more addresses are to be programmed, return to step 2.
9. Write a logic 0 to the MCR[PGM] bit to terminate the program sequence.

The first write after a program is initiated determines the quad-page address to be programmed. Program may be initiated with the 0 to 1 transition of the MCR[PGM] bit or by clearing the MCR[EHV] bit at the end of a previous program. This first write is referred to as an interlock write. The interlock write determines if the UTest NVM or normal array space is to be programmed by sampling SoC-specific UTest NVM enable and causing MCR[PEAS] to be set/cleared.

In the case of an erase-suspended program, the values in MCR[PEAS] may be modified via the program interlock write, enabling erase-suspended programs to and from UTest NVM space.

An interlock write must be performed before setting MCR[EHV]. A program sequence may be terminated by clearing MCR[PGM] prior to setting MCR[EHV].

After the interlock write, additional writes affect the data to be programmed at the word location determined by address bits [4:2], as well as the page location within a 1024-bit segment (determined by address bits [6:5]). Unwritten locations default to a data value of 0xFFFF_FFFF. If multiple writes are done to the same location, the data for the last write is used in programming.

While DONE is low, EHV is high, and PSUS is low, then EHV may be cleared, resulting in a program abort. A program abort forces the embedded flash memory to step 8 of the program sequence. An aborted program results in PEG being set low, indicating a failed operation. The data space being operated on before the abort contains indeterminate data. A program sequence may not be aborted while in program suspend.

Warning

Aborting a program operation leaves the flash core addresses being programmed in an indeterminate data state. This may be recovered by executing an erase on the affected blocks.

30.9.1.2.1 Program software locking

A software mechanism is provided to independently lock/unlock blocks against program and erase.

Software locking is done through the LOCK0, LOCK1, LOCK2, and LOCK3 registers. These may be written through register writes, and may be read through register reads.

30.9.1.2.2 Program hardware locking

Hardware locking which only affects the program operation is also available for UTest, 16 KB, 32 KB, 64 KB, and 256 KB blocks.

30.9.1.2.3 Program suspend/resume

The program sequence may be suspended to allow read access to the flash core. It is not possible to erase during a program suspend, or to program during a program suspend. These operations are prevented by register locking described in the Module Configuration Register (MCR) section. Read-while-write operation may also be used to read the array during a program sequence, providing the read is to a different partition.

A program suspend can be initiated by changing the value of the MCR[PSUS] bit from a 0 to a 1. MCR[PSUS] can be set high at any time when MCR[PGM] and MCR[EHV] are high. A 0 to 1 transition of MCR[PSUS] causes the embedded flash memory to start the sequence to enter program suspend, which is a read state. MCR[DONE] must become a logic level 1 before the embedded flash memory is suspended. At this time flash core reads may be attempted. Once suspended, the flash core may only be read. During suspend, all reads to flash core locations targeted for program, and blocks targeted for erase, return indeterminate data.

The program sequence is resumed by writing a logic 0 to MCR[PSUS]. MCR[EHV] must be set to a 1 before clearing MCR[PSUS] to resume operation. When the operation resumes, the embedded flash memory continues the program sequence from one of a set of predefined points. This may extend the time required for the program operation.

NOTE

Repeated suspends at a high frequency may result in the operation timing out, and the embedded flash memory will respond by completing the operation with a fail code (MCR[PEG] = 0). Although suspend frequency in not limited, enabling at least 20uS between suspend resume and future suspend requests improve the opportunity for the flash to complete the operation.

30.9.1.2.4 Over-program protection enable

One-time programming can be enabled on a block basis. In an over-program protection enabled block, any doubleword which has already been programmed cannot be programmed again. The over-program protection enable does not affect erase operation. Attempts to over-program will result in MCR[PEG] being cleared. If any doubleword within the quad-page has an over-program protection violation, MCR[PEG] will be cleared, and no doublewords will be programmed.

One-time programming can be enabled for 16 KB, 32 KB, 64 KB, or 256 KB blocks.

Over-program protection is enabled by sideband signals, and if enabled, the program operation will be modified to check for programmed bits prior to doing a high-voltage program event.

30.9.1.2.5 Successful program address

To enable the SoC to create logic to authenticate operations with a program (such as in a diary operation), the flash module provides the last successful address programmed as a sideband signal. This value will be held until the next successful program. A successful program is defined as a program operation of data that is not all 1's to an unlocked block, and the operation results in MCR[PEG] returning a 1. If any doubleword within the pages being programmed is interlocked with all 1's, the sideband signal will not be updated. An address of all 1's, and each of the block size selects selected, is the default (and invalid) address after reset.

30.9.1.3 Erase

Erase changes the value stored in all bits of the selected block(s) to logic 1. An erase sequence operates on any combination of blocks in the main address space. The erase sequence is fully automated within the flash memory. Blocks to be erased must be selected prior to initiating the erase sequence. Locked/disabled blocks cannot be erased. If multiple blocks are selected for erase during an erase sequence, the blocks are erased sequentially starting with the lowest numbered block and terminating with the highest (low first, mid second, high last, 16 KB first, 32 KB second, 64 KB third, last blocks are 256 KB). The erase sequence consists of the following events:

1. Change the value in the MCR[ERS] bit from 0 to 1.
2. Select the block or blocks to be erased by writing ones to the appropriate registers in SEL0, SEL1, SEL2, or SEL3 registers.

NOTE

Lock and Select are independent. If a block is selected and locked, no erase occurs.

3. Write to any address in flash memory. This is referred to as an erase interlock write. An erase interlock write to UTest NVM space is not allowed if sealed (since erase is not allowed in the UTest NVM space when sealed).
4. Write a logic 1 to the MCR[EHV] bit to start an internal erase sequence or skip to step 9 to terminate.
5. Wait until the MCR[DONE] bit goes high.

NOTE

Since MCR[DONE] clears with MCR[EHV] being set, it may not be possible for software to read MCR[DONE] as a 0 prior to this step, depending on the operation selected.

6. Confirm MCR[PEG] = 1.
7. Write a logic 0 to the MCR[EHV] bit.
8. If more blocks are to be erased, return to step 2.
9. Write a logic 0 to the MCR[ERS] bit to terminate the erase.

After setting ERS, one write, referred to as an interlock write, must be performed before EHV can be set to a 1. Data words written during erase sequence interlock writes are ignored. The erase sequence may be terminated by clearing ERS before setting EHV.

An erase operation may be aborted by clearing EHV, assuming DONE is low, EHV is high, and ESUS is low. An erase abort forces the embedded flash memory to step eight of the erase sequence. An aborted erase results in PEG being set low, indicating a failed operation. The block(s) being operated on before the abort contain indeterminate data. An erase sequence may not be aborted while in erase suspend.

30.9.1.3.1 Erase software locking

Software Locking affect erase operation. For details on this see Program software locking.

30.9.1.3.2 Erase hardware locking

Hardware locking which only affects the erase operation is also available for UTest, 16 KB, 32 KB, 64 KB, and 256 KB blocks.

30.9.1.3.3 Erase suspend/resume

The erase sequence may be suspended to allow read access to the flash core. The erase sequence may also be suspended to program (Erase-Suspended Program) the flash core. A program started during erase suspend can in turn be suspended. Only one erase suspend and one program suspend are allowed at a time during an operation. It is not possible to erase during an erase suspend, or program during a program suspend. During suspend, all reads to flash core locations targeted for program and blocks targeted for erase return indeterminate data.

Read-while-write operation may also be used to read the array during an erase sequence, provided the read is to a partition not selected for erase.

An erase suspend can be initiated by changing the value of the MCR[ESUS] bit from a 0 to a 1. MCR[ESUS] can be set to a 1 at any time when MCR[ERS] and MCR[EHV] are high and MCR[PGM] is low. A 0 to 1 transition of MCR[ESUS] causes the embedded flash memory to start the sequence which places it in erase suspend. MCR[DONE] must become a logic level 1 before the embedded flash memory is suspended and further actions are attempted. Once suspended, the array may be read or a program sequence may be initiated (erase-suspended program). Before initiating a program sequence, MCR[EHV] must first be cleared. If a program sequence is initiated, the values of SoC-specific UTest NVM enable is recaptured. Once the erase-suspended program is completed, the value of PEAS is returned to its erase value. Flash core reads that occur while MCR[ESUS] = 1 from the block(s) being erased will return indeterminate data.

The erase sequence is resumed by writing a logic 0 to MCR[ESUS]. MCR[EHV] must be set to a 1 and MCR[PGM] must be cleared (in the event of an erase-suspended program) before MCR[ESUS] can be cleared to resume the operation. The embedded flash memory continues the erase sequence from one of a set of predefined points. This may extend the time required for the erase operation.

Warning

In an erase-suspended program, programming flash core locations in blocks which were being operated on will respond by completing the operation with a fail code (MCR[PEG] = 0). Programming voltages will not be applied to the memory array in this case.

Warning

Repeated suspends at a high frequency may result in the operation timing out, and the embedded flash memory will respond by completing the operation with a fail code (MCR[PEG] = 0). Although suspend frequency is not limited, enabling at least 5mS between suspend resume and future suspend requests improve the opportunity for the flash to complete the operation.

30.9.2 UTest mode

UTest mode is a mode into which customers can put the embedded flash memory so as to do specific tests that check the integrity of the embedded flash memory.

30.9.2.1 Array integrity self check

Array integrity is checked using a predefined address sequence (based on UT0[AIS]), and this operation is executed on selected blocks. The data to be read is customer-specific - user code may be programmed into the flash memory and the correct MISR signature is calculated based on that code. Any random or nonrandom code is valid. Once the operation is completed, the results of the reads can be checked by reading the MISR value, to determine if an incorrect read or ECC detection was noted. Array integrity MISR value is calculated after ECC detection and correction. Array integrity requires that the Read Wait States and Address Pipelined control registers in the BIU be set to match the system frequency being used. The array integrity check consists of the following sequence of events:

1. Enable UTest mode.
2. Select the block or blocks to be receive the array integrity check by writing 1's to the appropriate registers in SEL0, SEL1, SEL2, or SEL3 registers. Blocks selected for array integrity check do not need to be unlocked.

NOTE

Unselected blocks are still read as part of the array integrity sequence, to enable full transition coverage. The result read on unselected blocks will not be captured in the MISR. Selecting fewer blocks may not result in a faster array integrity execution time, depending on the combinations of blocks selected and the sequence.

NOTE

Blocks protected with the Test Mode Disable Seal are still read as part of the array integrity sequence. The resulting read on sealed blocks will not be captured in the MISR, but the Single Bit Correction, Double Bit Detection and breakpoints will still be honored.

NOTE

It is not possible to do array integrity operations on the UTest NVM block.

3. If desired, set the UT0[AIS] bit to 1 for sequential addressing only.

NOTE

For normal integrity checks of the flash memory, sequential addressing is recommended. If it is required to more fully check the read path (in a diagnostic mode), it is recommended that AIS be left at 0, to use the address sequence that checks the read path more fully, and examine read transitions. This sequence takes more time.

4. Seed the MISR registers (UM0 — UM9) with desired values.
5. If breakpoints are desired, set UT0[AIBPE] to 1, and ensure that MCR[EER] and MCR[SBC] are cleared. If it is desired to break on single-bit correction, ensure that UT0[SBCE] is set.
6. Set the UT0[AIE] bit.
   a. If desired, the array integrity operation may be aborted prior to UT0[AID] going high. This may be done by clearing the UT0[AIE] bit and then continuing to the next step. It should be noted that in the event of an aborted array integrity check the MISR registers will contain a signature for the portion of the operation that was completed prior to the abort, and will not be deterministic. Prior to doing another array integrity operation, the UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8, and UM9 registers may need to be initialized to the desired seed value by doing register writes.
   b. If desired, the array integrity operation may be suspended prior to UT0[AID] going high. UT0[AISUS] may be set to request an array integrity suspend. After

the UT0[AISUS] bit is set, the user should wait for UT0[AID] bit to go high, which indicates the flash has entered the suspend state, and normal reads to the flash may be done. Once [AID] goes high, UT0[AISUS] may be cleared to resume the array integrity sequence.

NOTE

User mode array reads requested during the array integrity test will be ignored, to ensure that the array integrity operation is not corrupted. The memory array will not respond to array read requests during this time. User mode array reads may be executed if suspended or at a breakpoint.

7. Wait until the UT0[AID] bit goes high.
8. If breakpoints were enabled, check UT0[NAIBP], and if this is set to 1, the MCR[EER], MCR[SBC], and ADDR registers may be checked to determine the cause of the break and the address of the break. Prior to resuming the operation, clear MCR[SBC] or MCR[EER] bits. Then the operation may be resumed by clearing UT0[NAIBP]. Continue to wait until the UT0[AID] bit goes high. If breakpoints were not enabled, or if UT0[NAIBP] is low when UT0[AID] goes high, then the operation is complete. Continue to the next step.
9. Read values in the MISR registers (UM0 — UM9) to ensure correct signature.

NOTE

Array integrity reads may be done to unselected (or non-present) locations. Reads done to these locations do not update the MISR, and do not update the MCR[EER] and MCR[SBC] error bits.

10. Write a logic 0 to the UT0[AIE] bit.

30.9.2.2 User margin read

User margin read may be done using the array integrity interface, and has all the associated features of the array integrity interface (MISR and Breakpoints). User margin reads are done at a read margin level checking for erased bits or programmed bits encroaching on the nominal read level. User margin read is a self-timed event, and is independent of system clocks or wait states selected. Margin ECC corrections and detections are noted during the user margin read test. Margin read MISR value is calculated after ECC detection and correction.

The data to be read is customer-specific, and user code may be programmed into the flash memory. Any random or non-random code is valid. Once the operation is completed, the margin read results can be checked by reading the MCR[EER] bit and the MCR[SBC] bits to determine if zero, one, or two bits are being detected by the margin read, as well as checking the MISR.

The use model for margin read is in the event of a user-detected single bit correction (through user reads). A margin read may be done to check for a possible second bit falling within the selected margin levels.

The procedure for doing margin reads is:

1. Enable UTest mode.
2. Select the block or blocks to receive margin read check by writing 1's to the appropriate registers in SEL0, SEL1, SEL2, or SEL3 registers. Blocks selected for margin read do not need to be unlocked.

NOTE

Unselected blocks are still read as part of the margin read sequence, to enable full transition coverage. The result read on unselected blocks will not be captured in the MISR. Selecting fewer blocks will not result in a faster margin read execution time.

NOTE

Blocks protected with the Test Mode Disable Seal are still read as part of the margin read sequence. The resulting read on sealed blocks will not be captured in the MISR, but the Single Bit Correction, Double Bit Detection and breakpoints will still be honored.

NOTE

It is not possible to do margin read operations on the UTest NVM block.

3. Set the UT0[AIS] bit to 1 for sequential addressing only.

NOTE

For margin read checks of the flash memory, sequential addressing is recommended. Setting AIS to 0 is possible for Margin Reads, but using the sequence takes more time, and is not recommended.

4. Seed the MISR registers (UM0 — UM9) with desired values.
5. Ensure that the MCR[EER] and MCR[SBC] bits are cleared.
6. If you want to detect single bits during margin read, set UT0[SBCE] to 1.

7. Set the UTO[MRE] bit.
8. Set the UT0[MRV] bit to the desired value depending on if it is desired to do one's margin or zero's margin.
9. Set the UT0[AIE] bit.

NOTE

User mode array reads requested during the margin read test will be ignored, to ensure that the margin read operation is not corrupted. The memory array will not respond to array read requests during this time. User mode array reads may be executed if suspended or at a breakpoint.

NOTE

During Margin Read operations, with AID low, it is not recommended to attempt write operations to the MCR[SBC] and MCR[EER] bits. It is recommended that these bits only be written during suspend, breakpoints, or at completion of the Margin Read operation.

10. Wait until the UT0[AID] bit goes high.
    a. If breakpoints were enabled or a suspend was requested during margin read, the operation may be at a breakpoint or a suspend state. See Array integrity self check for more information.
11. Read values in the MISR registers (UM0 — UM9) to ensure correct signature.

NOTE

Margin reads may be done to unselected (or non-present) locations. Reads done to these locations do not update the MISR, and do not update the MCR[EER] and MCR[SBC] error bits.

12. Write a logic 0 to the UT0[AIE] bit.

30.10 Initialization information

A reset is the highest priority operation for the embedded flash memory and terminates all other operations.

The embedded flash memory uses reset to initialize register and status bits to their default reset values. If the embedded flash memory is executing a program or erase operation (PGM or ERS = 1) and a reset is issued, the operation is aborted and the embedded flash

memory disables the high-voltage logic without damage to the high-voltage circuits. Reset aborts all operations and forces the embedded flash memory into user mode ready to receive accesses.

After reset is requested, MCR[DONE] goes low, and remains low during reset and reset recovery. At the end of reset recovery, MCR[DONE] transitions from a 0 to a 1.

After reset is completed, register reads may be done, although it should be noted that registers that require updating from UTest NVM information, or other inputs, may not read updated values until MCR[DONE] transitions high.

During reset recovery, register writes are not allowed until the MCR[DONE] bit transitions high to indicate reset recovery is completed.

Warning

Resetting during a program or erase operation leaves the flash core blocks being programmed or erased in an indeterminate data state. This may be recovered by executing an erase on the affected blocks.

Chapter 31
Decorated Storage Memory Controller (DSMC)

31.1 Decorated Storage Memory Controller (DSMC) chip-specific module information

The DSMC supports five store and three load operations, including:

- Bit field inserts
- Compare-and-store
- Bitwise AND, OR, and XOR operators
- Simple memory load
- Swap
- Load-and-set-1(bit)

As the DSMC generates the atomic read-modify-write bus transactions to the attached slave memory controller, the two transactions (read, write) are fully pipelined with no idle cycles introduced. During these transactions, the AHB hlock control signal is asserted to the slave during the entire read-modify-write.

The module includes error checking logic that validates the decoration field, making sure that it is properly defined (all illegal commands are rejected and the transfer error terminated). Additionally, the decorated storage memory controller only operates on aligned, single data transfers (misalignment and/or bursts are not supported and error terminated if attempted).

The DSMC is instantiated multiple times within the core platform and physically resides between the slave ports of the crossbar and the targeted memory controllers.

For the this device, the DSMC is instantiated five times, three times in the computational platform shell and two times in the peripheral platform shell. The access size is forced to 64-bit for system RAM, 32-bit for the core xMEMs and PBRIDGE.

Table 31-1. DSMC instantiations

DSMC and memory controller association	
Computational shell	System RAM
	Core 0 xMEM (Backdoor Port to the core local Instruction and Data Memories)
	Core 1 xMEM (Backdoor Port to the core local Instruction and Data Memories)
Peripheral shell	PBRIDGE_0 (Bridge to peripheral subsystem)
	PBRIDGE_1 (Bridge to peripheral subsystem)

31.2 Introduction

This section details the hardware support for atomic read-modify-write memory operations. In the Power Architecture, these capabilities are called "*decorated storage*". It is supported by capabilities in the processor cores plus instantiations of a Decorated Storage Memory Controller (DSMC).

The Decorated Storage APU defines instructions for providing load and store operations to memory addresses that require *additional semantics* beyond just the reading and writing of data values to the addressed memory locations. Decorated storage operations are intended to be used for specific devices or memory targets that require these additional semantics. A "decoration" is the additional semantic information to be applied to the decorated storage operation by the DSMC.

Consider the basic mnemonics, syntax and formats for the decorated load instructions.

For loads, the syntax is `l[b,h,w]d{cb}x rT,rB,rA` where the data size specifier is defined as 8-bit (`b` = byte), 16-bit (`h` = halfword) or 32-bit (`w` = word), and the three register specifiers include `rT` as the destination target register, `rB` as the effective address and `rA` as the decoration. The optional `{cb}` specifier defines a version of the instruction which is treated as a cache bypass operation.

The syntax for store instruction is `st[b,h,w]d{cb}x rS,rB,rA` where the same data size specifiers are used, and the three register specifiers are `rS` as the source data register, `rB` as the effective address and `rA` as the decoration. The cache bypass attribute is again specified with the use of the optional `{cb}` in the instruction mnemonic.

For all decorated loads and stores, the memory effective address is simply defined by the rB register (hence the "x" mnemonic suffix, signaling an "indexed" addressing mode) and the decoration is specified by the rA register. The core transmits the contents of the rA register as the decoration value along with the access address (register rB) to the DSMC.

The decorated load and store instruction formats are shown in the following figure.

Table 31-2. Power Architecture decorated load and store instruction formats

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
lbdx	0	1	1	1	1	1	rT						rA				rB				1	0	0		0	0	0	0	0	1	1	/
lhdx	0	1	1	1	1	1	rT						rA				rB				1	0	0		0	1	0	0	0	1	1	/
lwdx	0	1	1	1	1	1	rT						rA				rB				1	0	0		1	0	0	0	0	1	1	/
lbdcbx	0	1	1	1	1	1	rT						rA				rB				1	0	0		0	0	0	0	0	1	0	/
lhdcbx	0	1	1	1	1	1	rT						rA				rB				1	0	0		0	1	0	0	0	1	0	/
lwdcbx	0	1	1	1	1	1	rT						rA				rB				1	0	0		1	0	0	0	0	1	0	/
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
stbdx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1		0	0	0	0	0	1	1	/
sthdx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1		0	1	0	0	0	1	1	/
stwdx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1		1	0	0	0	0	1	1	/
stbdcbx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1		0	0	0	0	0	1	0	/
sthdcbx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1		0	1	0	0	0	1	0	/
stwdcbx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1		1	0	0	0	0	1	0	/

The basic decorated load or store memory reference supplements the access address and attribute information with a 32-bit "decoration". The DSMC decoration defines the special memory operation to be performed and optionally includes bit specifiers and/or operand data for the command.

The default format for the 32-bit decoration field defines a 4-bit command in the most significant bits (decoration[0:3]).

The cache bypass decoration load (`l[b,h,w]dcbx`) and store (`st[b,h,w]dcbx`) instructions do not distinguish between cacheable and cache-inhibited storage attributes, and instead have semantics to effectively emulate a cache-inhibited operation to storage regardless of the actual storage attributes. This is desired in order to allow a small portion of an otherwise cacheable storage area to be treated as non-cacheable using instruction supplied semantics, regardless of storage attributes provided by a memory management, protection or other memory control scheme. This allows decoration or other accesses to be applied to a portion of memory that is normally treated as cacheable according to its storage attributes, as well as to ensure that any stale copies of storage locations present in the cache are not used.

The combination of the decorated load and store instruction support in the e200zX cores plus the decorated storage memory controller in the core platform adds a robust atomic read-modify-write capability to the Power Architecture. The architecture capability defined by these core and platform functions is targeted at manipulation of *n*-bit fields in peripheral registers (and is consistent with I/O hardware addressing in the Embedded C standard) as well as software synchronization data structures needed in multi-core systems (mutexes, semaphores, test-and-set and compare-and-swap structures).

For additional information on the decorated load and store instructions from the core's perspective, refer to the appropriate e200zX core reference manual.

31.3 Decorated stores: st[b,h,w]d{cb}x rS,rA,rB

The next sections present descriptions of the specific operations, based on the 4-bit command field defined in decoration[0:3]. These descriptions include the bit pattern definitions for the 32-bit decoration value and include pseudo-code detailing the sequence of operations. The decoration formats are defined using a <command>.<size> syntax where the operand size specifier is b (byte, 8-bit), h (halfword, 16-bit) or w (word, 32-bit). The operand size is specified directly in the decorated instruction executed in the core, and this attribute is driven to the system bus as part of the decorated data transfer. Additionally, the write data is taken from the right-justified 8, 16 or 32 bits of the rS register, that is:

```
 8-bit wdata = rS[24:31]
16-bit wdata = rS[16:31]
32-bit wdata = rS[ 0:31]
```

Likewise, read data is loaded into the right-justified 8, 16 or 32 bits of the rT register:

```
 8-bit rdata = rT[24:31]
16-bit rdata = rT[16:31]
32-bit rdata = rT[ 0:31]
```

For the byte and halfword decorated load instructions, the upper bits of the rT register are zero filled.

The starting bit (SRTBIT) position follows the Power Architecture convention where the MSB is bit 0 and the LSB is bit 31. The bit field width (BFW) value defines the width and BFW = 0 specifies the maximum width, that is, the container size.

The basic decorated store includes three data transfer fields sourced from the core: the access address (rB), the decoration (rA) and the write data (wdata) operand (rS). There are five operations defined and most of these transactions convert a single core AHB write bus cycle into an atomic read-modify-write, that is, an indivisible read followed by

write sequence. Support for three basic boolean logic functions (AND, OR, XOR) is provided along with a compare-and-store operation plus a bit field insert operation. These operations do not support any type of bit field wrapping.

In the next sections detailing the decorated store operations, the following associations between the pseudocode variables and the core registers apply.

- decoration = rA
- accessAddress = rB
- wdata = rS

31.3.1 Bit Field Insert (BFINS) into an 8, 16 or 32-bit memory container

Table 31-3. Decoration format: BFINS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
bfins.b	0	0	0	0	0	0	0	SRTBIT			0	0	0	BFW			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bfins.h	0	0	0	0	0	0	SRTBIT				0	0	BFW				-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bfins.w	0	0	0	0	0	SRTBIT					0	BFW					-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command inserts a bit field defined by SRTBIT and BFW into the memory "container" defined by the access size associated with the decorated store instruction using an atomic read-modify-write sequence.

```
tmp = mem[accessAddress, size] // memory read
if bfw == 0
         then bfw = container

if ((srtbit + bfw) <= container) // generate bit mask
    mask = ((1 << bfw) - 1) << (container - (srtbit + bfw))
else
    mask = ((1 << bfw) - 1) >> ((srtbit + bfw) - container)

tmp = tmp  & ~mask// modify
    | wdata &  mask

mem[accessAddress, size] = tmp // memory write
```

The write data operand (wdata) associated with the decorated store instruction contains the bit field to be inserted. *It must be properly aligned within a right-justified container in the source register (rS)*, that is, within the lower 8 bits for a byte operation, the lower 16 bits for a halfword or the entire 32 bits for a word operation.

To illustrate, consider the following example of the insertion of the 3-bit field "xyz" into an 8-bit memory container, initially set to "abcd_efgh". For all cases, the bit field width (BFW) is 3.

```
if SRTBIT = 0 and the decorated store rS register[24:31] = xyz-_----,
   then destination is "xyzd_efgh"
if SRTBIT = 1 and the decorated store rS register[24:31] = -xyz_----,
   then destination is "axyz_efgh"
if SRTBIT = 2 and the decorated store rS register[24:31] = --xy_z---,
   then destination is "abxy_zfgh"
if SRTBIT = 3 and the decorated store rS register[24:31] = ---x_yz--,
   then destination is "abcx_yzgh"
if SRTBIT = 4 and the decorated store rS register[24:31] = ---_xyz-,
   then destination is "abcd_xyzh"
if SRTBIT = 5 and the decorated store rS register[24:31] = ---_-xyz,
   then destination is "abcd_exyz"
if SRTBIT = 6 and the decorated store rS register[24:31] = ---_--xy,
   then destination is "abcd_efxy"
if SRTBIT = 7 and the decorated store rS register[24:31] = ---_---x,
   then destination is "abcd_efgx"
```

If the bit field insert operation specifies SRTBIT as 0 and BFW as 0 (indicating the width matches the container width), then the operation is logically equivalent to a simple register store operation.

31.3.1.1 Compare-and-Store (CAST)

Table 31-4. Decoration format: CAST

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
cast.b	1	0	0	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	CD8							
cast.h	1	0	0	1	-	-	-	-	-	-	-	-	-	-	-	-	CD16															
cast.w	1	0	0	1	CD28																											

This command begins by performing a read of the referenced memory address and comparing that data with the operand included in the decoration word. If the read data is equal to the compare operand, the write data associated with the decorated store instruction is placed into the memory location, else the original read data is rewritten into the memory location. The write operand is selected from the appropriate data byte lanes in the same manner as any memory store operation; this implies the write data operand is right justified in the rS source register.

```
tmp = mem[accessAddress, size]  // memory read

if (size == 8)             // define compare_operand
    compare_operand = CD8
else if (size == 16)
          compare_operand = CD16
     else compare_operand = {0x0, CD28}// zero-filled data

if (tmp == compare_operand) // compare - "modify"
```

```
     mem[accessAddress, size] = wdata// memory write
else mem[accessAddress, size] = tmp
```

Note for the word size operation (cast.w), the data value specified in the low-order 28 bits of the decoration is zero filled in the most significant bits to create the required 32 bit compare operand.

31.3.1.1.1 Logical AND (AND)

Table 31-5. Decoration format: AND

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
and.{b,h,w}	1	0	1	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical AND operation using the write operand defined by the rS register; finally, the result of the AND operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp & wdata      // modify
mem[accessAddress, size] = tmp// memory write
```

31.3.1.1.2 Logical OR (OR)

Table 31-6. Decoration format: OR

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
or.{b,h,w}	1	1	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical OR operation using the write operand defined by the rS register; finally, the result of the OR operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp | wdata         // modify
mem[accessAddress, size] = tmp// memory write
```

31.3.1.1.3 Logical Exclusive-OR (XOR)

Table 31-7. Decoration format: XOR

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
xor.{b,h,w}	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical XOR operation using the write operand defined by the rS register; finally, the result of the XOR operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp ^ wdata          // modify
mem[accessAddress, size] = tmp// memory write
```

31.3.1.2 Decorated Loads: `l[b,h,w]d{cb}x rT,rA,rB`

The basic decorated load includes two data transfer fields sourced from the core: the access address (rB), the decoration (rA) plus the read data (rdata) operand returned from DSMC to the core and loaded into the destination register (rT). There are 3 operations defined and two of these transactions convert a single core AHB read bus cycle into an atomic read-modify-write. Support for a swap function, a load-and-set-1-bit functions are provided along with a simple load (aka an "extract") operation.

In the next sections detailing the decorated load operations, the following associations between the pseudocode variables and the core registers apply.

- decoration = rA
- accessAddress = rB
- rdata = rT

31.3.1.2.1 Simple Load (SLD)

Table 31-8. Decoration format: SLD

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
sld. {b,h,w}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command performs a simple memory load of the reference size from the access address. For this operation, the 12-bit field defined in decoration[4:15] must be zeroes else the transfer is not performed and error terminated.

```
rdata = mem[accessAddress, size] // memory read
```

Support for generic bit field extract operations is best handled using existing core and compiler capabilities involving standard load instructions followed by left and right shift operations to emulate these functions.

31.3.1.2.2 Registers-and-memory exchange (SWAP)

Table 31-9. Decoration format: SWAP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
swap.b	0	1	0	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	WD8							
swap.h	0	1	0	1	-	-	-	-	-	-	-	-	-	-	-	-	**WD16**															
swap.w	0	1	0	1	WD28																											

This command loads the contents of the referenced memory location into the core's rT register and stores the source operand from the core's rA register (the decoration) into the memory location.

```
tmp   = mem[accessAddress, size] // memory read
rdata = tmp// rT = rdata = tmp

if (size == 8)             // store_operand - "modify"
    store_operand = WD8
else if (size == 16)
           store_operand = WD16
    else store_operand = {0x0, WD28} // zero-filled write data

mem[accessAddress, size] = store_operand// memory write
```

Note for the word size operation (swap.w), the data value specified in the low-order 28 bits of the decoration is zero filled in the most significant bits to create the required 32 bit store operand.

31.3.1.2.3 Load-and-Set-1(Bit) (LAS1)

Table 31-10. Decoration format: LAS1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
las1.b	0	1	1	0	0	0	0	BIT			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
las1.h	0	1	1	0	0	0	BIT				-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
las1.w	0	1	1	0	0	BIT					-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command first reads the referenced memory location; it then modifies the read operand by setting the single bit position defined in the decoration (BIT); the modified data is then written back to the referenced memory location and the original read data returned to the initiating processor core.

```
tmp   = mem[accessAddress, size] // memory read
rdata = tmp // rT = rdata = tmp
mask  = 1 << (container - bit - 1) // generate bit mask
tmp   = tmp | mask// modify
mem[accessAddress, size] = tmp// memory write
```

31.4 DSMC instantiations

The decorated storage memory controller which performs these operations is instantiated multiple times within the core platform and physically resides between the slave ports of the crossbar and the targeted memory controllers.

The module includes error checking logic that validates the decoration field is properly defined (all illegal commands are rejected and the transfer error terminated). Additionally, the decorated memory controller only operates on aligned, single data transfers (misalignment and/or bursts are not supported and error terminated if attempted).

As the DSMC generates the atomic read-modify-write bus transactions to the attached slave memory controller, the two transactions (read, write) are fully pipelined with no idle cycles introduced. During these transactions, the AHB hlock control signal is asserted to the slave during the entire read-modify-write.

Chapter 32
Flash Memory Programming and Configuration

32.1 Introduction

This chapter provides basic guidance on activities related to configuring and programming the embedded flash memory in the MPC5746R microcontroller.

Multiple layers of protection are available to prevent unintended updates to flash contents and unauthorized reads and writes to flash contents. The protection mechanisms range from simple write-protect bits for individual flash blocks to four levels of 256-bit password read and/or write protection and configuration of flash memory blocks as One Time Programmable (OTP).

Additionally, the MPC5746R microcontroller includes a tamper detection feature and debug port enable/disable.

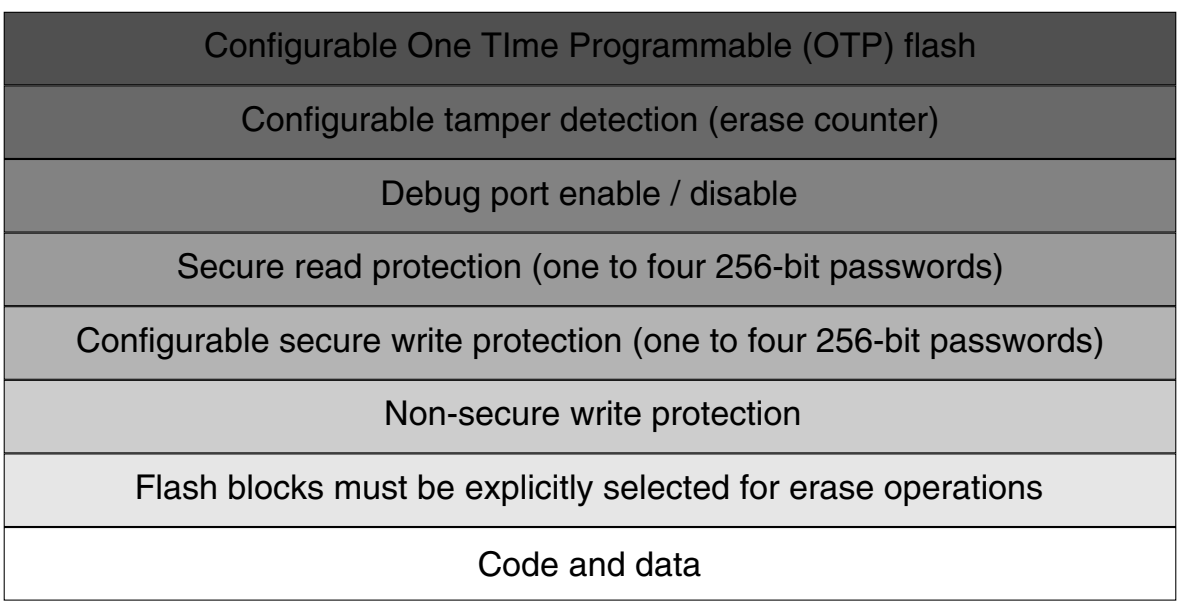


Figure 32-1. Flash memory protection layers

At the lowest layer of protection, all flash erase operations require the explicit selection of one or more blocks of flash memory before the operation is performed. When executed, the operation is only performed on the selected block(s).

The next layer is the chip's ability to lock individual flash blocks against unintended programming and erase operations. This level of protection does not offer any password protection and is intended to provide protection against *accidental* changes to code and data.

The first layer of *secure* protection is the microcontroller's ability to support up to four levels of 256-bit password-secured write protection for individual flash blocks.

Similarly, up to four levels of password-secured read protection can be implemented. Unlike the password-secured *write* protection, which is implemented on a per-block basis, password-secured read protection applies to three pre-defined (at the factory) groups of flash blocks. The read protection applied to a group applies to each block within the group. In contrast, secure write protection is defined at the individual block level.

Secure password protection can also be implemented for the debug port.

The MPC5746R microcontroller also includes a tamper detection feature. In its most basic implementation, tamper detection acts as an erase counter. If tamper detection is implemented, a DCF record in the tamper detection *diary* must be written by software before a protected flash block can be erased. Multiple tamper detection regions can be defined by the customer, with each region containing one or more blocks. A block can be included in more than one tamper detection regions if desired.

Another protection feature, related to tamper detection, is the ability to configure any flash block as one-time programmable (OTP).

The remainder of the chapter provides details on these protection mechanisms and gives a brief overview of security configuration planning.

NOTE

Most of the topics in this chapter build on information presented in previous topics. To get a clear overview of how the flash protection features work and relate to each other, please read the chapter in sequence, from beginning to end.

Access protection configuration is applied on a per-block basis (except read protection and tamper detection applied to groups of flash memory blocks), i.e., a custom level of access protection can be provided for each flash memory block. This is accomplished through the use of registers containing bits that are mapped to individual flash blocks. Understanding the mapping is key to understanding the implementation and use of the MPC5746R microcontroller's flash protection features.

Depending on the type of protection to be applied, different sets of registers (or even DCF records) are used, but the mapping scheme is consistent. This mapping scheme is first presented in Selection of flash memory blocks for erase. It is important to understand the topic before proceeding to other sections.

32.2 Selection of flash memory blocks for erase

The registers used for selecting flash blocks for erase operations are the flash control registers named C55FMC_SEL0 and C55FMC_SEL2.

NOTE

There are also a SEL1 and SEL3 registers, but no flash blocks are mapped to them.

The SEL0 and SEL2 registers are 32-bit registers containing fields associated with low, mid, and 256 KB address spaces. Each non-reserved bit in the fields is mapped to a specific flash block.

For example, the following figure shows the C55FMC_SEL0 register. This register provides a way of selecting blocks in the low and mid address spaces for erase operations. (See the Embedded Flash Memory chapter for details of the SEL0 register.)

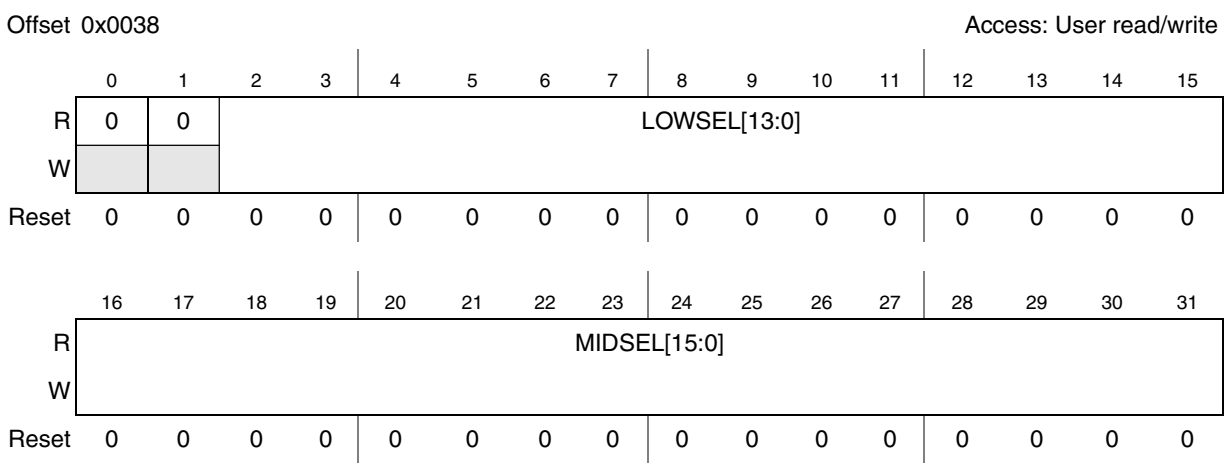


Figure 32-2. SEL0 flash control register

The mapping of the register bits to flash blocks is shown in C55FMC_SEL0 register bit mapping. Note that not all bits of the LOWSEL and MIDSEL fields are mapped; only LOWSEL bits 0-11 and MIDSEL bits 0-9 are mapped.

The following table shows the mapping of SEL0 register's LOWSEL and MIDSEL field bits to individual flash blocks with start and end addresses. Before any mapped block can be erased, it must be selected by writing a '1' to its corresponding bit in one of the SEL*n* registers.

NOTE

Values written to the SEL registers do not persist through device reset. As indicated by the reset values shown in the figure above, all bits are reset to '0'. As a result, after a device reset all blocks mapped to the register are, by default, not selected for erase.

Table 32-1. SEL0 to low and mid address space flash block bit mapping

Start Address	End Address	Size (KB)	Partition	SEL*n* Register	SEL*n* Register Field Bit	Block No.	Description
0x00404000	0x00407FFF	16	0	SEL0	LOWSEL[0]	0	16KB BAF block0[1]
0x00FA0000 0x08FA0000	0x00FA3FFF 0x08FA3FFF	16	0	SEL0	LOWSEL[1]	1	Code Flash block1
0x00FA4000 0x08FA4000	0x00FA7FFF 0x08FA7FFF	16	0	SEL0	LOWSEL[2]	2	Code Flash block2
0x00FA8000 0x08FA8000	0x00FABFFF 0x08FABFFF	16	1	SEL0	LOWSEL[3]	3	Code Flash block3
0x00FAC000 0x08FAC000	0x00FAFFFF 0x08FAFFFF	16	1	SEL0	LOWSEL[4]	4	Code Flash block4
0x00F9C000 0x08F9C000	0x00F9FFFF 0x08F9FFFF	16	1	SEL0	LOWSEL[5]	5	Code Flash block5
0x00FB0000 0x08FB0000	0x00FB7FFF 0x08FB7FFF	32	0	SEL0	LOWSEL[6]	0	Code Flash block0
0x00FB8000 0x08FB8000	0x00FBFFFF 0x08FBFFFF	32	1	SEL0	LOWSEL[7]	1	Code Flash block1
0x00FC0000 0x08FC0000	0x00FCFFFF 0x08FCFFFF	64	0	SEL0	LOWSEL[8]	0	Code Flash block0
0x00FD0000 0x08FD0000	0x00FDFFFF 0x08FDFFFF	64	0	SEL0	LOWSEL[9]	1	Code Flash block1
0x00FE0000 0x08FE0000	0x00FEFFFF 0x08FEFFFF	64	1	SEL0	LOWSEL[10]	2	Code Flash block2
0x00FF0000 0x08FF0000	0x00FFFFFF 0x08FFFFFF	64	1	SEL0	LOWSEL[11]	3	Code Flash block3
0x00800000	0x00803FFF	16	2	SEL0	MIDSEL[0]	0	FMC EEPROM Block0
0x00804000	0x00807FFF	16	2	SEL0	MIDSEL[1]	1	FMC EEPROM Block1
0x00808000	0x0080BFFF	16	2	SEL0	MIDSEL[2]	2	FMC EEPROM Block2
0x0080C000	0x0080FFFF	16	2	SEL0	MIDSEL[3]	3	FMC EEPROM Block3
0x00810000	0x00813FFF	16	3	SEL0	MIDSEL[4]	4	FMC EEPROM Block4
0x00814000	0x00817FFF	16	3	SEL0	MIDSEL[5]	5	FMC EEPROM Block5
0x00818000	0x0081BFFF	16	3	SEL0	MIDSEL[6]	6	FMC EEPROM Block6

Table continues on the next page...

Table 32-1. SEL0 to low and mid address space flash block bit mapping (continued)

Start Address	End Address	Size (KB)	Partition	SEL*n* Register	SEL*n* Register Field Bit	Block No.	Description
0x0081C000	0x0081FFFF	16	3	SEL0	MIDSEL[7]	7	FMC EEPROM Block7
0x00820000	0x0082FFFF	64	2	SEL0	MIDSEL[8]	0	FMC EEPROM Block0
0x00830000	0x0083FFFF	64	3	SEL0	MIDSEL[9]	1	FMC EEPROM Block1

1. The Boot Assist Flash (BAF) is one time programmable and the majority of it is programmed in the factory. The BAF handles the initial start up of the device.

Example 1. Selecting blocks for erase

Referring to the table above, writing the value 0x0030_0000 to the SEL0 register results in the SEL_LOWSEL[4] and SEL_LOWSEL[5] bits being set to '1', selecting the following blocks for erase:

Table 32-2. Selecting blocks for erase

Start Address	End Address	Size (KB)	Partition	SEL*n* Register	SEL*n* Register Field Bit	Block No.	Description
0x00FAC000 0x08FAC000	0x00FAFFFF 0x08FAFFFF	16	1	SEL0	LOWSEL[4]	4	Code Flash block4
0x00F9C000 0x08F9C000	0x00F9FFFF 0x08F9FFFF	16	1	SEL0	LOWSEL[5]	5	Code Flash block5

32.3 Non-secure write protection

Non-secure write protection refers to the chip's ability to lock individual flash blocks against programming and erase operations. This level of protection is intended to provide protection against *accidental* changes to code and data. Although similar in function, this is not associated with the built-in register protection (REG_PROT) feature included with many of the chip's modules. The non-secure write protection feature detailed here applies only to embedded flash memory.

Implementing non-secure write protection is almost identical to selecting specific flash blocks for an erase operation. The only difference is the register used. To provide non-secure write protection for a block, write a '1' to the corresponding mapped bit found in one of the C55FMC_LOCK0 or LOCK2 flash control registers.

NOTE

As indicated by the reset values shown in the following figure, after reset all blocks mapped to the register are, by default locked against programming and erase. Values written to the LOCK and SEL registers do not persist through device reset.

The following figure shows the C55FMC_LOCK0 register.

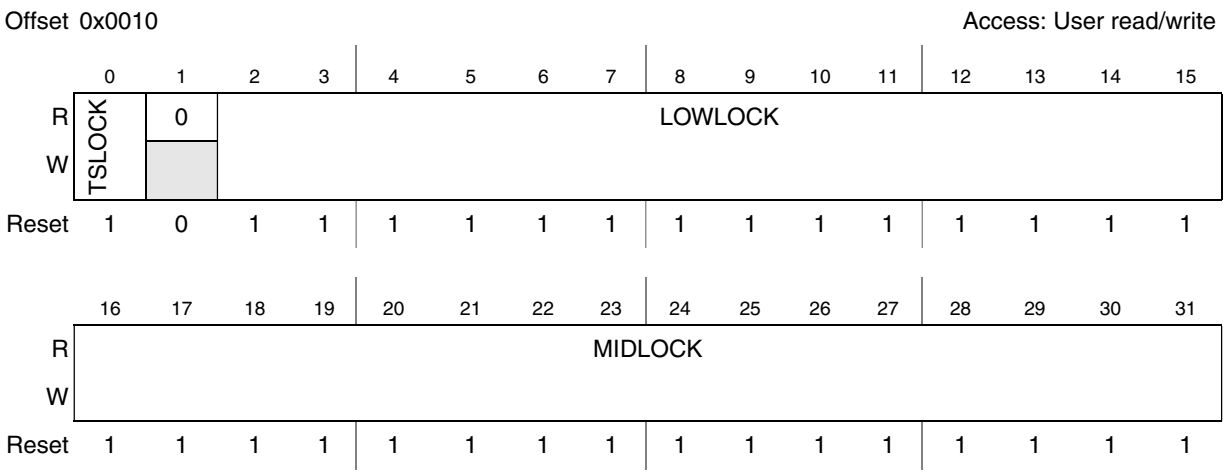


Figure 32-3. LOCK0 flash control register

The same bit mapping that applies to the SEL0 and SEL2 registers also applies to the LOCK0 and LOCK2 registers.

Example 2. Selecting blocks for lock

Referring to Table 32-1 (since bits in the SEL0 and LOCK0 registers map to the same flash blocks), writing the value 0x0030_0000 (same value used in Example 1) to the LOCK0 register (figure above) locks the following blocks (same blocks selected in Example 1) so they cannot be programmed or erased, even if selected for erase:

Table 32-3. Selecting blocks for lock

Start Address	End Address	Size (KB)	Partition	LOCK*n* Register	LOCK*n* Register Field Bit	Block No.	Description
0x00FAC000 0x08FAC000	0x00FAFFFF 0x08FAFFFF	16	1	LOCK0	LOWLOCK[4]	4	Code Flash block4
0x00F9C000 0x08F9C000	0x00F9FFFF 0x08F9FFFF	16	1	LOCK0	LOWLOCK[5]	5	Code Flash block5

As long as the LOCK0[LOWLOCK] bits are unchanged, the above blocks cannot be modified unless the protection is overridden (discussed in Secure write protection). All other blocks mapped to bits in the LOCK0 can be programmed because their corresponding bits have been set to '0'.

32.4 Security configuration planning

Designing the flash security framework consists of selecting the security features to be implemented and determining the scope of implementation for each selected feature.

The security features available in the MPC5746R microcontroller include:

- Secure write protection
- Secure read protection
- Debug port enable/disable
- One-time-programmable (OTP) flash memory block assignment

32.4.1 Creating password groups

Most basic security features in the MPC5746R microcontroller are password-based. Therefore, one of the early tasks that must be performed is the creation of four 256-bit passwords. They are programmed by the customer into the locations identified in the UTest memory map in the Memory Map chapter.

The UTest memory map fields to be written are:

- PASS Password Group 0
- PASS Password Group 1
- PASS Password Group 2
- PASS Password Group 3

Passwords must not be one of the following, which are considered insecure and therefore invalid:

- 0x0000_0000
- 0xFFFF_FFFF
- 0x5555_5555

NOTE

The password fields are in OTP UTest flash, so they can only be written once.

32.5 Secure write protection

Secure write protection refers to the chip's ability to implement up to four levels of 256-bit password-secured write protection for individual flash blocks. Use of this feature is optional but the default behavior of the MPC5746R microcontroller is to apply secure write protection. If there is a need to not write-protect specific blocks, that needs to be configured by creating the appropriate DCF records.

Note that the secure write protection settings override the non-secure write protection detailed in Non-secure write protection.

Implementing a custom secure write protection scheme requires creating DCF records that define the level of write protection to be applied to each flash block. The DCF records use a bit mapping scheme identical to the mapping in the flash module's SEL*n* and LOCK*n* registers. The write protection scheme becomes a permanent part of the device but blocks can be temporarily unlocked. Depending on the specific implementation, temporarily unlocking the write protection requires matching up to four 256-bit passwords.

The easiest way to understand the secure write protection functionality is to examine the role played by the PASS_LOCK0_PG*n*-PASS_LOCK3_PG*n* registers.

The PASS_LOCK0_PG*n*-PASS_LOCK3_PG*n* registers are four identical sets of registers contained in the PASS module. They are distinguished by the PG*n* suffix. The PG*n* (Password Group n) suffix indicates the password required to unlock that set of registers. Password 0 unlocks PASS_LOCK0_PG0, PASS_LOCK1_PG0, PASS_LOCK2_PG0, and PASS_LOCK3_PG0. Similarly, the registers suffixed with PG1 are unlocked by matching password 1, and so on.

The PASS_LOCK0_PG*n*-PASS_LOCK3_PG*n* register sets have two functions related to secure write protection:

- Reading the registers provides the secure write lock status for each flash block.
- Changing the value of a mapped bit in a register changes the secure write protect status of the corresponding flash block.

Each set of registers contains the same mapping of bits to flash blocks as the flash module's SEL*n* and LOCK*n* registers. In any of the four PASS_LOCK*x*_PG*n* register sets, if a mapped bit is set to '1', locking a flash block, that value can only be changed by successfully supplying the password value for that set of registers to gain write access to the register, and setting the bit to '0'. If the same bit is set to '1' in more than one

PASS_LOCK*x*_PG*n* register set, each must be unlocked using the corresponding password and the bit must be set to '0' before the flash block is available for programming or erase.

The following figure shows how the registers interact. Note that secure write protection is only affected by the settings in PASS_LOCK0_PG*n*-PASS_LOCK2_PG*n*. The PASS_LOCK3_PG*n* registers have additional security functions, discussed in later topics.

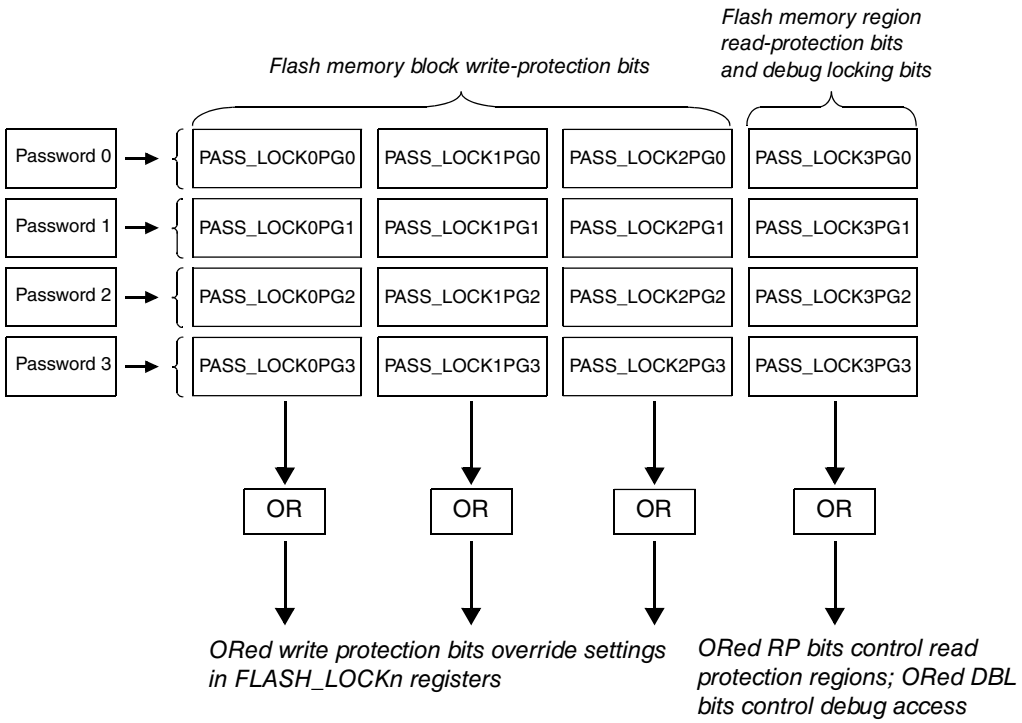


Figure 32-4. Secure write protection

The next sections discuss implementing secure write protection and overriding secure write protection.

32.5.1 Implementing secure write protection

Implementing secure write protection is a one-time task performed either at the factory or by the customer during chip configuration activities. It normally requires creating a DCF record for each of the PASS_LOCK0_PG*n*-PASS_LOCK2_PG*n* registers[1]. The DCF records have a structure that includes the same mapping of bits to flash memory blocks as

1. The default behavior for secure write protection in the case where passwords are created but no secure write protection DCF records are created, is for all blocks to have four levels of password-secured write protection. To vary the protection among all blocks, a DCF record for each password group must be created.

the flash module's SEL*n* and LOCK*n* registers. The values assigned become the permanent register reset values for the PASS_LOCK0_PG*n*-PASS_LOCK2_PG*n* registers.

See the Device Configuration Format (DCF) Records chapter for complete details on DCF records. This section summarizes the contents of a DCF record as required to implement secure write protection.

Even though the mapping of bits to flash memory blocks is identical to the mapping used for selecting flash blocks for erase of non-secure write locking, there are two significant differences:

1. Configuring flash blocks for secure write protection is done only during device configuration to customer specifications, either at the factory or by the customer during OEM production activities. The selection becomes a permanent part of the device configuration.
2. Configuration of blocks for secure write protection is done by writing *DCF records*, which is why the selection becomes a permanent part of the device.

NOTE

Careful planning is required before writing DCF records. They are created in a write-once area of flash. Multiple instances of any DCF record can be created, but in some cases only the record created first is used and with others only the most-recently written record is used. Be sure to understand the characteristics of a DCF record before creating it.

The DCF records required for implementing a secure write protection scheme are detailed in the PASS section of Table 6-5. Table entries provide the CS and Address value combinations that correspond to PASS_LOCK0_PG*n*-PASS_LOCK3_PG*n* registers.

The format of a secure write protection record is summarized here for convenience.

DCF records appear as contiguous double-word (64-bit) entries programmed in a reserved area of OTP UTest flash memory. The structure of a DCF Record is shown in the following figure.

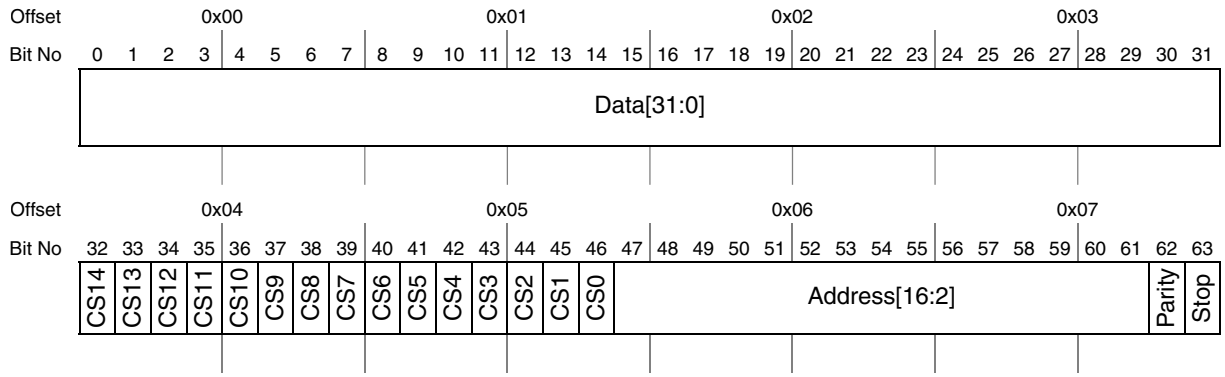


Figure 32-5. DCF record structure

NOTE

1. The values in the PASS_LOCK*x*PG*n* registers override the settings in the flash module's LOCK0-3 registers.
2. Values written to the PASS_LOCK*x*PG*n* registers do not persist through device reset. The reset value is determined by DCF record or, if no relevant DCF record has been created, each mapped bit has a reset value of '1', which causes the associated flash block to be locked against programming or erase.
3. The PASS_LOCK*x*PG*n* bits take effect when the life cycle is matured to "OEM Production" or older.

Example 3. Implementing secure write protection

Implementation of secure write protection is done during initial flash setup as follows:

1. Four 256-bit passwords (passwords 0-3) must be stored in a protected OTP area of device flash called UTest flash. Each password secures a set of registers in the PASS module (PASS_LOCK0_PG*n*-PASS_LOCK3_PG*n*) that have the same mapping of register bits to flash blocks as the flash module's SEL0-3 registers and LOCK0-3 registers.
2. DCF records must be created to define which flash memory blocks are to be locked and which blocks are not locked.

The records defined in step 2, determine the reset values of the PASS_LOCK0_PG*n*-PASS_LOCK3_PG*n* registers and therefore the write protection settings that take effect each time the microcontroller is reset.

Assume four passwords have been previously implemented and the following blocks are to be locked in the password 0 group:

Table 32-4. Flash blocks to implement secure write protection

Start Address	End Address	Size (KB)	Partition	LOCK*n* Register	LOCK*n* Register Field Bit	LOCK*n* DCF Record DATA Bit	Block No.	Description
0x00FAC000 0x08FAC000	0x00FAFFFF 0x08FAFFFF	16	1	LOCK0	LOWLOCK[4]	20	4	Code Flash block4
0x00F9C000 0x08F9C000	0x00F9FFFF 0x08F9FFFF	16	1	LOCK0	LOWLOCK[5]	20	5	Code Flash block5

From the PASS section of Table 6-5 :

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0000_1000	000_0000_0100_0000	LOCK0_PG0

Referring to the Figure 32-5, the values for the DCF record are:

- Data[31:0] is 0xFFFF_FFFF (bits 0-31) (same value used in previous examples)
- CS[14:0] is 0b000_0000_0000_1000 (bits 32-46)
- Address[16:2] is 0b000_0000_0100_0000 (bits 47-61)
- Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
- Recommended stop value is 0 for programming subsequent DCF records (see footnotes in the figure below).

The following DCF record provides secure write protection for the blocks in Table 32-1 by setting the reset values PASS_LOCK0_PG0 to the appropriate value to lock the blocks.

Bit No	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit No	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	x[1]	x[2]

NOTES:
1 Bit 62 indicates parity strategy, which is not used in the PASS DCF records so it is a “don’t care”.
2 Bit 63 indicates whether the current record is the last DCF. Generally, this should be written as ‘0’ so the first unprogrammed (all 1’s) DCF record acts as the stop record.

Figure 32-6. Sample DCF record for secure write protection

The remaining blocks mapped to the PASS_LOCK0_PG*n* registers do *not* have secure write protection provided by the password 0 register set because the DATA field of DCF record has only the bits mapped to flash blocks 4 and 5 set to '1'.

NOTE

The above example is intended to highlight the need to carefully understand how DCF records work. The protected blocks are secured by password 0 but by default they are also secured by passwords 1, 2, and 3. In contrast the blocks *not* secured by password 0 in this example *are* secured by passwords 1, 2, and 3.

See the PASS section of Table 6-5 for details on the DCF records.

32.5.2 Overriding secure write protection

The "PG*n*" (Password Group *n*) suffix on each the register names indicates the password each set of registers is associated with. For example, password 0, is used to unlock PASS_LOCK0_PG0, PASS_LOCK1_PG0, PASS_LOCK2_PG0, and PASS_LOCK3_PG0. The bits in the first three registers (PASS_LOCK0_PG0, PASS_LOCK1_PG0, and PASS_LOCK2_PG0) are mapped exactly the same as the bits in the flash module's SEL0-2 registers and LOCK0-2 registers.

Like the SEL0 and LOCK0 registers, the PASS_LOCK0_PG*n* registers all contain bits mapped to the low and mid address spaces, and the mapping scheme is the same. A mapped bit set to '1' in any of the registers causes the corresponding block to be locked against program or erase. A mapped bit set to '0' in any of the registers makes the corresponding block available for programming or erase.

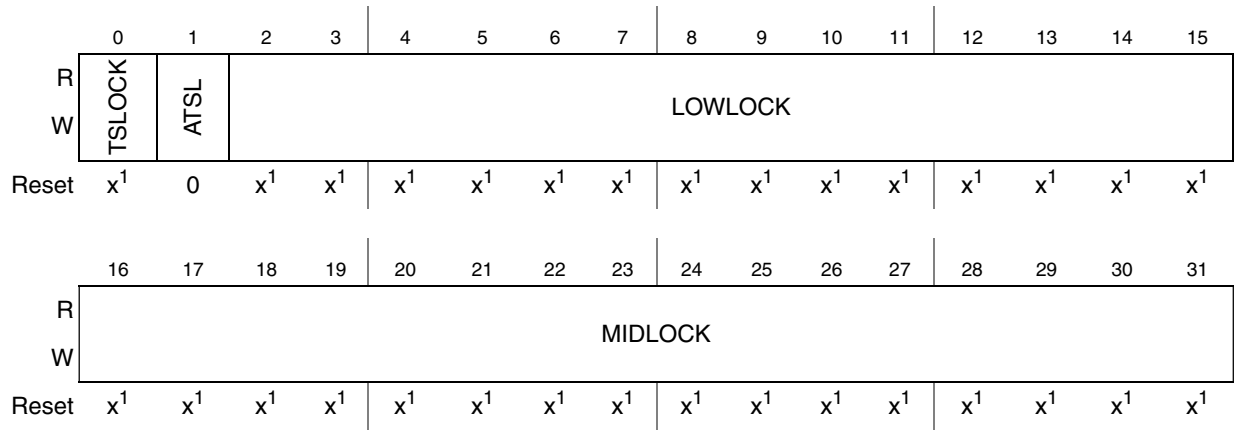
If a bit is set to '1', locking a flash block, that value can only be changed by successfully supplying the password value for that set of registers to gain write access, and setting the bit to '0'. If the same bit is set to '1' in more than one of the PASS_LOCKxPG*n* register sets, each must be unlocked using the corresponding password and the bit must be set to '0' before the flash block is available for programming or erase.

The following figure shows the PASS_LOCK0_PG*n* register. As you can see, it has the same structure as the flash module's LOCK0 register. In addition to the main difference of being among a group of four parallel sets of locking registers, the other difference between the PASS_LOCKx_PG*n* registers and the flash LOCK*n* registers is how the reset values are determined.

By default, all mapped bits are set to '1', locking their corresponding flash blocks from programming and erase. The reset value is user-configurable, though, by writing a DCF record. This is discussed in a later section.

Offset 0x0100 LOCK0_PG0
0x0110 LOCK0_PG1
0x0120 LOCK0_PG2
0x0130 LOCK0_PG3

Access: Read/Write



NOTES:
1 If no DCF records have been written to change the lock state, the bit will reset to '1'

Figure 32-7. LOCK0_PG*n* register

NOTE

1. The values in the PASS_LOCK*x*PG*n* registers override the settings in the flash module's LOCK0-3 registers.
2. Values written to the PASS_LOCK*x*PG*n* registers do not persist through device reset. The reset value is determined by DCF record or, if no relevant DCF record has been created, each mapped bit has a reset value of '1', which causes the associated flash block to be locked against programming or erase. The purpose of the registers is to indicate secure write protection status and to provide a way to temporarily override secure write protection.
3. The PASS_LOCK*x*PG*n* bits take effect after the life cycle is matured to "OEM Production" or older.

Example 4. Overriding secure write protection

Assume four passwords have been created but no DCF records have been written to set the reset values of the PASS_LOCK*x*_PG*n* registers, so the reset value for each mapped bit defaults to '1', causing all mapped blocks to have four levels of 256-bit password-secured write protection.

Since the settings in the PASS_LOCK*x*_PG*n* registers override the settings in the flash module's LOCK0-3 registers, the settings in those registers are irrelevant. Each of the four sets of the PASS module's LOCK registers must be unlocked by supplying the correct password for each set and then the appropriate bits must be set to '0' to make the desired flash blocks available for program or erase.

1. First write 0b00 to the PASS_CHSEL[GRP] register field to indicate registers controlled by password 0 are to be unlocked.
2. Write the 256-bit value for password 0 to the set of eight 32-bit password challenge input registers (PASS_CIN*n*) beginning with PASS_CIN0. The password must be written as a sequence of eight 32-bit writes, with the most significant bits of the password written to PASS CIN0.
3. Write 0xFFCF_FFFF to the PASS_LOCK0_PG0 register and verify a successful write.
4. Repeat the above steps for passwords 1-3.

Just as in previous examples, the result is the blocks are available for program and erase operations but at a much higher level of security.

32.6 Secure read protection

Read Protection is ability of the flash controller to block read accesses to the flash memory from all bus masters. This includes the blocking of:

- Instruction fetches and load instructions from the CPU
- Debug reads from an outside tool
- DMA reads
- SIPI reads
- Ethernet reads
- All other types of bus master reads

When secure read protection is active, the affected flash blocks are readable by any bus master until a debugger is attached to the JTAG port and debug is enabled. If a debugger is attached and debug is enabled while an application is running, the next access to a flash read protected region terminates with a bus error.

Not all instances of flash memory read protection in the MPC5746R microcontroller are configurable by the customer. An example of this is the 256-bit passwords stored in UTest flash memory. Password-based read protection is customer-configurable[2] and is similar to the password-based write protection mechanism detailed in Secure write protection.

There are two significant differences between password-based read protection and password-based write protection in the MPC5746R microcontroller.

1. Read protection can only be applied to pre-defined groups of flash memory blocks. These groups, called "Read Lock groups", cannot be re-defined, meaning the blocks assigned to those groups are permanently assigned and cannot be assigned to other groups and no new groups can be created.
2. Read protection is only active either when a debugger is attached to the JTAG port and debug is enabled; or the lifecycle has been advanced to "Failure Analysis".

Another difference is that only the PASS_LOCK3_PG*n* register for each password group is used:

- PASS_LOCK3_PG0
- PASS_LOCK3_PG1
- PASS_LOCK3_PG2
- PASS_LOCK3_PG3

See PASS_LOCK3_PGn register bit mapping for mappings of PASS_LOCK3_PG*n* register bits to flash blocks.

The next sections discuss implementing secure read protection and overriding secure write protection.

32.6.1 Implementing secure read protection

Implementation of secure read protection is done during initial flash setup as follows:

1. Four 256-bit passwords (passwords 0-3) are stored in a protected OTP area of device flash named UTest flash. Each password secures a set of registers in the PASS module (PASS_LOCK0_PG*n*-PASS_LOCK3_PG*n*) that have the same mapping of register bits to flash blocks as the flash module's LOCK0-3 registers and LOCK0-3 registers. These are the same passwords used by secure write protection and other secure functions. The passwords are only created once.

2. Secure read protection is user-configurable in that the default lock/unlock state for each mapped block at reset is controlled by user-created DCF records, giving the user the ability to determine the number of levels of password protection implemented. The read protection is applied to groups of blocks instead of individual blocks and the groups are not user-definable.

2. DCF records must be created to define which flash memory blocks are to be locked. Those records determine the reset values of the PASS_LOCK3_PG*n* registers. Like the other PASS_LOCK*x*_PG*n* registers, the PASS_LOCK3_PG*n* registers have two functions.
   - Reading the registers provides a lock status for each Read Locking group.
   - Changing the value of a mapped bit in a register changes the read protect status of the corresponding Read Locking group.

The layout of the PASS_LOCK3_PG*n* registers is shown in the following figure. The fields relevant to implementing secure read protection are RL4-RL0.

Offset 0x010C LOCK3_PG0
0x011C LOCK3_PG1
0x012C LOCK3_PG2
0x013C LOCK3_PG3

Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PGL	DBL	MO	0	MSTR				0	0	0	0	0	RL2	RL1	RL0
W																
Reset	x[1]	x[2]	x[2]	0	1	1	1	1	0	0	0	x[2]	x[2]	x[2]	x[2]	x[2]

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	256LCK_U															
W																
Reset	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]

NOTES:
1 Reset value depends on life cycle. If life cycle is Customer delivery or earlier, the value will be 0, otherwise 1.
2 If no DCF records have been written to change the lock state, the bit will reset to '1'.

Figure 32-8. PASS_LOCK3_PG*n* Register

Example 5. Implementing secure read protection

Assume four passwords have been previously implemented.

The following table shows the mapping of flash blocks to Read Locking groups.

Table 32-6. Low and mid address space flash block to Read Locking group mapping

Start Address	End Address	Size (KB)	Partition	SEL*n* Register	SEL*n* Register Field Bit	Read Lock Group	Block No.	Description
0x00404000	0x00407FFF	16	0	SEL0	LOWSEL[0]	--	0	16KB BAF block0[1]

Table continues on the next page...

Table 32-6. Low and mid address space flash block to Read Locking group mapping (continued)

Start Address	End Address	Size (KB)	Partition	SEL*n* Register	SEL*n* Register Field Bit	Read Lock Group	Block No.	Description
0x00FA0000 0x08FA0000	0x00FA3FFF 0x08FA3FFF	16	0	SEL0	LOWSEL[1]	1	1	Code Flash block1
0x00FA4000 0x08FA4000	0x00FA7FFF 0x08FA7FFF	16	0	SEL0	LOWSEL[2]	1	2	Code Flash block2
0x00FA8000 0x08FA8000	0x00FABFFF 0x08FABFFF	16	1	SEL0	LOWSEL[3]	1	3	Code Flash block3
0x00FAC000 0x08FAC000	0x00FAFFFF 0x08FAFFFF	16	1	SEL0	LOWSEL[4]	1	4	Code Flash block4
0x00F9C000 0x08F9C000	0x00F9FFFF 0x08F9FFFF	16	1	SEL0	LOWSEL[5]	1	5	Code Flash block5
0x00FB0000 0x08FB0000	0x00FB7FFF 0x08FB7FFF	32	0	SEL0	LOWSEL[6]	1	0	Code Flash block0
0x00FB8000 0x08FB8000	0x00FBFFFF 0x08FBFFFF	32	1	SEL0	LOWSEL[7]	1	1	Code Flash block1
0x00FC0000 0x08FC0000	0x00FCFFFF 0x08FCFFFF	64	0	SEL0	LOWSEL[8]	1	0	Code Flash block0
0x00FD0000 0x08FD0000	0x00FDFFFF 0x08FDFFFF	64	0	SEL0	LOWSEL[9]	1	1	Code Flash block1
0x00FE0000 0x08FE0000	0x00FEFFFF 0x08FEFFFF	64	1	SEL0	LOWSEL[10]	1	2	Code Flash block2
0x00FF0000 0x08FF0000	0x00FFFFFF 0x08FFFFFF	64	1	SEL0	LOWSEL[11]	1	3	Code Flash block3
0x00800000	0x00803FFF	16	2	SEL0	MIDSEL[0]	2	0	FMC EEPROM Block0
0x00804000	0x00807FFF	16	2	SEL0	MIDSEL[1]	2	1	FMC EEPROM Block1
0x00808000	0x0080BFFF	16	2	SEL0	MIDSEL[2]	2	2	FMC EEPROM Block2
0x0080C000	0x0080FFFF	16	2	SEL0	MIDSEL[3]	2	3	FMC EEPROM Block3
0x00810000	0x00813FFF	16	3	SEL0	MIDSEL[4]	2	4	FMC EEPROM Block4
0x00814000	0x00817FFF	16	3	SEL0	MIDSEL[5]	2	5	FMC EEPROM Block5
0x00818000	0x0081BFFF	16	3	SEL0	MIDSEL[6]	2	6	FMC EEPROM Block6

Table continues on the next page...

Table 32-6. Low and mid address space flash block to Read Locking group mapping (continued)

Start Address	End Address	Size (KB)	Partition	SEL*n* Register	SEL*n* Register Field Bit	Read Lock Group	Block No.	Description
0x0081C000	0x0081FFFF	16	3	SEL0	MIDSEL[7]	2	7	FMC EEPROM Block7
0x00820000	0x0082FFFF	64	2	SEL0	MIDSEL[8]	2	0	FMC EEPROM Block0
0x00830000	0x0083FFFF	64	3	SEL0	MIDSEL[9]	2	1	FMC EEPROM Block1

1. The Boot Assist Flash (BAF) is one time programmable and the majority of it is programmed in the factory. The BAF handles the initial start up of the device.

To provide two level of password protection for only Read Lock group 2 and leave all other groups unlocked, the RL2 bit can be set in any two of the four PASS_LOCK3_PG*n* registers. For this example, we will choose password groups 0 and 1.

We need to create DCF records that define the reset values for all RLn bits as 0, with the exception of PASS_LOCK3_PG0 and PASS_LOCK3_PG1. The information required is from the PASS section of the Table 6-5.

Table 32-7. Inplementing secure read protection

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0000_1000	000_0000_0100_0011	LOCK3_PG0
000_0000_0000_1000	000_0000_0100_0111	LOCK3_PG1
000_0000_0000_1000	000_0000_0100_1011	LOCK3_PG2
000_0000_0000_1000	000_0000_0100_1111	LOCK3_PG3

Referring to Figure 32-5, the values for the DCF record are:

- For LOCK3_PG0
  - Data[31:0] is 0xFFFF_FFFF (bits 0-31)
  - CS[14:0] is 0b000_0000_0000_1000 (bits 32-46)
  - Address[16:2] is 0b000_0000_0100_0011 (bits 47-61)
  - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
  - Preferred Stop value is 0.
- For LOCK3_PG1
  - Data[31:0] is 0xFFFF_FFFF (bits 0-31)
  - CS[14:0] is 0b000_0000_0000_1000 (bits 32-46)
  - Address[16:2] is 0b000_0000_0100_0111 (bits 47-61)

    - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
    - Preferred Stop value is 0.
- For LOCK3_PG2
    - Data[31:0] is 0xFFFF_FFFF (bits 0-31)
    - CS[14:0] is 0b000_0000_0000_1000 (bits 32-46)
    - Address[16:2] is 0b000_0000_0100_1011 (bits 47-61)
    - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
    - Preferred Stop value is 0.
- For LOCK3_PG3
    - Data[31:0] is 0xFFFF_FFFF (bits 0-31)
    - CS[14:0] is 0b000_0000_0000_1000 (bits 32-46)
    - Address[16:2] is 0b000_0000_0100_1111 (bits 47-61)
    - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
    - Preferred Stop value is 0.

32.6.2 Overriding secure read protection

As noted previously, the PASS_LOCK3_PGn registers contain five read protection fields, RP0-RP2, that correspond to pre-defined Read Lock groups. To temporarily unlock a Read Lock group, the appropriate PASS_LOCK3_PG*n*[RL*n*] bits must be set to 0.

Like the register bits controlling secure write protection, the reset value of bits controlling secure read protection is determined by DCF record or, in the absence of a relevant DCF record, each bit resets to '1', causing the blocks associated with each read group to be read locked.

Example 6. Overriding secure read protection

Building on Example 5, where Read Lock group 2 was protected by passwords 0 and 1, temporarily overriding secure read protection for that group is accomplished by first unlocking the PASS_LOCK3_PG0 and PASS_LOCK3_PG1 registers and then setting the RL2 bit to '0' in both.

1. First write 0b00 to the PASS_CHSEL[GRP] register field to indicate registers controlled by password 0 are to be unlocked.
2. Write the 256-bit value for password 0 to the set of eight 32-bit password challenge input registers (PASS_CIN*n*) beginning with PASS_CIN0. The password must be written as a sequence of eight 32-bit writes, with the most significant bits of the password written to PASS CIN0.
3. Write a 0 to PASS_LOCK3_PG0[RL2] bit and verify a successful write.
4. Repeat the above steps for password 1.

The result of a successful sequence is that the blocks associated with the unlocked read lock group can be read by a debugger.

32.7 Debug port enable/disable

The mechanism for temporarily enabling the MPC5746R debug port after it has been locked by its life cycle state is similar to the flash memory read protection mechanism detailed in Secure read protection. The difference is that the PASS_LOCK3_PG*n*[DBL] field is used instead of the RP0-RP2 fields.

32.8 Tamper detection

The Tamper Detection Module (TDM) provides a type of flash memory write protection mechanism that forces software to write a record associated with one or more blocks in a Tamper Detection Region (TDR) before the block(s) can be erased. An additional feature is provided to enable customers to override the protection mechanism for any TDR so that no record is required to be written before a block within the TDR can be erased.

This section provides details on the mapping between DCF record bits and individual flash memory blocks to be selected for tamper detection.

See the Tamper Detection Module (TDM) chapter for details on tamper detection, including the DCF clients and two additional memory-mapped registers.

There are two significant differences between selecting blocks for tamper detection and selecting blocks for other activities, e.g., a flash erase operation:

1. Selection of blocks for tamper detection protection is done only during device configuration to customer specifications, either at the factory or by the customer during OEM production activities. The selection becomes a permanent part of the device configuration.
2. The selection of blocks for tamper detection is done by writing *DCF records*, which is why the selection becomes a permanent part of the device.

The Tamper Detect section of Table 6-5 gives the CS and Address value combinations that correspond to groups of LOCK0-LOCK3 registers, similar to the flash module's LOCK0-LOCK3 registers.

The mapping of individual flash blocks to TDM DCF record data field bits is shown in included in the detailed flash block map in the OTPENn DCF clients.

32.8.1 Implementing tamper detection

Implementing tamper detection requires:

1. Creating the tamper detect diary
2. Assigning blocks to Tamper Detection Regions (TDRs)

The tasks are detailed in the following sections

32.8.2 Creating the tamper detect diary

Tamper detection in the MPC5746R microcontroller is user-defined. The basic functionality provided is an erase counter, or diary, for 6 user-defined regions, called "tamper detect regions", or TDRs, within the flash memory. Each TDR is defined using a set of DCF records that is similar in function and layout to the flash module's SEL*n* and LOCK*n* registers, which contain bits matched to specific flash blocks.

For a program or erase operation to be performed on a block included in a TDR, a record must be written to a "diary" that tracks flash modification operations. The format and size of records is customer-defined. The only hardware constraints are that each diary has a maximum size of 2 KB and the minimum size of any programming operation is 8 bytes.

NOTE

The limiting factor for tamper detection records is the diary size. The total diary size is given by the number of TDRs (6 for MPC5746R) multiplied by 2 KB, the portion of the diary allotted to records associated with each TDR.

Since the minimum flash programming operation size is 8 bytes, the maximum number of tamper detect records for a given TDR is 256 8-byte entries.

The diary can be placed within any flash block in the flash array but to maintain the security of the TDR, there are two constraints:

- The block where the diary is placed must be assigned as OTP within the OTP registers (covered in the next section)
- The diary base address must be on a 4 KB boundary, therefore the least significant 12 bits of the base address must all be '0'

NOTE

The tamper detect diary base address DCF record is a "write once" record. All subsequent tamper detect diary base address DCF records created are ignored.

Example 7. Creating the tamper detect diary

If the total space required for a diary is 12 KB (as in the case of 6 TDRs), any flash block can be used because the smallest flash blocks in the MPC5746R microcontroller are 16 KB. Since the diary base address must be on a 4 KB, boundary, though, if the diary base address is within a 16 KB flash block, it can only be assigned to the base address of the block or the address given by the base address plus 4 KB.

If the base address of the 16 KB block is 0x00FC4000, the base address of the diary is given by:

- 0b0000_0000_1111_1100_0000_0000_0000_0000 (16 KB base address), or
- 0b0000_0000_1111_1100_0001_0000_0000_0000 (16 KB base address + 4 KB)

The base address of each TDR's section of the diary is:

- TDR0 = Diary_Base_Address + 0b00_0000_0000_0000
- TDR1 = Diary_Base_Address + 0b00_1000_0000_0000
- TDR2 = Diary_Base_Address + 0b01_0000_0000_0000
- TDR3 = Diary_Base_Address + 0b01_1000_0000_0000
- TDR4 = Diary_Base_Address + 0b10_0000_0000_0000
- TDR5 = Diary_Base_Address + 0b10_1000_0000_0000
- End of the diary = Diary_Base_Address + 0b11_0000_0000_0000

We need to create one DCF record that defines the base address for the diary. The information required is from the Tamper Detect section of Table 6-5.

Table 32-8. Creating the tamper detect diary

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0001_0000	000_0000_0000_0000	Diary Base Address

Referring to Figure 32-5, the values for the DCF record are:

- For LOCK3_PG0
  - Data[31:0] is 0x00FC4000 (the base address)
  - CS[14:0] is 0b000_0000_0001_0000 (bits 32-46)
  - Address[16:2] is 0b000_0000_0000_0000 (bits 47-61)
  - Parity value is irrelevant because it is not used in Tamper Detect DCF records (bit 62)
  - Preferred Stop value is 0.

See the Tamper Detect section of Table 6-5 for details.

NOTE

User software must check whether the diary is full. If the diary area for a TDR is full, user software must program a DCF record to override blocking of erase from TDR, disabling tamper detect, so that erases can be done to that TDR without any diary entry. This is detailed in Overriding tamper detection.

32.8.3 Assigning blocks to Tamper Detection Regions (TDRs)

Tamper detection regions are defined by creating TDR*x*_LOCK*n* DCF records. For example, in a microcontroller with six tamper detection regions, the following records define the TDRs:

- TDR0_LOCK0, TDR0_LOCK1, TDR0_LOCK2, TDR0_LOCK3
- TDR1_LOCK0, TDR1_LOCK1, TDR1_LOCK2, TDR1_LOCK3
- TDR2_LOCK0, TDR2_LOCK1, TDR2_LOCK2, TDR2_LOCK3
- TDR3_LOCK0, TDR3_LOCK1, TDR3_LOCK2, TDR3_LOCK3
- TDR4_LOCK0, TDR4_LOCK1, TDR4_LOCK2, TDR4_LOCK3
- TDR5_LOCK0, TDR5_LOCK1, TDR5_LOCK2, TDR5_LOCK3

Each set of TDR*x*LOCK*n* DCF records mapped in exactly the same way as the flash module's SEL*n* and LOCK*n* registers (see the detailed flash block map in TDRx_LOCKn DCF clients for the mapping).

In the TDR*x*LOCK*n* DCF records:

- A bit set to '1' means the corresponding flash block is assigned to this TDR
- A bit set to '0' means the corresponding flash block is NOT assigned to this TDR

The default state (no TDRs defined) is defined as 'Block NOT assigned to the TDR'.

32.8.4 Overriding tamper detection

When a TDR's diary section is full, a Tamper Override (TO) record must be created before a block assigned to the TDR can be programmed or erased.

The details of the data section of the DCF record used to override tamper detection is shown in the following figure.

NOTE

Bits in this DCF record is writable from 0 to 1only. Attempted writes of bits from 1 to 0 are ignored.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
W	0	0	0	0	0	0	0	0	0	0	TOE5	TOE4	TOE3	TOE2	TOE1	TOE0

Figure 32-9. Data portion of Tamper Region Override (TO) DCF record

Example 8. Implementing secure write protection

Assume the diary Tamper Detection Region 3 is full

We need to create one DCF record that defines that overrides tamper detection for the tamper detection region whose diary is full. The information required is from the Tamper Detect section of Table 6-5.

Table 32-9. Implementing secrure write protection

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0001_0000	000_0000_0000_0001	Tamper Region Override

Referring to Figure 32-5, the values for the DCF record are:

- Data[31:0] is 0x0000_0008 (TOE3)
- CS[14:0] is 0b000_0000_0001_0000 (bits 32-46)
- Address[16:2] is 0b000_0000_0000_0001 (bits 47-61)
- Parity value is irrelevant because it is not used in Tamper Detect DCF records (bit 62)
- Preferred Stop value is 0.

See the Tamper Detect section of Table 6-5 for details.

32.9 Implementing OTP

Any flash block within the flash array can be assigned, at any time, to be OTP. OTP means that flash erase of the entire block is disabled and only 64-bit double words that are already erased (i.e., flash content is 0xFFFF_FFFF_FFFF_FFFF) can be programmed. Over-programming is not possible.

Flash blocks are assigned as OTP by writing a DCF record. The block becomes OTP after the next reset.

NOTE

After a flash block is configured as OTP, it cannot be changed back.

This capability resides within the Tamper Detection Module (TDM). The records are detailed in Table 6-5. See the entries for the OTP_EN0, OTP_EN1, OTP_EN2, and OTP_EN3 DCF records in the Tamper Detect section of the table.

Chapter 33
Analog Peripheral Modules

33.1 Analog sub-system overview

The MPC5746R analog sub-system contains a total of four independent 12-bit SAR ADCs, three independent 16-bit Sigma-Delta ADCs, two Decimation Filter blocks, an ADC Cross-Triggering Unit (BCTU), two customer readable temperature sensors, and bandgap voltage reference.

Each ADC instance has an independent dedicated programming interface. The Decimation filter blocks are used for digital signal processing on the ADC conversion results. The BCTU is used to trigger SAR conversions, and to channel conversion results to the Reaction Module block on the device. The SAR ADCs are used for general purpose A/D conversions, and to read internal nodes on the device (temperature sensors, bandgap voltage reference, device supply voltages, LVD/HVD Comparator references, and SAR ADC references).

33.1.1 Block diagram

The analog sub-system block diagram for the MPC5746R is given in Figure 33-1.

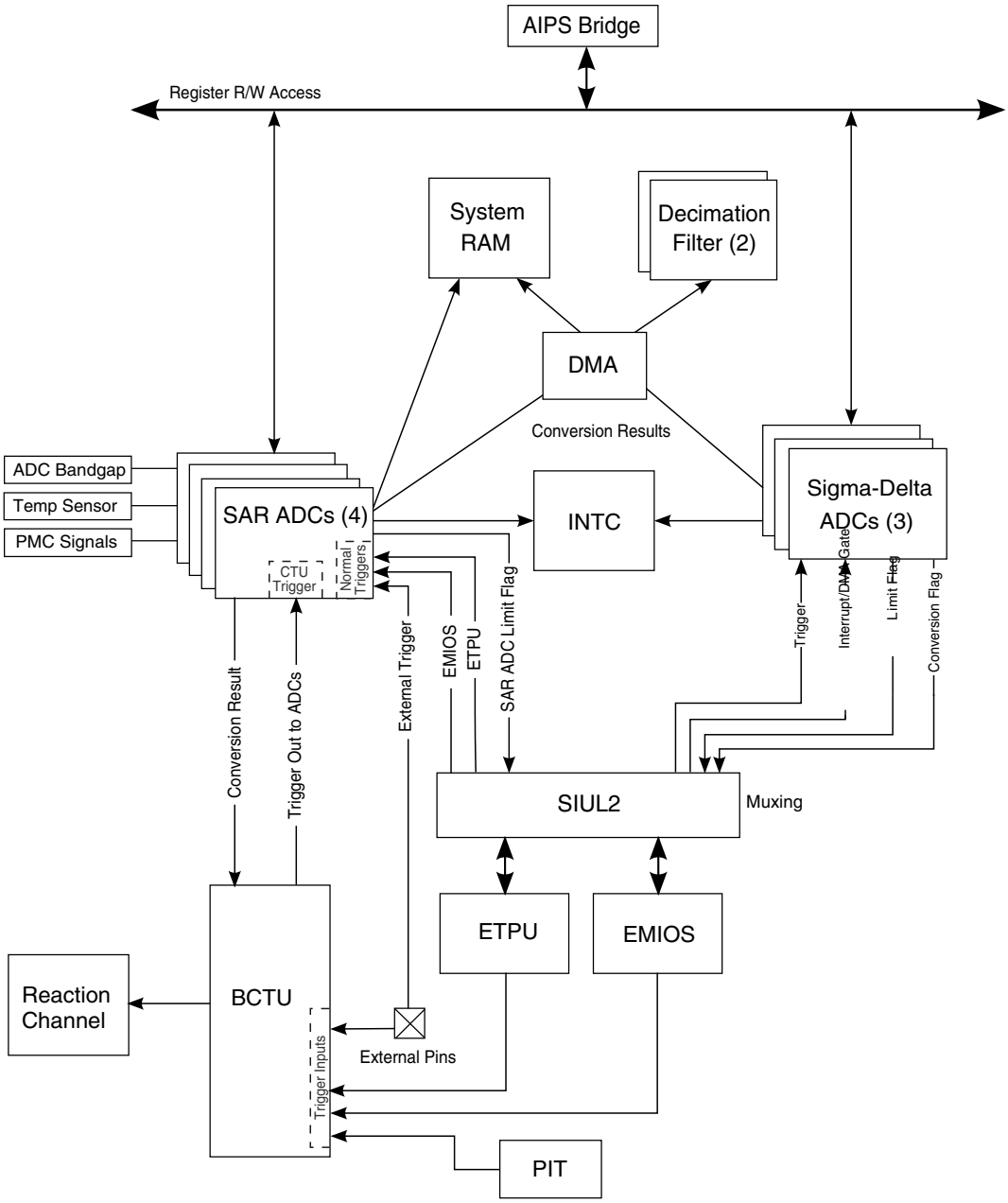


Figure 33-1. Analog sub-system

Chapter 34
Sigma Delta Analog-to-Digital Converter (SDADC) Digital Interface

34.1 Sigma Delta Analog-to-Digital Converter (SDADC) digital interface chip-specific information

34.1.1 Introduction

The following sections contain the chip-specific information for the on-chip SDADC modules. See Sigma Delta Analog-to-Digital Converter (SDADC) digital interface reference for a detailed description of the SDADC modules.

34.1.2 Sigma-Delta ADC features

There are three Sigma Delta ADCs on the MPC5746R. Each Sigma-Delta ADC has the following features:

- 16-bit resolution
- Single-ended or differential input
- Pass band 10 Hz - 100 kHz
- Output sample rate 30 kHz - 333 KHz (full performance)
- Programmable gain (1, 2, 4, 8, 16)
- Biasing of unused single-ended input to VREF/2 or ground
- Biasing of differential inputs to VREF/2 via 100 kΩ
- Optional high pass filter
- Interrupt and DMA support
- Synchronous start of all converters
- Conversion limit watchdog
- Hardware start trigger and mux advance/wrap-around

See the *MPC5746R Microcontroller Data Sheet* for electrical specifications.

34.1.3 Sigma-Delta ADC block diagram

Figure 34-1 gives the block diagram of the Sigma-Delta ADC and the integration into the MPC5746R. The SDADC block is comprised of three pieces: the converter, the bias generator, and the digital interface (shown inside the dotted-line box). All three Sigma Delta ADCs have the same features and integration on the MPC5746R.

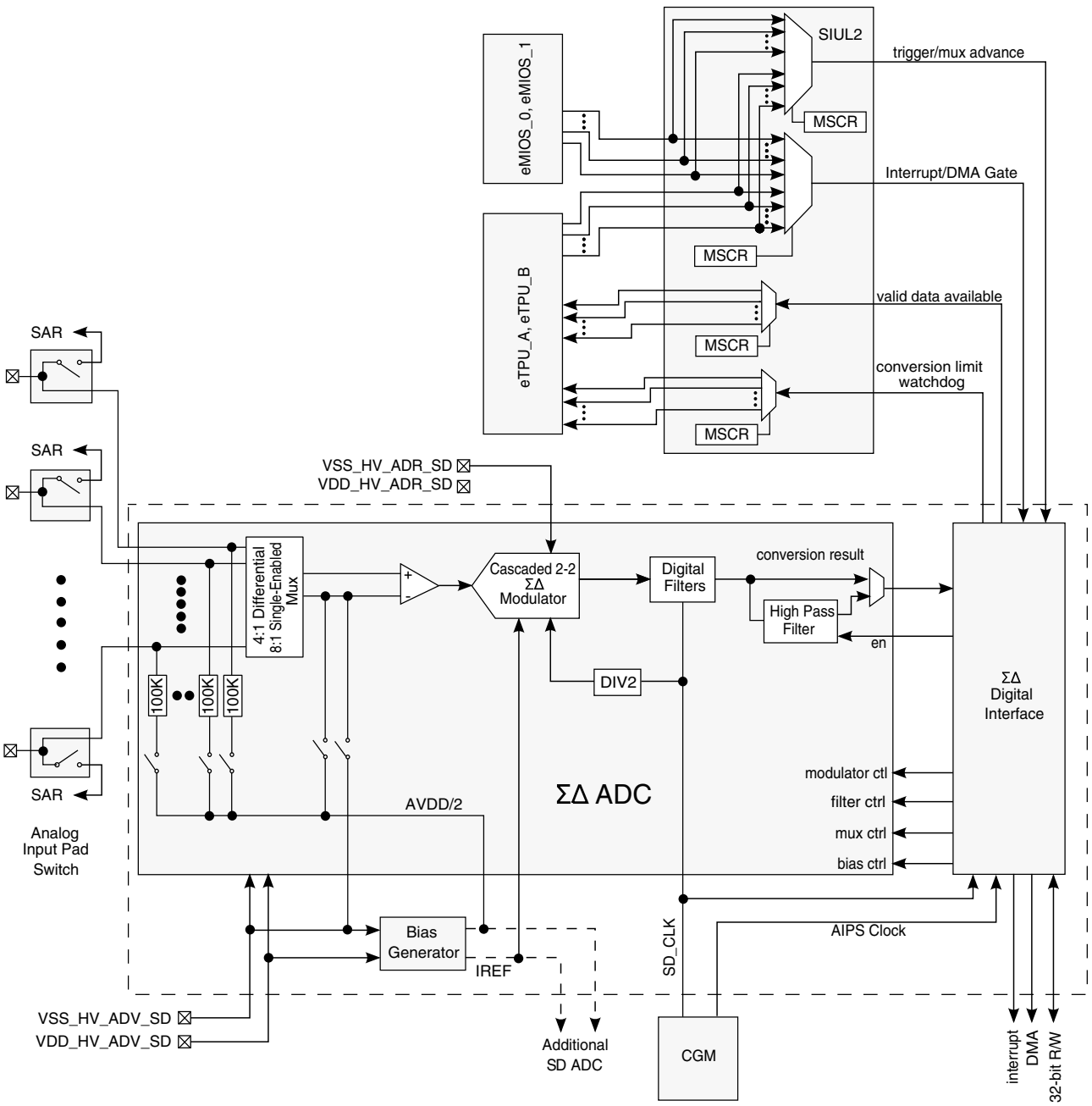


Figure 34-1. MPC5746R Sigma Delta ADC

34.1.4 Sigma-Delta ADC external signal description

The Sigma-Delta ADC external package pins are defined in Table 34-1 and the following sections.

Please refer to the *MPC5746R Microcontroller Data Sheet* for voltage level on signals.

Table 34-1. Sigma-Delta ADC external signal description

Signal Name	Signal Type	Description
PZ[n],PY[n]	Analog Input	Single-Ended or Differential Analog Input
VDD_HV_ADV_SD	Analog Supply	Sigma-Delta ADC Analog Power Supply
VSS_HV_ADV_SD	Analog Ground	Sigma-Delta ADC Analog Ground
VDD_HV_ADR_SD	Analog Supply	Sigma-Delta ADC Reference
VSS_HV_ADR_SD	Analog Ground	Sigma-Delta ADC Ground Reference

34.1.4.1 VDD_HV_ADV_SD/VSS_HV_ADV_SD

The Sigma-Delta ADCs on the MPC5746R share a single supply/ground pair in the packages. The VDD_HV_ADV_SD/VSS_HV_ADV_SD pins are dedicated to the Sigma-Delta ADCs, and are not shared with the SAR ADCs on the device.

See the *MPC5746R Microcontroller Data Sheet* for electrical characteristics.

34.1.4.2 VDD_HV_ADR_SD/VSS_HV_ADR_SD

The Sigma-Delta ADCs on the MPC5746R share a single reference pair in the packages, that is dedicated and not shared with the SAR ADCs.

34.1.5 Sigma-Delta ADC analog input multiplexing

The I/O Signal Description and Input Multiplexing Tables are included in a Microsoft Excel file attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab. Search for "SAR ADC" or "Sigma-Delta ADC" in the I/O Signal Table worksheet to locate the analog input pins.

NOTE

The only configuration option for an analog input is to shut off the digital input buffer to reduce leakage. See the SIUL2 chapter for details.

34.1.6 Sigma-Delta digital interface

Each SDADC has a dedicated digital interface comprised of a set of registers and control functions. All features in the digital interface apply to and are available for each Sigma-Delta with the following exceptions:

- The mapping of input channels is not the same for each ADC and is defined in Table 34-1
- The mapping of the DMA/interrupt gating input and end of conversion signal output to the ETPU is not the same for each ADC and is defined in the SIUL2 chapter of this document.
- External modulator clock and bitstream support is not included on the MPC5746R

34.1.7 Sigma-Delta ADC clock architecture

SDADC clock architecture is discussed in the following sections.

34.1.7.1 Clock source

Each Sigma-Delta ADC digital interface on the MPC5746R has a register clock input, which is separate from the module clock input. The register interface is clocked by the AIPS_CLK. The module clock is sourced by the modulator clock (SD_CLK). See the Clocking chapter for more detail on the clock sources. The input clock for the Sigma-Delta modulator is the SD_CLK. Figure 34-2 gives a block diagram of the required Sigma-Delta ADC clock architecture for the MPC5746R. The frequency range for the SD_CLK is 4 - 16 MHz.

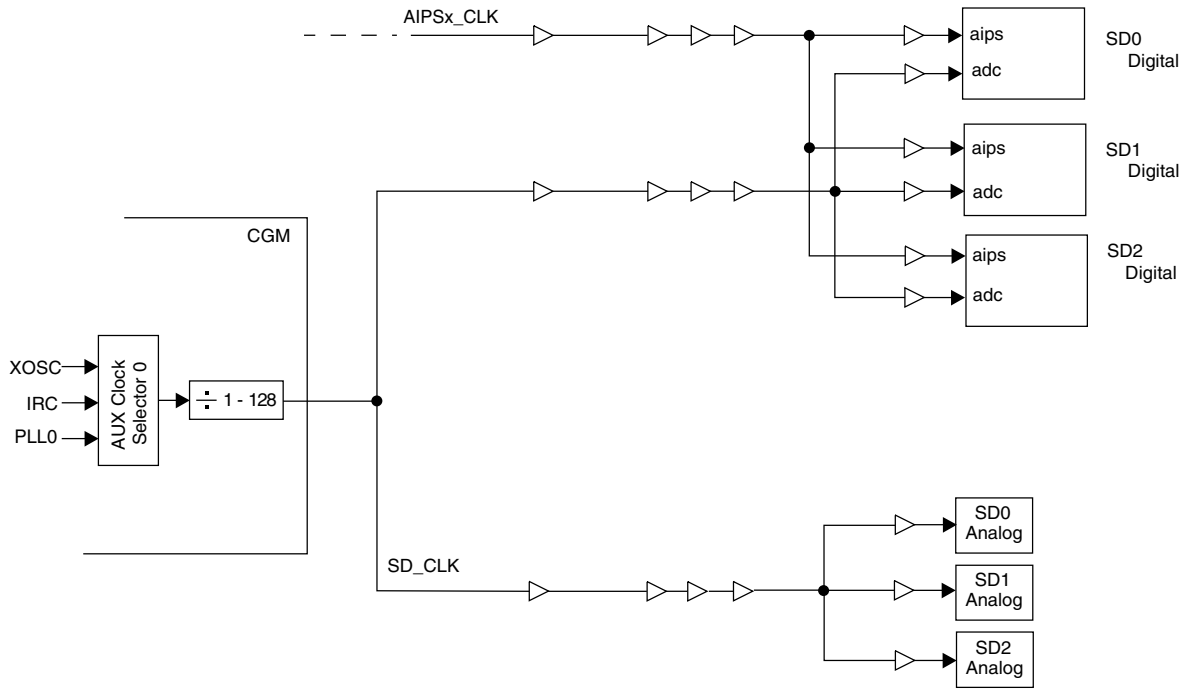


Figure 34-2. MPC5746R Sigma Delta ADC clock diagram

34.1.8 Sigma-Delta ADC starting and triggering

Each Sigma Delta ADC can be started by software or ETPU or EMIOS channel outputs. The external trigger input of the Sigma Delta ADC allows starting of the converter and advancing the channel mux. Mapping of ETPU and EMIOS channels to the Sigma Delta trigger inputs is done in the SIUL2 input multiplexing registers, and defined in the I/O spreadsheet for the device.

NOTE

A minimum startup time of 100 µs needs to be given to SDADC after enabling.

34.1.8.1 Simultaneous start of multiple Sigma-Delta converters

Each Sigma-Delta ADC on the device supports starting the converter with a software write to the start register in the digital interface. The trigger outputs of each Sigma-Delta ADC are connected to the trigger input of each other Sigma-Delta ADC and its own to allow simultaneous starting of all ADCs. Simultaneously is defined as all selected converters start sampling and provide results on the same SD_CLK clock edge.

The user controls which ADCs are simultaneously started by selecting the trigger input from the ADC that will initiate the simultaneous start and enabling the ADC. Then a software or hardware trigger start of the initiating ADC will cause all linked ADCs to start also.

Trigger selection is specified using the SDADC_MCR[TRIGSEL] field (see TRIGSEL)

NOTE

The possible SDADC_MCR[TRIGSEL] field values are specified generically. Use the following values for selecting triggers for the MPC5746R SD ADC:

- 0000 – SDADC_0 output trigger
- 0001 – SDADC_1 output trigger
- 0010 – SDADC_2 output trigger
- 0011 – SDADC START_TRIGGER MSCR
- 0100 – 1111 Reserved

The connections for the simultaneous start of the converters is given in Figure 34-3.

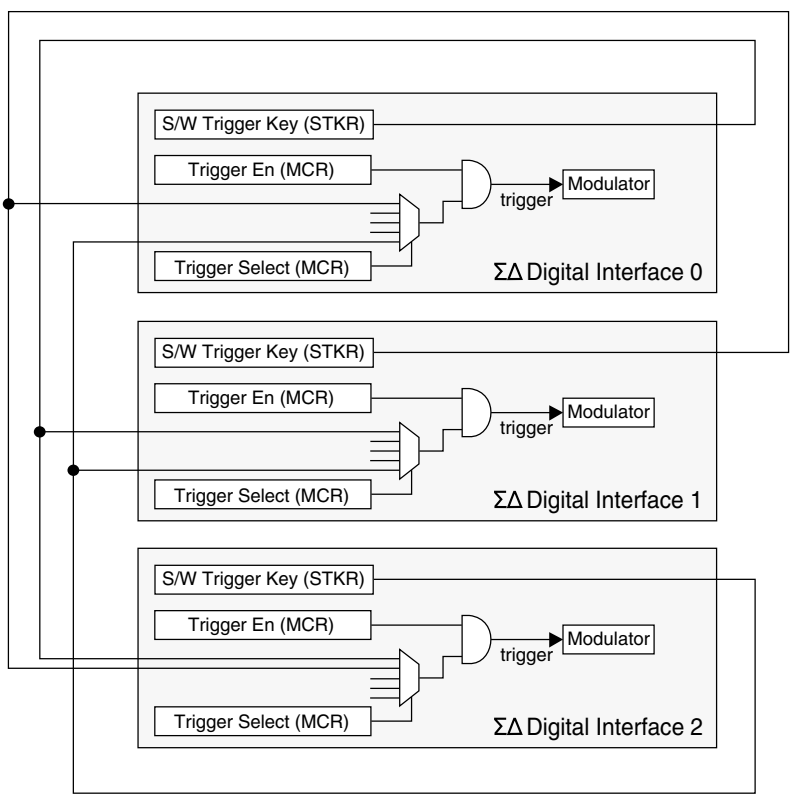


Figure 34-3. Sigma-Delta ADC simultaneous start connectivity

34.1.8.2 Sigma Delta mux input advance and wrap around

The Sigma Delta input mux advance and wrap around feature is enabled and controlled in the digital interface of the Sigma Delta ADC. The advancing of the mux can be controlled in the interface by software or by external ETPU and EMIOS channels. The mapping of ETPU and EMIOS output channels to the mux advance input for each Sigma Delta ADC is controlled in the SIUL2 MSCR registers, and the detailed mapping is given in the input muxing table of the device I/O definition spreadsheet for the device.

34.1.9 Sigma-Delta ADC input biasing

The Sigma Delta ADC inputs can be biased internally on the device. The differential inputs can be biased to half-scale through a 100 KΩ resistor. The unused single-ended input can be biased to half-scale or ground. This is controlled in the SDADC_MCR and SDADC_CSR registers.

34.1.10 Sigma-Delta ADC conversion result processing

Upon start of input sampling, a valid conversion value is available in the result register after twice the group delay of the Sigma Delta ADC. In the case of a switch of the input mux, an additional 100 µs is required for the mux settling time. Each Sigma Delta ADC has an independent 16-bit result register.

The Sigma Delta ADC always returns a 16-bit signed result with a result FIFO depth of 16.

NOTE

The Sigma-Deltas on the MPC5746R have a fixed FIFO depth of 16. The FIFO depth is set at the factory and is not programmable but the amount of available FIFO space used in an application is programmable using the SDADC_FCR[FTHLD] field.

An option for an interrupt/DMA is available for FIFO full. Optional interrupt requests are available for FIFO empty and FIFO overrun conditions.

34.1.10.1 Optional high-pass filter of conversion result

There is an optional high-pass filter that the output conversion result can be fed through before being captured in the result register in the digital interface. The filter is enabled/ disabled in the digital interface. The -3 db point of the filter is defined in the *MPC5746R Microcontroller Data Sheet*.

34.1.10.2 Interrupt support

Each Sigma-Delta ADC has an independent interrupt vector. The interrupt vectors for the Sigma-Delta ADC are defined in the interrupt chapter of this document.

Each ADC combines all interrupt sources into a single interrupt request output signal. Each interrupt source is cleared with an independent register flag bit.

34.1.10.3 DMA support

Each Sigma-Delta ADC is able to generate an independent DMA request for a result FIFO full request. The DMA channel assignments for the Sigma-Delta ADCs are defined in the DMA chapter of this document.

34.1.11 Sigma-Delta interface to ETPU module

Interfacing of SDADC to the ETPU module is discussed in the following sections.

34.1.11.1 Valid data available output to ETPU

The Sigma-Delta digital interface outputs a logic level 1 pulse on the end-of-conversion signal to the ETPU with each update to the data register. The actual operation is such that the valid data available signal is asserted on the rising edge of the SD_CLK. This aligns to the actual end-of-conversion. The converted value is written to the data register on the next falling edge of clock, introducing a half SD_CLK delay until the value can be read.

The mapping of Sigma-Delta valid data available signals to ETPU inputs is controlled in the SIUL2 MSCR registers, and the detailed mapping is given in the input muxing table of the device I/O definition spreadsheet for the device.

34.1.11.2 DMA/interrupt gating input from ETPU

The Sigma-Delta ADC digital interface has an input signal that is used to disable interrupt and DMA requests for conversion results. The input gating signal is connected to ETPU channel outputs. The digital interface has a register bit for enabling and disabling the interrupt/DMA gating feature.

When the gating signal from the ETPU is asserted, no interrupt or DMA request is generated. The polarity of the gating signal is such that when the signal is high, interrupt and DMA requests are passed. When the gating signal is low, interrupt and DMA requests are blocked.

When the ETPU gating signal is asserted, the Sigma-Delta FIFO cannot overflow.

The gating signal for each ADC is driven by several ETPU output channels. The mapping of ETPU outputs to Sigma-Delta DMA/IRQ gating inputs is controlled in the SIUL2 MSCR registers, and the detailed mapping is given in the input muxing table of the device I/O definition spreadsheet for the device.

34.1.11.3 Conversion limit watchdog output to ETPU

Each Sigma-Delta ADC provides an optional conversion limit check for upper and lower bounds. The enable bit and the two 16-bit limit registers for the feature are included in the Sigma-Delta ADC digital interface. If a conversion result exceeds one of the limits, a flag bit is asserted in a status register. The flag bit is output to the ETPU for each ADC. The mapping of Sigma-Delta conversion limit watchdog outputs to ETPU inputs is controlled in the SIUL2 MSCR registers, and the detailed mapping is given in the input muxing table of the I/O definition spreadsheet for the device.

34.1.12 Sigma-Delta interface to eMIOS module

Interfacing of SDADC to the eMIOS module is discussed in the following sections.

34.1.12.1 DMA/interrupt gating input from eMIOS

The Sigma-Delta ADC digital interface has an input signal that is used to disable interrupt and DMA requests for conversion results. The input gating signal is connected to eMIOS channel outputs. The digital interface has a register bit for enabling and disabling the interrupt/DMA gating feature.

When the gating signal from the eMIOS is asserted, no interrupt or DMA request is generated. The polarity of the gating signal is such that when the signal is high, interrupt and DMA requests are passed. When the gating signal is low, interrupt and DMA requests are blocked.

When the eMIOS gating signal is asserted, the Sigma-Delta FIFO cannot overflow.

The gating signal for each ADC is driven by several eMIOS output channels. The mapping of eMIOS outputs to Sigma-Delta DMA/IRQ gating inputs is controlled in the SIUL2 MSCR registers, and the detailed mapping is given in the input muxing table of the device I/O definition spreadsheet for the device.

34.1.13 Sigma-Delta interface to decimation filter modules

There is no dedicated hardware interface from the Sigma Delta ADCs to the Decimation Filter blocks on the device. Transfer of ADC conversion results to the decimation filters for application processing must be done by software or DMA on the MPC5746R.

NOTE

The Decimation Filter blocks on the MPC5746R should not be confused with the internal decimation filter of the Sigma Delta ADC.

34.1.14 Sigma-Delta interface to reaction channels

There is no dedicated hardware interface from the Sigma Delta ADCs to the reaction channels on the MPC5746R.

34.1.15 Sigma-Delta Interface to the Cross Triggering Unit (CTU)

There is no dedicated hardware interface from the Sigma Delta ADCs to the BCTU on the MPC5746R.

34.2 Sigma Delta Analog-to-Digital Converter (SDADC) digital interface reference

34.2.1 Introduction

The Sigma-Delta Analog-to-Digital Converter (SDADC) digital interface block controls the on-chip SDADC and holds control and status registers accessible for application. It provides accurate conversion data for a wide range of applications.

34.2.2 Overview

The SDADC digital interface block provides all control information for operating mode selection (single-ended or differential), analog input gain, decimation rate, high-pass filter enable, etc. It also generates the bias controls for common mode voltage selection, to bias the positive and negative terminals to different voltage levels (for example, half scale bias, reference ground, reference supply) for diagnostics. It generates the analog multiplexer control signals for the selected channels. To assist synchronous operation across multiple ADC blocks, hardware trigger support is implemented to start data conversion.

34.2.3 Features

The features of the SDADC are as follows:

- 16-bit data resolution output
- Single-ended input or differential input mode of operation
- Programmable wraparound mechanism for both modes of operation
    - Configurable trigger sources: hardware or software
    - Configurable initial entry and wraparound values for the loop
- Configurable biasing for negative input terminal in single-ended mode
- Gain and offset calibration support using fixed bias for input terminals
- Programmable decimation rate
- Programmable gain for analog inputs
- Optional high-pass filter for pure AC application
- Hardware trigger support for synchronous operation of multiple SDADC blocks
- Trigger event output generation by software
- Programmable FIFO structure for storing up to 16 datawords of 16-bit converted data
- Interrupt/DMA request generation based on various conditions:

  - Data buffer above threshold
  - Data buffer overrun
  - Watchdog (WDG) crossover event
- Programmable threshold range watchdog support
- Low consumption mode
- Optional conversion process halt mechanism on SoC debug request

34.2.4 Modes of operation

This section describes the different operation modes of the SDADC. When exiting reset, the mode of operation is determined by the Module Configuration Register (MCR).

34.2.4.1 Differential input mode

This mode is entered by negating the MCR[MODE] bit. In this mode, a pair of analog inputs are connected to positive and negative terminals of ADC modulator. In order to support gain and offset calibration for diagnostics, it is possible in this mode to select fixed bias voltages to both input terminals.

34.2.4.2 Single-ended input mode

This mode is entered by asserting the MCR[MODE] bit. In this mode, the negative input terminal is biased with a fixed voltage based on selection information in MCR, and the other analog input is connected to the selected external analog channel.

34.2.5 External signal description

The following table shows the list of signals driven by external pins. See the chip-specific details about SDADC signals and package pinouts for information about signal muxing and availability external to the chip.

Table 34-2. Signal properties

Name	Function	I/O	Reset	Pull-up
AN[0:7]	Single-ended analog inputs or Differential pairs of analog inputs	I	—	—
VREFP	Voltage reference for SDADC	I	—	—
VREFN	Ground reference for SDADC	I	—	—
AVDD	Analog voltage supply for SDADC	I	—	—

Table continues on the next page...

Table 34-2. Signal properties (continued)

Name	Function	I/O	Reset	Pull-up
AVSS	Analog ground supply for SDADC	I	—	—
DVDD	Digital voltage supply for SDADC	I	—	—
DVSS	Digital ground supply for SDADC	I	—	—

34.2.6 Memory map and register definition

This section provides the memory map for the SDADC block.

SDADC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (SDADC_MCR)	32	R/W	0000_0000h	34.26.1/ 944
4	Channel Selection Register (SDADC_CSR)	32	R/W	0000_0000h	34.26.2/ 947
8	Reset Key Register (SDADC_RKR)	32	R/W	0000_A50Fh	34.26.3/ 949
C	Status Flag Register (SDADC_SFR)	32	R/W	0000_0100h	34.26.4/ 950
10	Request Select and Enable Register (SDADC_RSER)	32	R/W	0000_0000h	34.26.5/ 952
14	Output Settling Delay Register (SDADC_OSDR)	32	R/W	0000_0000h	34.26.6/ 954
18	FIFO Control Register (SDADC_FCR)	32	R/W	0000_0006h	34.26.7/ 955
1C	Software Trigger Key Register (SDADC_STKR)	32	R/W	0000_0000h	34.26.8/ 956
20	Converted Data Register (SDADC_CDR)	32	R	0000_0000h	34.26.9/ 956
24	WDG Threshold Register (SDADC_WTHHLR)	32	R/W	0000_0000h	34.26.10/ 957

34.26.1 Module Configuration Register (SDADC_MCR)

The Module Configuration register (MCR) consists of different control bits to select the operating modes and various configuration settings which define the behavior of the SDADC block.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0			PDR					0	PGAN			Reserved	Reserved	0	HPFEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDGEN	TRIGEDSEL		TRIGEN	0		TRIGSEL		FRZ	0		VCOMSEL	WRMODE	GECEN	MODE	EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDADC_MCR field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 PDR	Programmable Decimation Rate This field selects the over-sampling ratio to be applied to support different passbands with a fixed input sampling clock. The output data rate f_d is equal to f_s / (2xOSR), where fs is the input sampling clock frequency. 00000 OSR = 24 00001 OSR = 28 00010 OSR = 32 00011 OSR = 36 00100 OSR = 40 00101 OSR = 44 00110 OSR = 48 00111 OSR = 56 01000 OSR = 64 01001 OSR = 72 01010 OSR = 75 01011 OSR = 80 01100 OSR = 88 01101 OSR = 96

Table continues on the next page...

SDADC_MCR field descriptions (continued)

Field	Description
	01110 OSR = 112 01111 OSR = 128 10000 OSR = 144 10001 OSR = 160 10010 OSR = 176 10011 OSR = 192 10100 OSR = 224 10101 OSR = 256 10110 - 11111 Reserved
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–11 PGAN	Programmable Gain This field selects the gain to be applied to the analog input stage of the SDADC. The effective analog input becomes the input voltage level multiplied by the gain factor. **NOTE:** When SDADC_MCR[PGAN]=111b (programmable gain is 16), the least significant bit of the conversion data in SDADC_CDR[CDATA] is always 0. 000 Gain = 1 001 Gain = 2 010 Gain = 4 011 Gain = 8 100 Reserved 101 Reserved 110 Reserved 111 Gain = 16
12 Reserved	This field is reserved. This field is reserved and always has the value 0. Only the reset value shall be written to this register field.
13 Reserved	This field is reserved. This field is reserved and always has the value 0. Only the reset value shall be written to this register field.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 HPFEN	High Pass Filter Enable 0 High-pass (DC removal) filter is disabled 1 High-pass (DC removal) filter is enabled
16 WDGEN	Watchdog Enable This bit enables the WDG monitor. 0 WDG is disabled 1 WDG is enabled
17–18 TRIGEDSEL	Trigger Edge Selection 00 Falling edge of trigger input is selected 01 Rising edge of trigger input is selected 10 Both edges of trigger input are selected 11 Both edges of trigger input are selected

Table continues on the next page...

SDADC_MCR field descriptions (continued)

Field	Description
19 TRIGEN	Trigger Enable This field enables the hardware trigger input to initiate a fresh conversion. Upon receiving a trigger event, SDADC reset input is asserted and deasserted synchronously with respect to the peripheral clock. The reset pulse width is fixed to four peripheral clock cycles. The trigger event is ignored if the SDADC internal modulator is not enabled (MCR[EN] = 0). 0 Trigger input is disabled 1 Trigger input is enabled
20–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 TRIGSEL	Trigger Input Selection This field selects which input will be used for hardware triggered conversions. 00 SDADC_0 output trigger is selected. 01 >SDADC_1 output trigger is selected. 10 SDADC_2 output trigger is selected. 11 SDADC START_TRIGGER MSCR is selected.
24 FRZ	Freeze This field enables stopping the SDADC conversions at the end of the current channel conversion when the chip enters debug mode. When the chip enters debug mode, further conversions by the SDADC analog block will be stopped, and converted data output from the SDADC analog block are ignored. The ongoing channel conversion will be completed and the converted data is stored. When the chip exits debug mode, SDADC resumes the job that was left before halting the conversions. 0 Conversions are not stopped 1 Conversions are stopped
25–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 VCOMSEL	Common Voltage Bias Selection This field selects the common voltage bias for the negative input terminal of the SDADC during single-ended mode (MODE = 1). 0 Negative input terminal is biased with VREFN 1 Negative input terminal is biased with VREFP/2 (half-scale bias)
28 WRMODE	Wrap-Around Mode This bit selects the wraparound mechanism for conversion of programmed sequence of channels. 0 Wraparound mechanism disabled (in this case the default software control mechanism will be enabled) 1 Wraparound mechanism enabled
29 GECEN	Accurate Gain Error Mode Enable This bit selects the accurate gain error mode, and must be set during gain error calibration as well as during conversions requiring accurate gain error mode. It can be kept '0' if gain error calibration mode is not needed. 0 Gain error calibration mode disabled 1 Gain error calibration mode enabled

Table continues on the next page...

SDADC_MCR field descriptions (continued)

Field	Description
30 MODE	Mode selection 0 Differential input mode selected 1 Single-ended input mode selected
31 EN	Enable for SDADC block 0 SDADC internal modulator placed in low consumption mode 1 SDADC internal modulator enabled

34.26.2 Channel Selection Register (SDADC_CSR)

The Channel Selection Register (CSR) selects analog inputs AN[0:7] to be connected to positive and negative terminals of the SDADC and controls the biasing of analog inputs to half-scale bias through 100 kOhm resistance.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								BIASEN							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					ANCHSEL_WRAP			0					ANCHSEL		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDADC_CSR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 BIASEN	Bias enable If BIASEN[i] = 1, analog input AN[i] (could be single-ended input or differential analog input) is connected to half-scale bias through a 100 kOhm resistor.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–23 ANCHSEL_ WRAP	Analog Channel Selection Wraparound Value (ANCHSEL_WRAP) When in wraparound mode, this field indicates the maximum value of the wraparound counter, after which it will wrap around back to initial value 000. This value should be programmed before enabling wraparound mechanism (MCR[WRMODE] = 1).
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SDADC_CSR field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>29–31
ANCHSEL</td><td>
Analog Channel Selection(ANCHSEL)

Based on single-ended or differential mode of operation (MCR[MODE]), common mode voltage selection (MCR[VCOMSEL]), this field defines the connectivity of analog inputs to either positive or negative polarity terminals of the SDADC according to the following table.

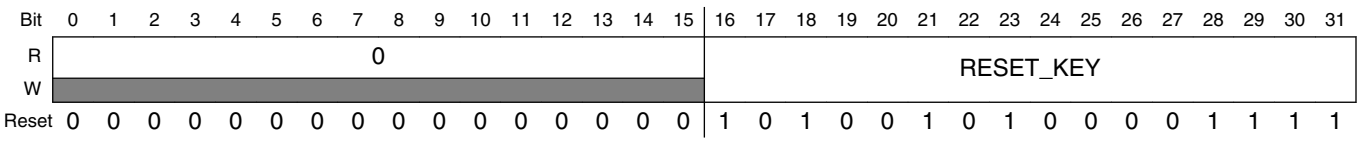
Wraparound Mode:
• This field also indicates the initial entry value for the first loop of the wraparound sequence from where the wraparound counter will start incrementing when wraparound mode is entered (when wraparound mechanism enabled (MCR[WRMODE] = 1)).
• If this field is written any time during wraparound mode (when MCR[WRMODE] = 1), the present value of the wraparound counter will be updated with the value of ANCHSEL - the next trigger will act on the updated channel, and the next increment will be on the updated channel. Here, the value of ANCHSEL chosen should be < ANCHSEL_WRAP; otherwise it will be ignored by hardware.
• Wraparound will always be from initial counter value 000.

<table>
<tr><th>MODE</th><th>VCOMSEL</th><th>ANCHSEL</th><th>INP (positive terminal)</th><th>INM (negative terminal)</th></tr>
<tr><td rowspan="16">1</td><td rowspan="8">0</td><td>000</td><td>AN[0]</td><td>VREFN</td></tr>
<tr><td>001</td><td>AN[1]</td><td>VREFN</td></tr>
<tr><td>010</td><td>AN[2]</td><td>VREFN</td></tr>
<tr><td>011</td><td>AN[3]</td><td>VREFN</td></tr>
<tr><td>100</td><td>AN[4]</td><td>VREFN</td></tr>
<tr><td>101</td><td>AN[5]</td><td>VREFN</td></tr>
<tr><td>110</td><td>AN[6]</td><td>VREFN</td></tr>
<tr><td>111</td><td>AN[7]</td><td>VREFN</td></tr>
<tr><td rowspan="8">1</td><td>000</td><td>AN[0]</td><td>VREFP/2</td></tr>
<tr><td>001</td><td>AN[1]</td><td>VREFP/2</td></tr>
<tr><td>010</td><td>AN[2]</td><td>VREFP/2</td></tr>
<tr><td>011</td><td>AN[3]</td><td>VREFP/2</td></tr>
<tr><td>100</td><td>AN[4]</td><td>VREFP/2</td></tr>
<tr><td>101</td><td>AN[5]</td><td>VREFP/2</td></tr>
<tr><td>110</td><td>AN[6]</td><td>VREFP/2</td></tr>
<tr><td>111</td><td>AN[7]</td><td>VREFP/2</td></tr>
<tr><td rowspan="8">0</td><td rowspan="8">0/1</td><td>000</td><td>AN[0]</td><td>AN[1]</td></tr>
<tr><td>001</td><td>AN[2]</td><td>AN[3]</td></tr>
<tr><td>010</td><td>AN[4]</td><td>AN[5]</td></tr>
<tr><td>011</td><td>AN[6]</td><td>AN[7]</td></tr>
<tr><td>100</td><td>VREFN</td><td>VREFN</td></tr>
<tr><td>101</td><td>VREFP/2</td><td>VREFP/2</td></tr>
<tr><td>110</td><td>VREFP</td><td>VREFN</td></tr>
<tr><td>111</td><td>VREFN</td><td>VREFP</td></tr>
</table>
</td></tr>
</table>

34.26.3 Reset Key Register (SDADC_RKR)

The Reset Key Register (RKR) is the target for a predefined key write operation used to reset the SDADC to start a fresh conversion.

Address: 0h base + 8h offset = 8h



SDADC_RKR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 RESET_KEY	Reset Key This field, when written with 5AF0h, is used to generate the reset for the SDADC block and to reload the internal counterwith the start value programmed in OSDR. Any write access to this register which is different from the predefined key is ignored. Read access will always return the inverted key.

34.26.4 Status Flag Register (SDADC_SFR)

The Status Flag Register (SFR) contains status and flag bits. These bits are set by hardware and reflect the status of the SDADC and indicate the occurrence of events that may generate an interrupt or a DMA request. Software can clear some flags by writing '1' to those bits. Writing a '0' to a flag has no effect.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0													ANCHSEL_CNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							DFEF	0			WTHH	WTHL	CDVF	DFORF	DFFF
W												w1c	w1c		w1c	w1c
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

SDADC_SFR field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 ANCHSEL_CNT	Analog Channel Selection Counter(ANCHSEL_CNT) This field reflects the current value of the internal Analog Channel Selection Counter. It indicates the present number of the channel selected through the analog multiplexer.
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 DFEF	Data FIFO Empty Flag This field is asserted when there is no dataword in the FIFO.

Table continues on the next page...

SDADC_SFR field descriptions (continued)

Field	Description
	0 Data FIFO is not empty 1 Data FIFO is empty
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 WTHH	Watchdog Upper Threshold Cross Over Event This field is set when Converted Data crosses Upper Threshold of WDG. It can be cleared by software writing '1' to clear. **NOTE:** This bit is also cleared by an acknowledgement from the DMA module when all of the following conditions are met: • SDADC_RSER[WTHDIRE]=1 (When this field is set by the occurrence of a watchdog upper threshold cross over event, a request is generated) • SDADC_SFR[WTHDIRS]=1 (The request generated when this field is set is a DMA request) • SDADC_RSER[GDIGE]=0 OR, if SDADC_RSER[GDIGE]=1, the external gating signal is asserted 0 Watchdog Upper Threshold Cross Over Event did not occur 1 Watchdog Upper Threshold Cross Over Event occurred
28 WTHL	Watchdog Lower Threshold Cross Over Event This field is set when Converted Data crosses Lower Threshold of WDG. It can be cleared by software writing '1' to clear. **NOTE:** This bit is also cleared by an acknowledgement from the DMA module when all of the following conditions are met: • SDADC_RSER[WTHDIRE]=1 (When this field is set by the occurrence of a watchdog lower threshold cross over event, a request is generated) • SDADC_SFR[WTHDIRS]=1 (The request generated when this field is set is a DMA request) • SDADC_RSER[GDIGE]=0 OR, if SDADC_RSER[GDIGE]=1, the external gating signal is asserted 0 Watchdog Lower Threshold Cross Over Event did not occur 1 Watchdog Lower Threshold Cross Over Event occurred
29 CDVF	Converted Data Valid Flag This field gives the actual status of data coming from the SDADC block. The converted data is not valid or is corrupted whenever there is a change in configuration settings determined by MCR, channel selection programmed in CSR. Any write access to the registers MCR and CSR will reset this flag. This flag is set automatically after an internal timer counts a number of output clock fd clock cycles specified by the value of SDADC_OSDR[OSD]. 0 Data output from SDADC is not valid 1 Data output from SDADC is valid
30 DFORF	Data FIFO Overrun Flag This field indicates that software or DMA failed to prevent FIFO from overflowing with converted datawords. Software can clear this bit by writing '1' to it after reading the FIFO contents through CDR. This flag remains set if software or DMA failed to clear the condition that caused this flag setting. If DFORF is set, further datawords will not be received into FIFO even if sufficient room exists. 0 No overrun has occurred since the last time the flag was cleared 1 Overrun has occurred or the DFORF has not been cleared since the last overrun occurred

Table continues on the next page...

SDADC_SFR field descriptions (continued)

Field	Description
31 DFFF	Data FIFO Full Flag This bit is set when the number of converted datawords in the FIFO is equal to or more than the number indicated by FCR[FTHLD]. Software can clear this bit by writing '1' to it. This bit is also cleared by an acknowledgement from the DMA controller when DMA request generation is selected. This bit remains set if software or DMA failed to clear the condition that caused this flag setting. Even if DFFF is set, datawords will continue to be received until an overrun condition occurs. Datawords will not be received if the converted data is not valid (CDVF = 0). **NOTE:** Whenever MCR[EN] is cleared in order to stop the SDADC or change the channel configuration, it is necessary to also clear the RSER[DFFDIRE] bit alongwith. Similarly if DMA or software is too slow in not reading the FIFO datawords resulting in DFFF condition it is advisable to stop the SDADC by clearing the MCR[EN] followed by clearing of RSER[DFFDIRE] bit. Both these action will ensure safe operation. 0 The number of datawords in FIFO is less than the number indicated by FCR[FTHLD] 1 The number of datawords in FIFO is equal to or greater than the number indicated by FCR[FTHLD] at some point in time since this flag was last cleared

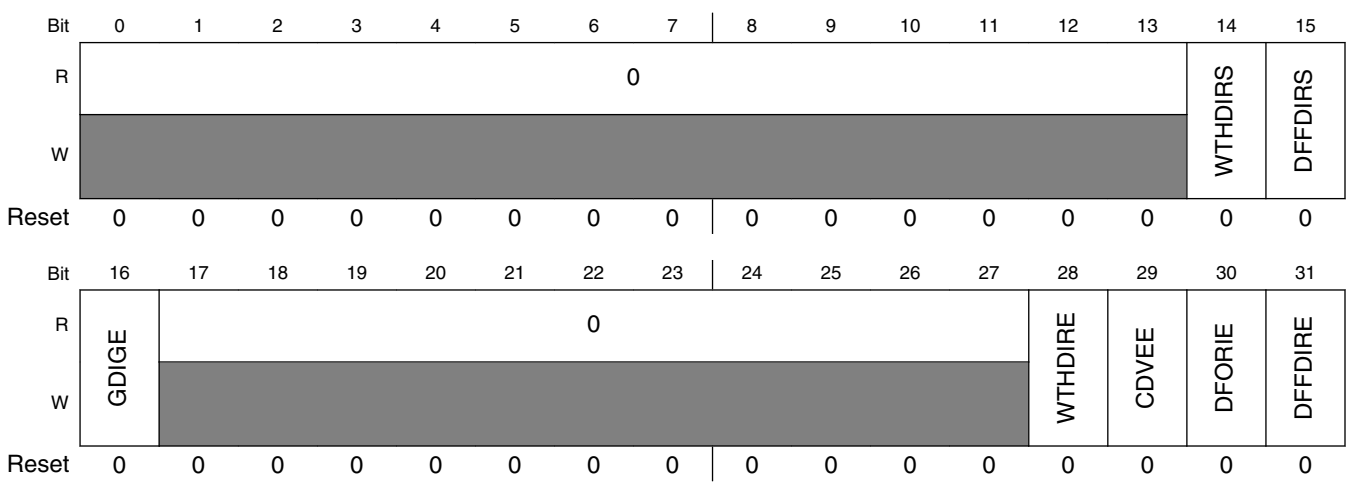
34.26.5 Request Select and Enable Register (SDADC_RSER)

The Request Select and Enable Register (RSER) serves two purposes:

- Enables the flag bits in SFR to generate DMA requests or interrupt requests.
- Selects the type of request to generate.

Refer to the field descriptions for the type of requests that are supported.

Address: 0h base + 10h offset = 10h



SDADC_RSER field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 WTHDIRS	WDG Threshold Cross Over Event DMA/Interrupt Request Select When SFR[WTHH/WTHL] and RSER[WTHDIRE] fields are set, this field selects between generating a DMA request or an interrupt request. 0 Interrupt request is selected 1 DMA request is selected
15 DFFDIRS	Data FIFO Full DMA/Interrupt Request Select When the SFR[DFFF] and SFR[DFFDIRE] fields are set, this field selects between generating a DMA request or an interrupt request. 0 Interrupt request is selected 1 DMA request is selected
16 GDIGE	Global DMA/Interrupt Gating Enable This field determines whether all module interrupts and DMA requests need to be gated by an external gating signal or not. 0 No impact of external gating signal on module DMA/interrupt requests 1 All module DMA/interrupt requests are qualified only when external gating signal is asserted
17–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 WTHDIRE	WDG Threshold Cross Over Event DMA/Interrupt Request Enable This field enables the SFR[WTHH/WTHL] field to generate a request. The WTHDIRS field determines which is selected, a DMA request or an interrupt request. The final DMA/interrupt request also depends on the gating signal if enabled by the GDIGE field. 0 Interrupt/DMA request is disabled on WDG Threshold Cross Over Event 1 Interrupt/DMA request is enabled on WDG Threshold Cross Over Event
29 CDVEE	Conversion Data Valid Event Enable The status of SFR[CDVF] gated with RSER[CDVEE] is passed on as an event on “conversion_flag" output port, which can be used at SoC level for indication of valid data conversion window. Once enabled, the event output will remain asserted as long as valid data conversions are ongoing. 0 Event output disabled 1 Event output assertion/deassertion based on the SFR[CDVF] field
30 DFORIE	Data FIFO Overrun Interrupt Enable This bit enables the SFR[DFORF] field to generate an interrupt request. The final interrupt request also depends on the gating signal if enabled by the GDIGE field. 0 Interrupt request is disabled when data FIFO overrun condition occurs 1 Interrupt request is enabled when data FIFO overrun condition occurs
31 DFFDIRE	Data FIFO Full DMA/Interrupt Request Enable This field enables the SFR[DFFF] field to generate a request. The DFFDIRS field determines which is selected, a DMA request or an interrupt request. The final DMA/interrupt request also depends on the gating signal if enabled by the GDIGE field.

Table continues on the next page...

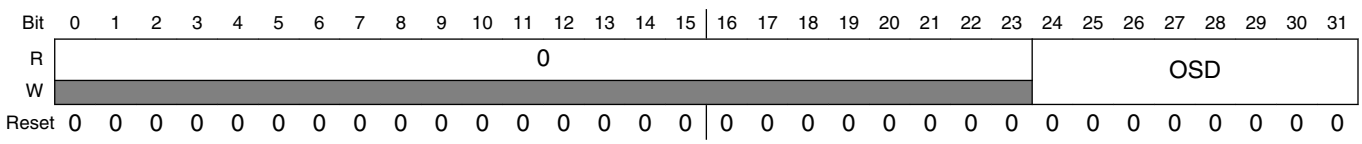
SDADC_RSER field descriptions (continued)

Field	Description
	0 Interrupt/DMA request is disabled when data FIFO full condition occurs 1 Interrupt/DMA request is enabled when data FIFO full condition occurs

34.26.6 Output Settling Delay Register (SDADC_OSDR)

The Output Settling Delay Register (OSDR) provides a delay value to qualify the converted output data coming from the SDADC.

Address: 0h base + 14h offset = 14h



SDADC_OSDR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 OSD	Output Settling Delay This field defines the delay to qualify the conversion data stored in CDR. Whenever the SDADC block is reset in order to start the conversion from a fresh state, an internal timer is loaded with this start value. The counter counts down with output clock f_d until it reaches zero, and then it generates a flag which qualifies the converted data. The output clock fd is equal to f_s / (2xOSR), where f_s is the input sampling clock frequency. See the electrical specifications for the settling delays. **NOTE:** Refer to the $t_{settling}$ and $t_{LATENCY}$ parameters in the device datasheet for the minimum allowed output settling delay values.

34.26.7 FIFO Control Register (SDADC_FCR)

The FIFO Control Register (FCR) provides the ability to enable or disable the FIFO functionality, and to control the amount of available FIFO depth to be used.

NOTE

1. It is important to note that the FIFO depth is set at the factory and cannot be changed but the amount of available FIFO space actually used can be set by specifying the value in the FTHLD field. The FSIZE field indicates the amount of FIFO space available. By setting the value of FTHLD to be less than the FIFO depth indicated by FSIZE, the FIFO is considered to be full when the number of data words in the FIFO is equal or greater than FTHLD.
2. This register may be read at any time, but should only be written when SDADC is disabled (MCR[EN] = 0).

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				FTHLD				0					FSIZE		FE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

SDADC_FCR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 FTHLD	FIFO Threshold When the number of datawords in the data FIFO is greater than the value in FTHLD field, the FIFO full event is flagged. An interrupt or a DMA request will be generated as determined by DFFDIRE and DFFDIRS fields of RSER. For proper operation, the value in the FTHLD field must be set to be less than or equal to the FIFO size as indicated by FSIZE.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 FSIZE	FIFO Size

Table continues on the next page...

SDADC_FCR field descriptions (continued)

Field	Description
	The maximum number of converted datawords that can be stored in the data FIFO before an overrun occurs. This field is read-only. The reset value of this field depends on the implementation parameter to define the FIFO size. 00 FIFO depth = 1 dataword 01 FIFO depth = 4 datawords 10 FIFO depth = 8 datawords 11 FIFO depth = 16 datawords
31 FE	FIFO Enable When this field is set the built-in FIFO structure is enabled. The size of the FIFO structure is indicated by the FSIZE field. If this field is not set then the FIFO depth is set to one dataword regardless of the value in the FSIZE field. 0 Data FIFO is not enabled for multiple datawords; FIFO depth is one dataword 1 Data FIFO is enabled; FIFO depth is indicated by FSIZE

34.26.8 Software Trigger Key Register (SDADC_STKR)

The Software Trigger Key Register (STKR) allows generating the trigger event output by a software write operation.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																ST_KEY															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDADC_STKR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 ST_KEY	Software Trigger Key This bitfield, when written with FFFFh, is used to generate a trigger event output which can be used to trigger conversions of multiple SDADC blocks synchronously depending on the chip's implementation. Any write access to this register which is different from the predefined key is ignored. Read access will always return 0000h.

34.26.9 Converted Data Register (SDADC_CDR)

The Converted Data Register (CDR) stores the converted data in the data FIFO.

The conversion result in the CDATA word is a 16-bit, signed value.

NOTE

When SDADC_MCR[PGAN]=111b (programmable gain is 16), the least significant bit of the conversion data in SDADC_CDR[CDATA] is always 0.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																CDATA															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDADC_CDR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 CDATA	Converted Data Register (CDATA[13:0]) The converted datawords can be read from FIFO by reading this register. The data width is 16 bits which has actual 16-bit data (bits 15 down to 0) coming from ADC.

34.26.10 WDG Threshold Register (SDADC_WTHHLR)

This is the Converted Data Range WDG Threshold register.

NOTE

A single WDG monitor will check for lower and upper threshold crossover events.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	THRH																THRL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SDADC_WTHHLR field descriptions

Field	Description
0–15 THRH	WDG Upper Threshold Value This indicates the upper threshold for the converted data on any channel, above which threshold a crossover event would be flagged through SFR[WTHH].
16–31 THRL	WDG Lower Threshold Value This indicates the lower threshold for the converted data on any channel, below which threshold cross over event would be flagged through SFR[WTHL].

34.2.7 Functional description

The SDADC has the following distinct available modes determined by fields in MCR:

- Differential input mode
- Single-ended input mode
- Low consumption mode

34.2.7.1 Differential input mode

Differential input mode is entered when the SDADC_MCR[MODE] field is set to '0' and the SDADC_MCR[EN] field is set to '1'. In differential input mode, a pair of analog inputs among AN[0:7] is connected to the input terminals of the SDADC modulator block. The module supports four differential pairs: AN[0,1], AN[2,3], AN[4,5], and AN[6,7].

The modulator block has two input terminals, positive (INP) and negative (INM). The selection of input pair depends on the SDADC_CSR[ANCHSEL] field. The connectivity of each analog input in a given differential pair, whether it is connected to positive or negative terminal, is determined according to the Analog input AN[0:7] selection table in the Channel Selection Register description.

The differential input mode can also be used to perform gain and offset calibration, where the input terminals are biased with fixed voltages coming from the bias block.

34.2.7.2 Single-ended input mode

Single-ended input mode is entered by setting the SDADC_MCR[MODE] and SDADC_MCR[EN] fields to '1'. In this mode, one of the analog inputs among AN[0:7] is selected via the SDADC_CSR[ANCHSEL] field and is connected to the positive terminal. The negative input terminal is biased with a fixed internal voltage. There are two internal bias voltages available: reference ground (VREFN) and half-scale bias (VREFP/2). The selection of this common mode voltage is controlled vua the SDADC_MCR[VCOMSEL] field.

34.2.7.3 Low consumption mode

This is the default mode after exiting from reset. This mode is exited by asserting the MCR[EN] field to enable the SDADC block. When data conversion is not required, the SDADC internal modulator can be put in low consumption mode to reduce power consumption.

34.2.7.4 Analog input multiplexing and bias support

The complete selection and connectivity of analog inputs AN[0:7] is based on the ANCHSEL field of CSR and the MODE and VCOMSEL bits of MCR, according to the Analog input AN[0:7] selection table in the Channel Selection Register description. Each analog input can be biased to half scale (AVDD/2) via on-chip 100 kΩ resistance. This allows simple AC coupling to the external analog inputs through a capacitor. This biasing is applicable irrespective of whether analog channel is selected through multiplexer or not.

34.2.7.5 Programmable gain and decimation rate

All analog inputs can be configured to have a selectable input gain as defined in the PGAN field description in MCR. This means the input signal is sampled and the result is amplified by the factor determined by PGAN before providing the same to modulator.

The SDADC module is provided with a fixed input sampling clock f_s. The input signal is grossly oversampled by the modulator in order to reduce quantization noise. To support different passbands with a fixed sampling clock, a programmable decimation rate is implemented. The process of decimation, to eliminate redundant data at the output while retaining the necessary information, is controlled by the field PDR in MCR.

34.2.7.6 High-pass filter support

For pure AC applications, it is useful to remove any DC component in the input signal. An optional high-pass filter is implemented which can be enabled by asserting the MCR[HPFEN] field. The –3 dB frequency of the filter is fixed (not programmable) and is specified in the device data sheet. This high-pass filter is implemented in the decimation filter logic of the SDADC.

34.2.7.7 Data conversion

The SDADC operates in a continuous conversion mode as soon as it is enabled. There is an inherent settling time required after SDADC startup before conversion data is considered valid.

NOTE

During that period, the Converted Data Valid Flag bit (SDADC_SFR[CDVF]) reads '0'.

Settling time duration depends on the latency and output settling time specifications. To account for the settling time, an internal timer is implemented that counts down from a start value determined by the value of the SDADC_OSDR[OSD] field.

The internal countdown timer is triggered by an SDADC reset, i.e., by writing the value 5AF0h to the SDADC_RKR[RESET_KEY] field.

NOTE

An SDADC reset is required any time there is a change to the SDADC configuration (via changes to the SDADC_MCR register settings), or a change to analog channel selection (via SDADC_CSR register settings).

When the timer reaches '0', the SDADC_SFR[CDVF] field is asserted and converted data is loaded into data FIFO on every rising edge of the output clock (f_d).

Any write access to the SDADC_MCR or SDADC_CSR register causes the SDADC_SFR[CDVF] flag to be cleared, indicating that data coming from SDADC is not valid. If the SDADC_SFR[CDVF] flag is '0', the following flags are blocked from becoming asserted:

- Data FIFO Full Flag (SDADC_SFR[DFFF])
- Data FIFO Overrun Flag (SDADC_SFR[DFORF])

If the SDADC_SFR[DFFF] bit is already set, the data FIFO is already full before SDADC_SFR[CDVF] deassertion.

Wraparound Control Mechanism: Here a wraparound logic includes all analog channels of the input mutiplexer from Ain("000") to Ain(CSR[ANCHSEL_WRAP]) in both Single ended and Differential ended mode in wraparound sequence triggered by hardware or software source:

- MCR[WRMODE] bit would enable or disable the wraparound mechanism. Clearing the WRMODE bit automatically switches FSM back to Software control mode.
- Every valid trigger advances the wraparound counter to next channel to be converted.
- Wraparound Counter: An internal wraparound counter which points to analog channel to be converted takes the initial entry value for the first loop of execution

from CSR[ANCHSEL] while entering wraparound mode. From next loop onwards the counter wraparounds to default initial value "000". Maximum value of the counter (wraparound value) should be programmed in CSR[ANCHSEL_WRAP] before entering wraparound mode. The definition of ANCHSEL and ANCHSEL_WRAP is provided in the Analog input AN[0:7] selection table in the Channel Selection Register description.

- Upon entering or during wraparound mode user should take care to program ANCHSEL value less than ANCHSEL_WRAP value; otherwise hardware would reject the new ANCHSEL value.
- Possible trigger sources:
  - SOURCE1 (Software Bit): writing to STKR.ST_KEY at any time during the wraparound mode (irrespective of MCR[TRIGEN or TRIGSEL] setting) generates a trigger pulse. Each such trigger initiates a fresh conversion and advances the wraparound counter by one.

NOTE

Care should be taken while programming the TRIGEN/ TRIGSEL of each SDADC instance that no unintended simultaneous SW triggering of multiple instances happen.

  - SOURCE2 (Hardware Trigger): Here external trigger output enabled by MCR[TRIGEN or TRIGEDSEL or TRIGSEL] before entering wraparound mode for selective or all SDADC instances provides trigger. Each such valid trigger initiates a fresh conversion and advances the wraparound counter by one. One of the hw_trig_in ports can be used to connect the external trigger output.
- Next channel number to be converted can be updated by software even in the middle of wraparound sequence execution. This can be done by rewriting CSR[ANCHSEL]. After CSR[ANCHSEL] is rewritten, the next trigger takes updated value and the wraparound counter would increment from the updated value.
- Current wraparound counter value can be read from the SFR[ANCHSEL_CNT] bitfield.

34.2.7.8 Hardware triggering

Software Control Mechanism: SDADC conversion can be triggered by various hardware events coming from either external pins or internal timers. The field TRIGSEL of MCR selects which hardware event is to be used for triggering conversions. This trigger event, which is synchronous to the peripheral clock, is enabled by asserting MCR[TRIGEN]. As soon as a trigger event is detected, reset input of SDADC block is asserted synchronously

for four cycles. This will ensure that SDADC starts from a fresh state when reacting to the input trigger event. The SDADC internal modulator must be enabled before enabling the trigger event — otherwise the trigger event will be ignored.

If the same trigger event is provided to multiple SDADC digital interface modules and the respective enable controls (TRIGEN bit MCR) are asserted, synchronous operation of multiple SDADC blocks can be achieved with output results being updated at the same time. Once the reset input is deasserted, the first valid conversion output is stored in data FIFO after a configurable delay determined by OSDR[OSD].

It is also possible to generate a software trigger event output by a write access of STKR with a predefined key. This event output can be used as a trigger to achieve synchronous software start of multiple SDADC blocks.

In case of a hardware trigger, the skew/jitter will be less than 100 ns. It will be more than one ADC clock cycle due to synchronization. In case of a software trigger, all SDADCs start synchronous sampling.

Wraparound Control Mechanism: In wraparound control mode, hardware and software triggers are equally valid. The hardware trigger source should be selected before enabling the wraparound mode.

34.2.7.9 Interrupt/DMA request support

The SDADC has one condition that generates interrupts only and another that generates interrupts or DMA request.

The DFFF field is asserted when the data FIFO full condition is detected, indicating that the number of converted datawords stored in data FIFO is equal to or greater than the value indicated by threshold determined by FCR[FTHLD]. The interrupt or DMA request is generated only when RSER[DFFDIRE] is asserted. RSER[DFFDIRS] selects whether a DMA request or an interrupt request is generated.

The data FIFO overrun flag (DFORF) indicates the FIFO overflow condition. Further converted datawords cannot be received and data is lost. The interrupt request is generated only when RSER[DFORIE] is asserted.

Apart from above, a WDG threshold crossover event can also generate an interrupt or DMA request.

If RSER[GDIGE] is set, all module interrupt/DMA requests are qualified only when an external global gating signal is asserted.

34.2.7.10 Gain calibration support

To perform gain calibration, the following sequence needs to be applied.

1. Select differential mode of operation by writing MCR[MODE] to '0'.
2. Enable gain error calibration mode by writing MCR[GECEN] to '1'.
3. Configure the mux selection ANCHSEL field of CSR to '110'. This configuration will apply VREFP on positive terminal and VREFN on negative terminal.
4. Disable the bias on all input analog channels by writing ENBIAS field of CSR to 00h.
5. Disable the high-pass filter by deasserting MCR[HPFEN].
6. Generate a reset event by writing 5AF0h to RESET_KEY of RKR.
7. Read the digital output (CDR[CDATA]) stored in FIFO after the output settling time. Expected output if there is no gain error is 0111_1001_1001_1001b, corresponding to full positive scale after attenuation inserted by the internal filter (1*0.95). The measurement should be repeated to reduce contribution of noise during the calibration process. D_p is the average value of attenuated positive full scale given by D_p = AVERAGE(CDR[CDATA]).
8. Change the mux selection ANCHSEL field of CSR to '111'. This configuration will apply VREFN on positive terminal and VREFP on negative terminal.
9. Generate a reset event by writing 5AF0h to RESET_KEY of RKR.
10. Read the digital output (CDR[CDATA]) stored in FIFO after the output settling time. Expected output if there is no gain error is 1000_0110_0110_0110b, corresponding to full negative scale after attenuation inserted by the internal filter (-1*0.95). The measurement should be repeated to reduce contribution of noise during calibration process. D_n is the average value of attenuated negative full scale given by D_n = AVERAGE(CDR[CDATA]).
11. The SDADC calibrated gain can be calculated as:
    $Gain = (D_p - D_n) / 2^{16}$
12. The measured gain value can be used to nullify the gain errors in the digital output. For calibrated conversion, the data CDR[CDATA] provided by the SDADC should be normalized using the calculated gain: $CDATA_{norm}$ is given by $CDATA_{norm}$ = CDR[CDATA]/Gain.

Note

a. The gain error calibration should only be done with gain equal to '1' (in other words, PGAN = 000). Nevertheless, the calculated gain can be applied to normalized data also for conversion with GAIN>1.
b. MCR[GECEN] = 1 ensures the accurate gain error mode is enabled. This bit should be set both during calibration process and after calibration, for application

data conversion. Application data conversion must then be normalized using the measured gain.

c. The higher the number of full-scale conversion (D_p, D_n) done, the higher the rejection of noise during the calibration. The number of conversion is dependent on application noise. It is recommended to run at least 16 conversions when calculating an average value.

34.2.7.11 Offset calibration support

To perform offset calibration, the following sequence must be applied.

1. Select differential mode of operation by writing SDADC_MCR[MODE] to '0'.
2. Configure the SDADC_CSR[ANCHSEL] mux selection field to '100' or '101' as required.
   a. SDADC_CSR[ANCHSEL] ='100' in case of data conversion after calibration in "single ended mode with negative input=$V_{SS_HV_ADR_D}$"
   b. SDADC_CSR[ANCHSEL] ='101' in case of data conversion after calibration in "differential mode" and "single ended mode with negative input = ($V_{DD_HV_ADR_D} - V_{SS_HV_ADR_D}$) / 2"
3. Disable the bias on all input analog channels by writing ENBIAS field of CSR to 00h.
4. Disable the high pass filter by deasserting SDADC_MCR[HPFEN].
5. Generate a reset event by writing 5AF0h to RESET_KEY of RKR.
6. Read the digital output stored in FIFO after the output settling time.
7. The measured offset can be used to nullify the offset error in the digital output. Expected output is 00_0000_0000_0000b. The SDADC offset can be calculated as:
   Offset = Expected Output – Actual Output

Note

The offset must be calculated for each SDADC_MCR[PGAN] field setting since it is expected to vary with gain configuration of SDADC.

34.2.8 Initialization information

To initialize the SDADC registers for data conversion, the following sequence is required.

1. Enable the SDADC by asserting MCR[EN].

2. Configure MCR to select the required mode, polarity, common mode voltage, input gain, and decimation rate.
3. Enable high-pass filter if required.
4. Select the required analog channel for data conversion. It is possible to select the bias for each channel for AC coupling applications.
5. Configure OSD delay according to SDADC startup time or latency from reset exit.
6. Generate a reset event by writing 5AF0h to RESET_KEY of RKR. Otherwise, if data conversion need to be triggered by a hardware event, assert MCR[TRIGEN].
7. DFFF interrupt or DMA request will be generated after the data FIFO has reached threshold to indicate that data can be transferred.

34.2.8.1 Data conversion step

To acquire a data from SDADC, the following sequence is required:

1. Enable the SDADC by asserting MCR[EN].
2. Configure MCR to select the required mode, polarity, common mode voltage, input gain, and decimation rate.
3. Enable high-pass filter is required.
4. Select the required analog channel for data conversion. It is possible to select the bias for each channel for AC coupling applications.
5. Configure OSD delay according to SDADC required startup time or latency from reset exit.
6. Generate a reset event writing 5AF0h in RESET_KEY field of RKR register.
7. Wait till FIFO empty flag DFEF of SFR register is clear.
8. Read data by CDATA of CDR register.

Repeat steps from 6 to 8 for new acquisitions.

NOTE

The time elapsed between reset event and Read data by CDATA field of CDR register are:

- Reset event <- -> Data Valid flag(CDVS) = Output settling Time delay (OSDR has to be set at least to 16) (Refer to the $t_{settling}$ and $t_{LATENCY}$ parameters in the device datasheet for the minimum allowed output settling delay values.).
- Data Valid flag <- -> Data FIFO is not empty = 0.5*CLK_Out + 3*CLK_In
- Data FIFO is not empty <- -> read Data (safety point) = 2*CLK_In, where CLK_Out = CLK_In/(2*OSR)

Chapter 35
Analog-to-Digital Converter (ADC)

35.1 Analog-to-Digital Converter (ADC) chip-specific information

35.1.1 Introduction

The following sections contain the chip-specific information for the on-chip SAR ADC modules. See Analog-to-Digital Converter (ADC) reference for detailed information on the SAR ADC modules.

35.1.2 SAR ADC block diagram

Figure 35-1 gives the block diagram of the SAR ADC and the integration into the MPC5746R. The IP is comprised of two pieces: the analog converter and the digital wrapper.

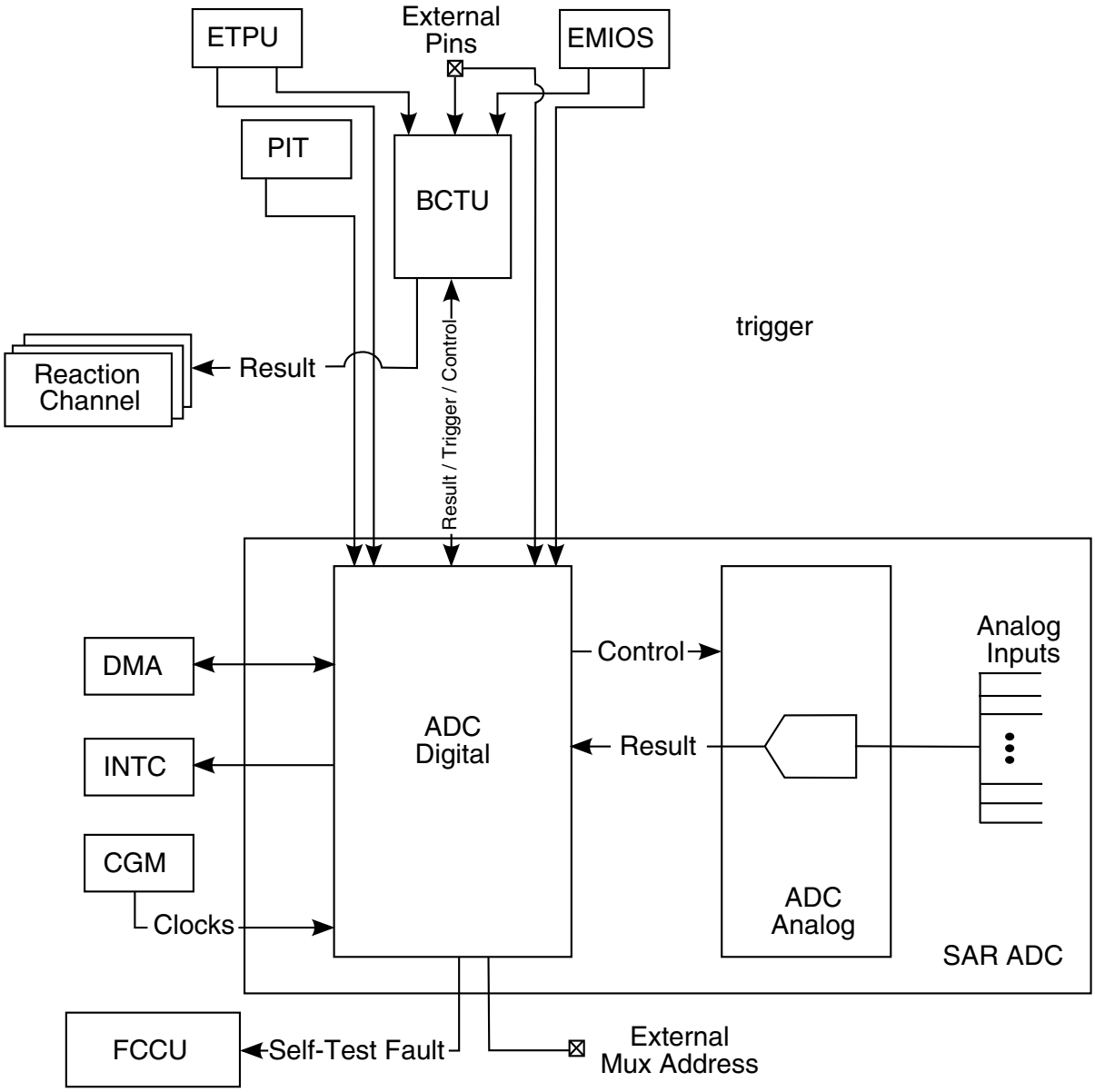


Figure 35-1. SAR ADC block diagram

35.1.3 SAR ADC external signal description

The SAR ADC external package pins are defined in the following sections.

NOTE

Please refer to the *MPC5746RMicrocontroller Data Sheet* for signal voltage specifications.

Table 35-1. SAR ADC external signal description

Signal Name	Signal Type	Description
PW[n],PX[n],PY[n]	Analog Input	Single-Ended Analog Input
VDD_HV_ADV_SAR	Analog Supply	SAR ADC Analog Power Supply
VSS_HV_ADV_SAR	Analog Ground	SAR ADC Analog Ground
VDD_HV_ADR_SAR	Analog Supply	SAR ADC Reference
VSS_HV_ADR_SAR	Analog Ground	SAR ADC Ground Reference

35.1.4 SARADC feature comparison

The following table lists the features for each of the 4 on-chip SAR ADC module.

Table 35-2. SAR ADC features

Feature	SARADC0	SARADC1	SARADC2	SARADC3
Resolution	12-bit	12-bit	12-bit	12-bit
No. of precision channels	32	32	32	21
No. of internal channels	7	6	4	4
Number of externally multiplexed channels	0	0	0	32
BCTU trigger mode support	yes	yes	yes	yes
BCTU control mode support	yes	yes	yes	yes
DMA support	yes	yes	yes	yes
Presampling support	yes	yes	yes	yes
No. of watchogs	4	4	4	4
BCTU asynchronous support	yes	yes	yes	yes
Self test support[1, 1]	yes	yes	yes	yes

1. ADC self test functions can only be initiated via the ADC registers. ADC self test cannot be initiated or controlled by the BCTU module.

35.1.5 SARADC channel map

The following table contains the channel map for each of the 4 on-chip SAR ADC module.

NOTE

In addition to the analog input pins in the device packages, there are internal voltage sources that must be monitored by the

SAR ADCs. These include the ADC temp sensor, bandgap reference, core and I/O supply voltages, LVD/HVD, and other internal monitors. Internal voltages mapped to SAR ADC channels are listed in the section of Table 35-3 labeled as **Internal Channels**.

Table 35-3. SAR ADC channel map

	Source of Inputs				
	Channel	ADC0	ADC1	ADC2	ADC3
Precision Channels	0	PZ[0]	PY[4]	PZ[0]	PY[3]
	1	PZ[1]	PY[5]	PZ[1]	PY[5]
	2	PZ[2]	PY[6]	PZ[2]	PY[6]
	3	PZ[3]	PY[7]	PZ[3]	PY[8]
	4	PZ[4]	PY[8]	PZ[4]	PY[9]
	5	PZ[5]	PY[9]	PZ[5]	PY[11]
	6	PZ[6]	PY[10]	PZ[6]	PY[12]
	7	PZ[7]	PY[11]	PZ[7]	PY[13]
	8	PZ[8]	PY[12]	PZ[8]	PY[14]
	9	PZ[9]	PY[13]	PZ[9]	PX[0]
	10	PZ[10]	PY[14]	PZ[10]	PX[1]
	11	PZ[11]	PY[15]	PZ[11]	PX[4]
	12	PZ[12]	PX[0]	PZ[12]	PX[5]
	13	PZ[13]	PX[1]	PZ[13]	PX[6]
	14	PZ[14]	PX[2]	PZ[14]	PX[7]
	15	PZ[15]	PX[3]	PZ[15]	PX[8]
	16	PY[0]	PX[4]	PY[0]	PX[11]
	17	PY[1]	PX[5]	PY[1]	PX[12]
	18	PY[2]	PX[6	PY[2]	PX[15]
	19	PY[3]	PX[7	PY[4]	PW[0]
	20	PY[4]	PX[8	PY[7]	PW[3]
	21	PY[5]	PX[9	PY[10]	—
	22	PY[6]	PX[10	PY[15]	—
	23	PY[7]	PX[11	PX[1]	—
	24	PY[8]	PX[12	PX[2]	—
	25	PY[9]	PX[13	PX[3]	—
	26	PY[10]	PX[14	PX[9]	—
	27	PY[11]	PX[15	PX[10]	—
	28	PY[12]	PW[0]	PX[13]	—
	29	PY[13]	PW[1]	PX[14]	—
	30	PY[14]	PW[2]	PW[1]	—
	31	PY[15]	PW[3]	PW[2]	—

Table continues on the next page...

Table 35-3. SAR ADC channel map (continued)

	Source of Inputs				
	Channel	**ADC0**	**ADC1**	**ADC2**	**ADC3**
Internal Channels	32	TEMP_SENS1	TEMP_SENS2	VSS_HV_ADR_SAR	ADC_BGREF
	33	ADC_BGREF[1, 1, 2, 2]	ADC_BGREF[1, 2]	VDD_HV_ADV_SAR	—
	34	PMC signals	VSS_HV_ADR_SAR	VSS_HV_ADR_SAR	VSS_HV_ADR_SAR
	35	VSS_HV_ADR_SAR	VDD_HV_ADV_SAR	VDD_HV_ADR_SAR	VDD_HV_ADV_SAR
	36	VDD_HV_ADV_SAR	VSS_HV_ADR_SAR	—	VSS_HV_ADR_SAR
	37	VSS_HV_ADR_SAR	VDD_HV_ADR_SAR	—	VDD_HV_ADR_SAR
	38	VDD_HV_ADR_SAR	—	—	—
	39	—	—	—	—
	40	—	—	—	—
	41	—	—	—	—
	42	—	—	—	—
	43	—	—	—	—
	44	—	—	—	—
	45	—	—	—	—
	46	—	—	—	—
	47	—	—	—	—
	48	—	—	—	—
	49	—	—	—	—
	50	—	—	—	—
	51	—	—	—	—
	52	—	—	—	—
	53	—	—	—	—
	54	—	—	—	—
	55	—	—	—	—
	56	—	—	—	—
	57	—	—	—	—
	58	—	—	—	—
	59	—	—	—	—
	60	—	—	—	—
	61	—	—	—	—
	62	—	—	—	—
	63	—	—	—	—

Table continues on the next page...

Table 35-3. SAR ADC channel map (continued)

	Source of Inputs				
	Channel	ADC0	ADC1	ADC2	ADC3
Externally Multiplexed Channels	64	—	—	—	PX[5]
	65	—	—	—	PX[5]
	66	—	—	—	PX[5]
	67	—	—	—	PX[5]
	68	—	—	—	PX[5]
	69	—	—	—	PX[5]
	70	—	—	—	PX[5]
	71	—	—	—	PX[5]
	72	—	—	—	PX[6]
	73	—	—	—	PX[6]
	74	—	—	—	PX[6]
	75	—	—	—	PX[6]
	76	—	—	—	PX[6]
	77	—	—	—	PX[6]
	78	—	—	—	PX[6]
	79	—	—	—	PX[6]
	80	—	—	—	PX[7]
	81	—	—	—	PX[7]
	82	—	—	—	PX[7]
	83	—	—	—	PX[7]
	84	—	—	—	PX[7]
	85	—	—	—	PX[7]
	86	—	—	—	PX[7]
	87	—	—	—	PX[7]
	88	—	—	—	PX[8]
	89	—	—	—	PX[8]
	90	—	—	—	PX[8]
	91	—	—	—	PX[8]
	92	—	—	—	PX[8]
	93	—	—	—	PX[8]
	94	—	—	—	PX[8]
	95	—	—	—	PX[8]

1. The bandgap reference voltage (ADC_BGREF) can be sampled from either ADC0 or ADC1 but must not be sampled simultaneously. Failure to adhere to this restriction results in inaccurate conversion.
2. This channel is functional from 4.5V to 5.5V of the corresponding reference (VDD_HV_ADR_SAR).

35.1.6 SARADC control registers

Not all control registers are applicable to every on-chip SAR ADC. The following table shows the registers applicable to each.

Table 35-4. SAR ADC control register applicability

Register	ADC0	ADC1	ADC2	ADC3
Main Configuration Register (ADC_MCR)	yes	yes	yes	yes
Main Status register (ADC_MSR)	yes	yes	yes	yes
Interrupt Status Register (ADC_ISR)	yes	yes	yes	yes
Channel Pending register 0 (ADC_CEOCFR0)	yes	yes	yes	yes
Channel Pending register 1 (ADC_CEOCFR1)	yes	yes	yes	yes
Channel Pending register 2 (ADC_CEOCFR2)	no	no	no	yes
Interrupt Mask Register (ADC_IMR)	yes	yes	yes	yes
Channel Interrup Mask Register 0 (ADC_CIMR0)	yes	yes	yes	yes
Channel Interrup Mask Register 1 (ADC_CIMR1)	yes	yes	yes	yes
Channel Interrup Mask Register 2 (ADC_CIMR2)	no	no	no	yes
Watchdog Threshold Interrupt Status Register (ADC_WTISR)	yes	yes	yes	yes
Watchdog Threshold Interrupt Mask Register (ADC_WTIMR)	yes	yes	yes	yes
DMA Enable register (ADC_DMAE)	yes	yes	yes	yes
DMA Channel Select Register 0 (ADC_DMAR0)	yes	yes	yes	yes
DMA Channel Select Register 1 (ADC_DMAR1)	yes	yes	yes	yes
DMA Channel Select Register 2 (ADC_DMAR2)	no	no	no	yes
Threshold Register (ADC_THRHLR0)	yes	yes	yes	yes
Threshold Register (ADC_THRHLR1)	yes	yes	yes	yes
Threshold Register (ADC_THRHLR2)	yes	yes	yes	yes
Threshold Register (ADC_THRHLR3)	yes	yes	yes	yes
Presampling Control Register (ADC_PSCR)	no	yes	yes	yes
Presampling register 0 (ADC_PSR0)	no	yes	yes	yes
Presampling register 1 (ADC_PSR1)	no	yes	yes	yes
Presampling register 2 (ADC_PSR2)	no	no	no	yes
Conversion Timing Register 0 (ADC_CTR0)	yes	yes	yes	yes
Conversion Timing Register 1 (ADC_CTR1)	yes	yes	yes	yes
Conversion Timing Register 2 (ADC_CTR2)	no	no	no	yes
Channel Pending register 0 (ADC_NCMR0)	yes	yes	yes	yes
Channel Pending register 1 (ADC_NCMR1)	yes	yes	yes	yes
Channel Pending register 2 (ADC_NCMR2)	no	no	no	yes
Injected Conversion Mask Register 0 (ADC_JCMR0)	yes	yes	yes	yes
Injected Conversion Mask Register 1 (ADC_JCMR1)	yes	yes	yes	yes
Injected Conversion Mask Register 2 (ADC_JCMR2)	no	no	no	yes

Table continues on the next page...

Table 35-4. SAR ADC control register applicability (continued)

Register	ADC0	ADC1	ADC2	ADC3
Decode Signals Delay Register (ADC_DSDR)	no	no	no	yes
Power Down Exit Delay Register (ADC_PDEDR)	yes	yes	yes	yes
Channel Data Register (Precision Channels) (ADC_CDR0)	yes	yes	yes	yes
Channel Data Register (Precision Channels) (ADC_CDR1)	yes	yes	yes	yes
Channel Data Register (Precision Channels) (ADC_CDR2)	yes	yes	yes	yes
Channel Data Register (Precision Channels) (ADC_CDR3)	yes	yes	yes	yes
Channel Data Register (Precision Channels) (ADC_CDR4)	yes	yes	yes	yes
Channel Data Register (Precision Channels) (ADC_CDR5)	yes	yes	yes	yes
Channel Data Register (Precision Channels) (ADC_CDR6)	yes	yes	yes	yes
Channel Data Register (Precision Channels) (ADC_CDR7)	yes	yes	yes	yes
Channel Data Register (Precision Channels) (ADC_CDR8)	yes	yes	yes	yes
Channel Data Register (Precision Channels) (ADC_CDR9)	yes	yes	yes	yes
Channel Data Register (Precision Channels) (ADC_CDR10)	yes	yes	yes	yes
Channel Data Register (Precision Channels) (ADC_CDR11)	yes	yes	yes	yes
Channel Data Register (Precision Channels) (ADC_CDR12)	yes	yes	yes	yes
Channel Data Register (Precision Channels) (ADC_CDR13)	yes	yes	yes	yes
Channel Data Register (Precision Channels) (ADC_CDR14)	yes	yes	yes	yes
Channel Data Register (Precision Channels) (ADC_CDR15)	yes	yes	yes	yes
Channel Data Register (Precision Channels) (ADC_CDR16)	yes	yes	yes	yes
Channel Data Register (Precision Channels) (ADC_CDR17)	yes	yes	yes	yes
Channel Data Register (Precision Channels) (ADC_CDR18)	yes	yes	yes	yes
Channel Data Register (Precision Channels) (ADC_CDR19)	yes	yes	yes	yes
Channel Data Register (Precision Channels) (ADC_CDR20)	yes	yes	yes	yes
Channel Data Register (Precision Channels) (ADC_CDR21)	yes	yes	yes	no
Channel Data Register (Precision Channels) (ADC_CDR22)	yes	yes	yes	no
Channel Data Register (Precision Channels) (ADC_CDR23)	yes	yes	yes	no
Channel Data Register (Precision Channels) (ADC_CDR24)	yes	yes	yes	no
Channel Data Register (Precision Channels) (ADC_CDR25)	yes	yes	yes	no
Channel Data Register (Precision Channels) (ADC_CDR26)	yes	yes	yes	no
Channel Data Register (Precision Channels) (ADC_CDR27)	yes	yes	yes	no
Channel Data Register (Precision Channels) (ADC_CDR28)	yes	yes	yes	no
Channel Data Register (Precision Channels) (ADC_CDR29)	yes	yes	yes	no
Channel Data Register (Precision Channels) (ADC_CDR30)	yes	yes	yes	no
Channel Data Register (Precision Channels) (ADC_CDR31)	yes	yes	yes	no
Channel Data Register (Internal Channels) (ADC_CDR32)	yes	yes	yes	yes
Channel Data Register (Internal Channels) (ADC_CDR33)	yes	yes	yes	yes
Channel Data Register (Internal Channels) (ADC_CDR34)	yes	yes	yes	yes
Channel Data Register (Internal Channels) (ADC_CDR35)	yes	yes	yes	yes
Channel Data Register (Internal Channels) (ADC_CDR36)	yes	yes	yes	yes

Table continues on the next page...

Table 35-4. SAR ADC control register applicability (continued)

Register	ADC0	ADC1	ADC2	ADC3
Channel Data Register (Internal Channels) (ADC_CDR37)	yes	yes	no	yes
Channel Data Register (Internal Channels) (ADC_CDR38)	yes	yes	no	yes
Channel Data Register (Internal Channels) (ADC_CDR39)	yes	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR40)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR41)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR42)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR43)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR44)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR45)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR46)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR47)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR48)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR49)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR50)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR51)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR52)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR53)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR54)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR55)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR56)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR57)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR58)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR59)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR60)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR61)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR62)	no	no	no	no
Channel Data Register (Internal Channels) (ADC_CDR63)	no	no	no	no
Channel Data Register (External Channels) (ADC_CDR64)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR65)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR66)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR67)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR68)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR69)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR70)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR71)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR72)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR73)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR74)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR75)	no	no	no	yes

Table continues on the next page...

Table 35-4. SAR ADC control register applicability (continued)

Register	ADC0	ADC1	ADC2	ADC3
Channel Data Register (External Channels) (ADC_CDR76)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR77)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR78)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR79)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR80)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR81)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR82)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR83)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR84)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR85)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR86)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR87)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR88)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR89)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR90)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR91)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR92)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR93)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR94)	no	no	no	yes
Channel Data Register (External Channels) (ADC_CDR95)	no	no	no	yes
Channel Watchdog Select Register 0 (ADC_CWSELR0)	yes	yes	yes	yes
Channel Watchdog Select Register 1 (ADC_CWSELR1)	yes	yes	yes	yes
Channel Watchdog Select Register 2 (ADC_CWSELR2)	yes	yes	yes	yes
Channel Watchdog Select Register 3 (ADC_CWSELR3)	yes	yes	yes	no
Channel Watchdog Select Register 4 (ADC_CWSELR4)	yes	yes	yes	yes
Channel Watchdog Select Register 8 (ADC_CWSELR8)	no	no	no	yes
Channel Watchdog Select Register 9 (ADC_CWSELR9)	no	no	no	yes
Channel Watchdog Select Register 10 (ADC_CWSELR10)	no	no	no	yes
Channel Watchdog Select Register 11 (ADC_CWSELR11)	no	no	no	yes
Channel Watchdog Enable Register 0 (ADC_CWENR0)	yes	yes	yes	yes
Channel Watchdog Enable Register 1 (ADC_CWENR1)	yes	yes	yes	yes
Channel Watchdog Enable Register 2 (ADC_CWENR2)	no	no	no	yes
Analog Watchdog Out of Range Register (ADC_AWORR0)	yes	yes	yes	yes
Analog Watchdog Out of Range Register (ADC_AWORR1)	yes	yes	yes	yes
Analog Watchdog Out of Range Register (ADC_AWORR2)	no	no	no	yes
Self Test Configuration Register 1 (ADC_STCR1)	yes	yes	yes	yes
Self Test Configuration Register 2 (ADC_STCR2)	yes	yes	yes	yes
Self Test Configuration Register 3 (ADC_STCR3)	yes	yes	yes	yes
Self Test Baud Rate Register (ADC_STBRR)	yes	yes	yes	yes

Table continues on the next page...

Table 35-4. SAR ADC control register applicability (continued)

Register	ADC0	ADC1	ADC2	ADC3
Self Test Status Register 1 (ADC_STSR1)	yes	yes	yes	yes
Self Test Status Register 2 (ADC_STSR2)	no	no	no	no
Self Test Status Register 3 (ADC_STSR3)	yes	yes	yes	yes
Self Test Status Register 4 (ADC_STSR4)	yes	yes	yes	yes
Self Test Data Register 1 (ADC_STDR1)	no	no	no	no
Self Test Data Register 2 (ADC_STDR2)	no	no	no	no
Self Test Analog Watchdog Register 0 (ADC_STAW0R)	no	no	no	no
Self Test Analog Watchdog Register 1A (ADC_STAW1AR)	yes	yes	yes	yes
Self Test Analog Watchdog Register 1B (ADC_STAW1BR)	yes	yes	yes	yes
Self Test Analog Watchdog Register 2 (ADC_STAW2R)	yes	yes	yes	yes
Self Test Analog Watchdog Register 4 (ADC_STAW4R)	yes	yes	yes	yes
Self Test Analog Watchdog Register 5 (ADC_STAW5R)	yes	yes	yes	yes
Calibration, BIST Control and status Register (ADC_CALBISTREG)	yes	yes	yes	yes
Offset and Gain User Register (ADC_OFSGNUSR)	yes	yes	yes	yes

35.1.7 SAR ADC trigger inputs

The SAR ADC supports two trigger input modes, normal and injected. The timer channel and external pin trigger inputs to the BCTU are connected to the SAR ADC through the injected trigger, and the normal trigger input is connected to ETPU and EMIOS outputs and external pins through the SIUL2 MSCR registers.

Each SAR ADC has the same set of trigger inputs (ETPU, EMIOS, PIT, package pin) connected through the BCTU to the injected trigger. This allows triggering of more than one ADC from the same trigger source. Each SAR ADC also has the same set of normal trigger inputs and group of sources, but actual channel numbers and pins are different.

The mapping of ETPU, EMIOS, PIT, and package pins to BCTU trigger inputs and SAR normal trigger inputs is defined in the I/O Signal Description and Input Multiplexing Tables, which are included in a Microsoft Excel file attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the Excel file to open it and select the I/O Signal Description Table tab.

35.1.8 SAR ADC externally muxed inputs

SAR_3 supports externally muxed ADC inputs. SAR_0, SAR_1, and SAR_2 do not support externally muxed ADC inputs.

SAR_3 supports 4 external multiplexers, with 8 channels each.

The following figure gives the external mux connections for SAR_3.

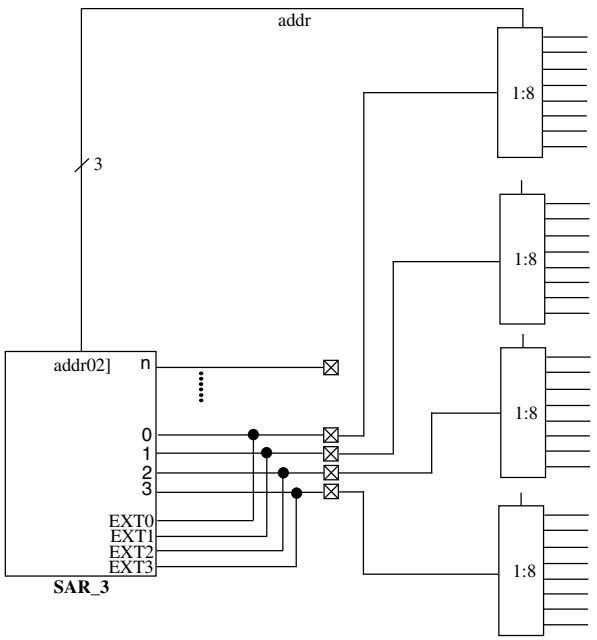


Figure 35-2. SAR_3 external mux inputs

The 3 address outputs to the external muxes are connected to package pins on the device. External mux addresses are available on the following pins:

- External mux address 0 : PF[0] , PG[1]
- External mux address 1 : PF[1] , PF[13]
- External mux address 2 : PF[2] , PF[12]

The following table shows the correspondence between the channels and the external mux addresses.

Table 35-5. Channel and External Mux Address Mapping

Channel	External Mux Input	External Mux Address
64	PX[5]	3'b000
65	PX[5]	3'b001
66	PX[5]	3'b010
67	PX[5]	3'b011
68	PX[5]	3'b100
69	PX[5]	3'b101
70	PX[5]	3'b110
71	PX[5]	3'b111
72	PX[6]	3'b000
73	PX[6]	3'b001

Table continues on the next page...

Table 35-5. Channel and External Mux Address Mapping (continued)

Channel	External Mux Input	External Mux Address
74	PX[6]	3'b010
75	PX[6]	3'b011
76	PX[6]	3'b100
77	PX[6]	3'b101
78	PX[6]	3'b110
79	PX[6]	3'b111
80	PX[7]	3'b000
81	PX[7]	3'b001
82	PX[7]	3'b010
83	PX[7]	3'b011
84	PX[7]	3'b100
85	PX[7]	3'b101
86	PX[7]	3'b110
87	PX[7]	3'b111
88	PX[8]	3'b000
89	PX[8]	3'b001
90	PX[8]	3'b010
91	PX[8]	3'b011
92	PX[8]	3'b100
93	PX[8]	3'b101
94	PX[8]	3'b110
95	PX[8]	3'b111

35.1.9 SAR ADC conversion result processing

SAR ADC conversions are written to the result registers in the digital interface, where they can be read by software. In addition, the conversion results can be transferred via DMA to system RAM or the decimation filter block input registers. When BCTU triggered conversions are enabled, results are automatically sent to the BCTU result registers, in addition to the ADC digital interface result registers.

35.2 Analog-to-Digital Converter (ADC) reference

35.2.1 Introduction

The analog-to-digital converter (ADC) uses Successive Approximation Register (SAR) architecture and provides accurate and fast conversions for a wide range of applications.

35.2.2 Overview

35.2.2.1 Parts of the ADC

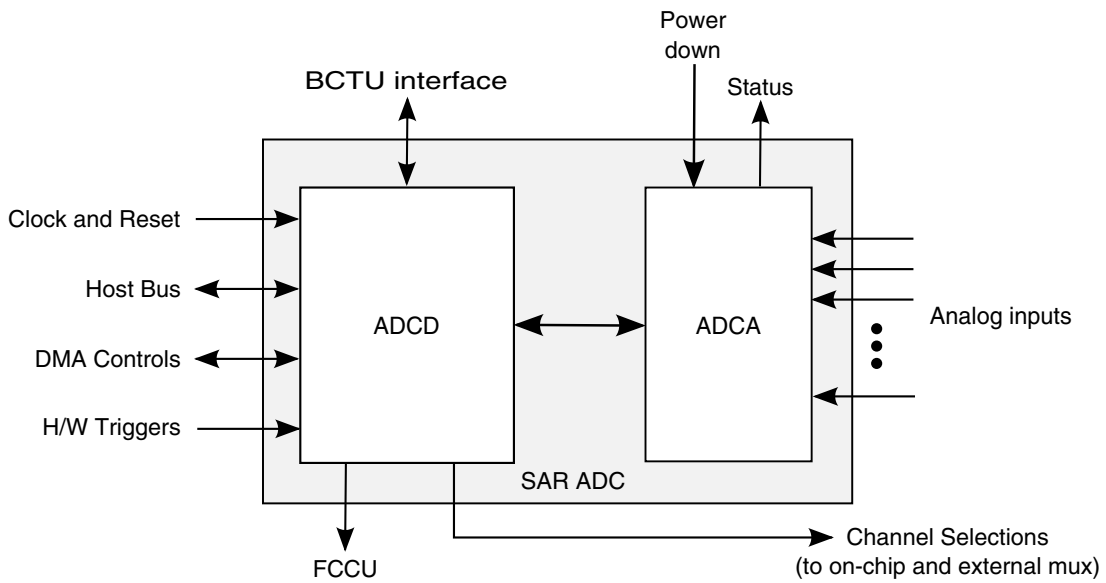


Figure 35-3. ADC high-level block diagram

The ADC is comprised of:

- An ADC digital interface (ADCD). The ADCD holds the configuration, control, and status registers accessible to software via the host bus. The converted value is stored in a register and then transferred through the DMA or host access.
- An ADC analog hard macrocell (ADCA). The analog channel inputs are fed to the inputs of the ADCA. Each channel is sampled for a specific duration and compared with an analog voltage generated with digital code via a digital-to-analog converter (DAC) in the ADCA. The comparison result is given to the ADCD to generate the converted digital value.

35.2.2.2 ADC clock signals

The ADC uses the following clock signals:

- Bus clock—bus clock at ADCD. Used to access the registers and the operating clock of inter-module interfaces.
- SAR_CLK—derived from *bus clock*, this signal serves as the operating clock for the ADCA and the SAR controller inside the ADCD.

The relation between the two clocks is determined by the setting of the ADC_MCR[ADCLKSEL] field.

- If ADC_MCR[ADCLKSEL] is 1, the clock signals are the same
- If ADC_MCR[ADCLKSEL] is 0, SAR_CLK is half of the bus clock.

NOTE

1. Regardless of the setting of the ADC_MCR[ADCLKSEL] field, the maximum clock speed for the ADC clock (SAR_CLK) is 80 MHz.
2. Depending on your chip's configuration and timing needs, MCR[ADCLKSEL] may always need to be 1.

If the bus clock for the microcontroller is 80 MHz and ADC_MCR[ADCLKSEL] is 1, the ADC clock (SAR_CLK) is 80 MHz.

If the bus clock for the microcontroller is 80 MHz and ADC_MCR[ADCLKSEL] is 0, the ADC clock (SAR_CLK) is 40 MHz.

See ADCLKSEL for details on the ADC_MCR register fields.

35.2.2.3 About ADC channels

ADC channels are divided into 3 groups, each with independent configuration settings. Each group may have different accuracy/performance levels depending on system need (check device configuration for details).

The groups are:

- Group 1, referred to as *precision* channels:
  - 0 - 31 (ADC0)
  - 0 - 31 (ADC1)

    - 0 - 31 (ADC2)
    - 0 - 20 (ADC3)
- Group 2, referred to as *internal* channels:
    - 32 - 39 (ADC0)
    - 32 - 38 (ADC1)
    - 32 - 36 (ADC2)
    - 32 - 38 (ADC3)
- Group 3, referred to as *external* channels[1, 1] :
    - none (ADC0)
    - none (ADC1)
    - none (ADC2)
    - 64 to 95 (ADC3)

NOTE

Accuracy of *external* channels can be increased by increasing the sampling duration, i.e., by increasing the value in the ADC_CTR*n*[INPSAMP] register field. This should be done with caution, though, because increasing sampling duration also increases total conversion time, resulting in a *decrease* in the number of samples per second.

35.2.2.3.1 Channel configuration

ADC channels can be configured in the following ways:

- The sampling duration of each channel type, i.e., *precision* channels, *internal* channels, *external* channels, is controlled independently, through programming the respective Conversion Timing Register (CTR).
- Separate mask registers for each channel type can be programmed to configure the channels to be converted (each bit in these registers represents one channel).
- External channel selection is provided through external decode signals and is available at the port. Each set of eight external channels can be mapped to an analog input. (See the chip-specific configuration for availability.)
- Temperature Sensor (TSENS) and other special channels (band gap, supply etc.) are available on specific channels. (See the chip-specific configuration for availability.)
- Analog watchdogs allow continuous hardware monitoring of the converted value from input channels. (See the chip-specific configuration for availability.)
- Capacitive self test
- Presampling

1. This group has an 8 x 1 external multiplexing option. A 3-bit output control is provided for shared use by 4 multiplexers.

NOTE

Each type of channel is converted in the same way.

35.2.2.3.2 Initiating conversions

Conversions can be initiated by either software or hardware.

The ADCD has an on-chip Cross-Triggering Unit (BCTU) interface that can also initiate conversions. The BCTU interface supports BCTU Trigger mode and BCTU Control mode (mode is selected via setting the ADC_MCR[CTU_MODE] field).

An example of an application for which BCTU triggering is suitable is a current control loop for lamps or LEDs. Using an on-chip timer module as trigger source, a BCTU can initiate periodic conversions.

35.2.2.3.3 Interrupt/DMA support

The ADCD provides interrupt/DMA support for each type of channel for various end-of-channel conversion conditions. Data can be transferred via DMA.

35.2.2.3.4 Self-test

The ADCD can be configured to periodically check the health of the ADCA through various self-tests and communicate any critical/non-critical faults to an on-chip Fault Collection and Control Unit (FCCU) if available (See the chip-specific section for availability). The severity (critical/noncritical) of the different tests is programmable.

35.2.3 Features

The following general features are implemented in the ADC. (See the chip-specific section for availability of features, as each instance of an ADC within a chip may support different features.)

- Resolution: 12-bit
- Maximum speed: 1 Mega samples/second at 80 MHz
- Up to 60 channels
- Separate sampling time register for each available range of channels:
  - *precision* channels:
    - 0 - 31 (ADC0)
    - 0 - 31 (ADC1)
    - 0 - 31 (ADC2)
    - 0 - 20 (ADC3)

  - *internal* channels:
    - 32 - 39 (ADC0)
    - 32 - 38 (ADC1)
    - 32 - 36 (ADC2)
    - 32 - 38 (ADC3)
  - *external* channels:
    - none (ADC0)
    - none (ADC1)
    - none (ADC2)
    - 64 to 95 (ADC3)

NOTE

The Temperature Sensor uses the value in the CTR1 register

- External decode signals available for (8 × 1) configuration (external channels only)
- Modes of operation:
  - Normal
  - Injected
  - BCTU
- Normal mode supports One-Shot/Scan (continuous)
- Injected mode supports One-Shot
- Presampling
- Power-Down mode
- Two abort features that allow you to abort either a single channel in a chain or the full chain in normal and injected mode
- Each channel has a dedicated register for conversion results and mode of operation (normal, injected, or BCTU)
- Analog watchdog support:
  - Up to 4 analog watchdogs
  - Configurable for different channels
- Auto clock-off feature
- Programmable clock prescaler (bus clock divided by 2)
- Two BCTU modes: Control and Trigger
- DMA support for each channel
- Interrupts for the following conditions:
  - End of conversion for a single channel for both normal and injected conversions
  - End of conversion for a chain for both normal and injected conversions
  - End of BCTU conversion
  - Watchdog thresholds crossover
- Software-initiated calibration
- Self-testing feature

35.2.4 Memory Map/Register Definition

The following tables describe the registers and bit meanings for this module.

NOTE

There will be an Access Error for Reserved and out-of-range addresses. Special exceptions are the Reserved spaces at 039Ch and 03A4h: Do not write or change any value in these locations, or the ADC might malfunction.

NOTE

See the chip-specific configuration information because not all registers are available in all ADC instances.

Control logic registers:

- Main Configuration Register (ADC_MCR)
- Main Status register (ADC_MSR)
- Interrupt Status Register (ADC_ISR)
- Channel Pending register 0 (ADC_CEOCFR0)
- Channel Pending register 1 (ADC_CEOCFR1)
- Channel Pending register 2 (ADC_CEOCFR2)
- Interrupt Mask Register (ADC_IMR)
- Channel Interrupt Mask Register 0 (ADC_CIMR0)
- Channel Interrupt Mask Register 1 (ADC_CIMR1)
- Channel Interrupt Mask Register 2 (ADC_CIMR2)
- Watchdog Threshold Interrupt Status Register (ADC_WTISR)
- Watchdog Threshold Interrupt Mask Register (ADC_WTIMR)
- Analog Watchdog Out of Range Register 0 (ADC_AWORR0)
- Analog Watchdog Out of Range Register 1 (ADC_AWORR1)
- Analog Watchdog Out of Range Register 2 (ADC_AWORR2)

DMA registers:

- DMA Enable register (ADC_DMAE)
- DMA Channel Select Register 0 (ADC_DMAR0)
- DMA Channel Select Register 1 (ADC_DMAR1)
- DMA Channel Select Register 2 (ADC_DMAR2)

Threshold registers:

These registers are used to store the user programmable lower and upper thresholds 12-bit values. These registers include:

- Threshold Register (ADC_THRHLR*n*)

Presampling registers:

- Presampling Control Register (ADC_PSCR)
- Presampling register 0 (ADC_PSR0)
- Presampling register 1 (ADC_PSR1)
- Presampling register 2 (ADC_PSR2)

Conversion timing registers:

- Conversion Timing Register 0 (ADC_CTR0)
- Conversion Timing Register 1 (ADC_CTR1)
- Conversion Timing Register 2 (ADC_CTR2)

Mask register: These registers are used to program which input channels must be converted during Normal and Injected conversion. These registers include:

- Normal Conversion Mask Register 0 (ADC_NCMR0)
- Normal Conversion Mask Register 1 (ADC_NCMR1)
- Normal Conversion Mask Register 2 (ADC_NCMR2)
- Injected Conversion Mask Register 0 (ADC_JCMR0)
- Injected Conversion Mask Register 1 (ADC_JCMR1)
- Injected Conversion Mask Register 2 (ADC_JCMR2)

Delay registers:

- Power Down Exit Delay Register (ADC_PDEDR)
- Decode Signals Delay Register (ADC_DSDR)

Data registers:

The conversion results for the internal channels are loaded into data registers. Each data register also gives information regarding the corresponding result. These registers start with CDR0:

- Channel Data Register *n* (Precision Channels) (ADC_CDR*n*)

Self test registers:

- Self Test Configuration Register 1 (ADC_STCR1)
- Self Test Configuration Register 2 (ADC_STCR2)
- Self Test Configuration Register 3 (ADC_STCR3)
- Self Test Status Register 1 (ADC_STSR1)
- Self Test Status Register 2 (ADC_STSR2)
- Self Test Status Register 3 (ADC_STSR3)
- Self Test Status Register 4 (ADC_STSR4)
- Self Test Baud Rate Register (ADC_STBRR)
- Self Test Analog Watchdog Register 0 (ADC_STAW0R)
- Self Test Analog Watchdog Register 4 (ADC_STAW4R)

- Self Test Analog Watchdog Register 1A (ADC_STAW1AR)
- Self Test Analog Watchdog Register 1B (ADC_STAW1BR)
- Self Test Analog Watchdog Register 2 (ADC_STAW2R)
- Self Test Analog Watchdog Register 5 (ADC_STAW5R)
- Self Test Data Register 1 (ADC_STDR1)
- Self Test Data Register 2 (ADC_STDR2)

Other registers include:

- Calibration, BIST Control and status Register (ADC_CALBISTREG)
- Offset and Gain User Register (ADC_OFSGNUSR)

ADC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Main Configuration Register (ADC_MCR)	32	R/W	0000_0101h	35.24.1/ 994
4	Main Status register (ADC_MSR)	32	R	0000_0001h	35.24.2/ 997
10	Interrupt Status Register (ADC_ISR)	32	R/W	0000_0000h	35.24.3/ 999
14	Channel Pending register 0 (ADC_CEOCFR0)	32	R/W	0000_0000h	35.24.4/ 1000
18	Channel Pending register 1 (ADC_CEOCFR1)	32	R/W	0000_0000h	35.24.5/ 1003
1C	Channel Pending register 2 (ADC_CEOCFR2)	32	R/W	0000_0000h	35.24.6/ 1005
20	Interrupt Mask Register (ADC_IMR)	32	R/W	0000_0000h	35.24.7/ 1008
24	Channel Interrupt Mask Register 0 (ADC_CIMR0)	32	R/W	0000_0000h	35.24.8/ 1008
28	Channel Interrupt Mask Register 1 (ADC_CIMR1)	32	R/W	0000_0000h	35.24.9/ 1011
2C	Channel Interrupt Mask Register 2 (ADC_CIMR2)	32	R/W	0000_0000h	35.24.10/ 1013
30	Watchdog Threshold Interrupt Status Register (ADC_WTISR)	32	R/W	0000_0000h	35.24.11/ 1016
34	Watchdog Threshold Interrupt Mask Register (ADC_WTIMR)	32	R/W	0000_0000h	35.24.12/ 1017
40	DMA Enable register (ADC_DMAE)	32	R/W	0000_0000h	35.24.13/ 1018
44	DMA Channel Select Register 0 (ADC_DMAR0)	32	R/W	0000_0000h	35.24.14/ 1019
48	DMA Channel Select Register 1 (ADC_DMAR1)	32	R/W	0000_0000h	35.24.15/ 1022
4C	DMA Channel Select Register 2 (ADC_DMAR2)	32	R/W	0000_0000h	35.24.16/ 1024

Table continues on the next page...

ADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
60	Threshold Register (ADC_THRHLR0)	32	R/W	0FFF_0000h	35.24.17/ 1026
64	Threshold Register (ADC_THRHLR1)	32	R/W	0FFF_0000h	35.24.17/ 1026
68	Threshold Register (ADC_THRHLR2)	32	R/W	0FFF_0000h	35.24.17/ 1026
6C	Threshold Register (ADC_THRHLR3)	32	R/W	0FFF_0000h	35.24.17/ 1026
80	Presampling Control Register (ADC_PSCR)	32	R/W	0000_0000h	35.24.18/ 1027
84	Presampling register 0 (ADC_PSR0)	32	R/W	0000_0000h	35.24.19/ 1028
88	Presampling register 1 (ADC_PSR1)	32	R/W	0000_0000h	35.24.20/ 1030
8C	Presampling register 2 (ADC_PSR2)	32	R/W	0000_0000h	35.24.21/ 1032
94	Conversion Timing Register 0 (ADC_CTR0)	32	R/W	0000_0014h	35.24.22/ 1034
98	Conversion Timing Register 1 (ADC_CTR1)	32	R/W	0000_0014h	35.24.23/ 1035
9C	Conversion Timing Register 2 (ADC_CTR2)	32	R/W	0000_0014h	35.24.24/ 1036
A4	Normal Conversion Mask Register 0 (ADC_NCMR0)	32	R/W	0000_0000h	35.24.25/ 1037
A8	Normal Conversion Mask Register 1 (ADC_NCMR1)	32	R/W	0000_0000h	35.24.26/ 1039
AC	Normal Conversion Mask Register 2 (ADC_NCMR2)	32	R/W	0000_0000h	35.24.27/ 1041
B4	Injected Conversion Mask Register 0 (ADC_JCMR0)	32	R/W	0000_0000h	35.24.28/ 1043
B8	Injected Conversion Mask Register 1 (ADC_JCMR1)	32	R/W	0000_0000h	35.24.29/ 1045
BC	Injected Conversion Mask Register 2 (ADC_JCMR2)	32	R/W	0000_0000h	35.24.30/ 1047
C4	Decode Signals Delay Register (ADC_DSDR)	32	R/W	0000_0000h	35.24.31/ 1050
C8	Power Down Exit Delay Register (ADC_PDEDR)	32	R/W	0000_0000h	35.24.32/ 1050
100	Channel Data Register n (Precision Channels) (ADC_CDR0)	32	R	0000_0000h	35.24.33/ 1051
104	Channel Data Register n (Precision Channels) (ADC_CDR1)	32	R	0000_0000h	35.24.33/ 1051
108	Channel Data Register n (Precision Channels) (ADC_CDR2)	32	R	0000_0000h	35.24.33/ 1051

Table continues on the next page...

ADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10C	Channel Data Register n (Precision Channels) (ADC_CDR3)	32	R	0000_0000h	35.24.33/ 1051
110	Channel Data Register n (Precision Channels) (ADC_CDR4)	32	R	0000_0000h	35.24.33/ 1051
114	Channel Data Register n (Precision Channels) (ADC_CDR5)	32	R	0000_0000h	35.24.33/ 1051
118	Channel Data Register n (Precision Channels) (ADC_CDR6)	32	R	0000_0000h	35.24.33/ 1051
11C	Channel Data Register n (Precision Channels) (ADC_CDR7)	32	R	0000_0000h	35.24.33/ 1051
120	Channel Data Register n (Precision Channels) (ADC_CDR8)	32	R	0000_0000h	35.24.33/ 1051
124	Channel Data Register n (Precision Channels) (ADC_CDR9)	32	R	0000_0000h	35.24.33/ 1051
128	Channel Data Register n (Precision Channels) (ADC_CDR10)	32	R	0000_0000h	35.24.33/ 1051
12C	Channel Data Register n (Precision Channels) (ADC_CDR11)	32	R	0000_0000h	35.24.33/ 1051
130	Channel Data Register n (Precision Channels) (ADC_CDR12)	32	R	0000_0000h	35.24.33/ 1051
134	Channel Data Register n (Precision Channels) (ADC_CDR13)	32	R	0000_0000h	35.24.33/ 1051
138	Channel Data Register n (Precision Channels) (ADC_CDR14)	32	R	0000_0000h	35.24.33/ 1051
13C	Channel Data Register n (Precision Channels) (ADC_CDR15)	32	R	0000_0000h	35.24.33/ 1051
140	Channel Data Register n (Precision Channels) (ADC_CDR16)	32	R	0000_0000h	35.24.33/ 1051
144	Channel Data Register n (Precision Channels) (ADC_CDR17)	32	R	0000_0000h	35.24.33/ 1051
148	Channel Data Register n (Precision Channels) (ADC_CDR18)	32	R	0000_0000h	35.24.33/ 1051
14C	Channel Data Register n (Precision Channels) (ADC_CDR19)	32	R	0000_0000h	35.24.33/ 1051
150	Channel Data Register n (Precision Channels) (ADC_CDR20)	32	R	0000_0000h	35.24.33/ 1051
154	Channel Data Register n (Precision Channels) (ADC_CDR21)	32	R	0000_0000h	35.24.33/ 1051
158	Channel Data Register n (Precision Channels) (ADC_CDR22)	32	R	0000_0000h	35.24.33/ 1051
15C	Channel Data Register n (Precision Channels) (ADC_CDR23)	32	R	0000_0000h	35.24.33/ 1051
160	Channel Data Register n (Precision Channels) (ADC_CDR24)	32	R	0000_0000h	35.24.33/ 1051

Table continues on the next page...

ADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
164	Channel Data Register n (Precision Channels) (ADC_CDR25)	32	R	0000_0000h	35.24.33/ 1051
168	Channel Data Register n (Precision Channels) (ADC_CDR26)	32	R	0000_0000h	35.24.33/ 1051
16C	Channel Data Register n (Precision Channels) (ADC_CDR27)	32	R	0000_0000h	35.24.33/ 1051
170	Channel Data Register n (Precision Channels) (ADC_CDR28)	32	R	0000_0000h	35.24.33/ 1051
174	Channel Data Register n (Precision Channels) (ADC_CDR29)	32	R	0000_0000h	35.24.33/ 1051
178	Channel Data Register n (Precision Channels) (ADC_CDR30)	32	R	0000_0000h	35.24.33/ 1051
17C	Channel Data Register n (Precision Channels) (ADC_CDR31)	32	R	0000_0000h	35.24.33/ 1051
180	Channel Data Register n (Internal Channels) (ADC_CDR32)	32	R	0000_0000h	35.24.34/ 1053
184	Channel Data Register n (Internal Channels) (ADC_CDR33)	32	R	0000_0000h	35.24.34/ 1053
188	Channel Data Register n (Internal Channels) (ADC_CDR34)	32	R	0000_0000h	35.24.34/ 1053
18C	Channel Data Register n (Internal Channels) (ADC_CDR35)	32	R	0000_0000h	35.24.34/ 1053
190	Channel Data Register n (Internal Channels) (ADC_CDR36)	32	R	0000_0000h	35.24.34/ 1053
194	Channel Data Register n (Internal Channels) (ADC_CDR37)	32	R	0000_0000h	35.24.34/ 1053
198	Channel Data Register n (Internal Channels) (ADC_CDR38)	32	R	0000_0000h	35.24.34/ 1053
19C	Channel Data Register n (Internal Channels) (ADC_CDR39)	32	R	0000_0000h	35.24.34/ 1053
1A0	Channel Data Register n (Internal Channels) (ADC_CDR40)	32	R	0000_0000h	35.24.34/ 1053
1A4	Channel Data Register n (Internal Channels) (ADC_CDR41)	32	R	0000_0000h	35.24.34/ 1053
1A8	Channel Data Register n (Internal Channels) (ADC_CDR42)	32	R	0000_0000h	35.24.34/ 1053
1AC	Channel Data Register n (Internal Channels) (ADC_CDR43)	32	R	0000_0000h	35.24.34/ 1053
1B0	Channel Data Register n (Internal Channels) (ADC_CDR44)	32	R	0000_0000h	35.24.34/ 1053
1B4	Channel Data Register n (Internal Channels) (ADC_CDR45)	32	R	0000_0000h	35.24.34/ 1053
1B8	Channel Data Register n (Internal Channels) (ADC_CDR46)	32	R	0000_0000h	35.24.34/ 1053

Table continues on the next page...

ADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1BC	Channel Data Register n (Internal Channels) (ADC_CDR47)	32	R	0000_0000h	35.24.34/ 1053
200	Channel Data Register n (External Channels) (ADC_CDR64)	32	R	0000_0000h	35.24.35/ 1055
204	Channel Data Register n (External Channels) (ADC_CDR65)	32	R	0000_0000h	35.24.35/ 1055
208	Channel Data Register n (External Channels) (ADC_CDR66)	32	R	0000_0000h	35.24.35/ 1055
20C	Channel Data Register n (External Channels) (ADC_CDR67)	32	R	0000_0000h	35.24.35/ 1055
210	Channel Data Register n (External Channels) (ADC_CDR68)	32	R	0000_0000h	35.24.35/ 1055
214	Channel Data Register n (External Channels) (ADC_CDR69)	32	R	0000_0000h	35.24.35/ 1055
218	Channel Data Register n (External Channels) (ADC_CDR70)	32	R	0000_0000h	35.24.35/ 1055
21C	Channel Data Register n (External Channels) (ADC_CDR71)	32	R	0000_0000h	35.24.35/ 1055
220	Channel Data Register n (External Channels) (ADC_CDR72)	32	R	0000_0000h	35.24.35/ 1055
224	Channel Data Register n (External Channels) (ADC_CDR73)	32	R	0000_0000h	35.24.35/ 1055
228	Channel Data Register n (External Channels) (ADC_CDR74)	32	R	0000_0000h	35.24.35/ 1055
22C	Channel Data Register n (External Channels) (ADC_CDR75)	32	R	0000_0000h	35.24.35/ 1055
230	Channel Data Register n (External Channels) (ADC_CDR76)	32	R	0000_0000h	35.24.35/ 1055
234	Channel Data Register n (External Channels) (ADC_CDR77)	32	R	0000_0000h	35.24.35/ 1055
238	Channel Data Register n (External Channels) (ADC_CDR78)	32	R	0000_0000h	35.24.35/ 1055
23C	Channel Data Register n (External Channels) (ADC_CDR79)	32	R	0000_0000h	35.24.35/ 1055
240	Channel Data Register n (External Channels) (ADC_CDR80)	32	R	0000_0000h	35.24.35/ 1055
244	Channel Data Register n (External Channels) (ADC_CDR81)	32	R	0000_0000h	35.24.35/ 1055
248	Channel Data Register n (External Channels) (ADC_CDR82)	32	R	0000_0000h	35.24.35/ 1055
24C	Channel Data Register n (External Channels) (ADC_CDR83)	32	R	0000_0000h	35.24.35/ 1055
250	Channel Data Register n (External Channels) (ADC_CDR84)	32	R	0000_0000h	35.24.35/ 1055

Table continues on the next page...

ADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
254	Channel Data Register n (External Channels) (ADC_CDR85)	32	R	0000_0000h	35.24.35/ 1055
258	Channel Data Register n (External Channels) (ADC_CDR86)	32	R	0000_0000h	35.24.35/ 1055
25C	Channel Data Register n (External Channels) (ADC_CDR87)	32	R	0000_0000h	35.24.35/ 1055
260	Channel Data Register n (External Channels) (ADC_CDR88)	32	R	0000_0000h	35.24.35/ 1055
264	Channel Data Register n (External Channels) (ADC_CDR89)	32	R	0000_0000h	35.24.35/ 1055
268	Channel Data Register n (External Channels) (ADC_CDR90)	32	R	0000_0000h	35.24.35/ 1055
26C	Channel Data Register n (External Channels) (ADC_CDR91)	32	R	0000_0000h	35.24.35/ 1055
270	Channel Data Register n (External Channels) (ADC_CDR92)	32	R	0000_0000h	35.24.35/ 1055
274	Channel Data Register n (External Channels) (ADC_CDR93)	32	R	0000_0000h	35.24.35/ 1055
278	Channel Data Register n (External Channels) (ADC_CDR94)	32	R	0000_0000h	35.24.35/ 1055
27C	Channel Data Register n (External Channels) (ADC_CDR95)	32	R	0000_0000h	35.24.35/ 1055
2B0	Channel Watchdog Select Register 0 (ADC_CWSELR0)	32	R/W	0000_0000h	35.24.36/ 1056
2B4	Channel Watchdog Select Register 1 (ADC_CWSELR1)	32	R/W	0000_0000h	35.24.37/ 1057
2B8	Channel Watchdog Select Register 2 (ADC_CWSELR2)	32	R/W	0000_0000h	35.24.38/ 1058
2BC	Channel Watchdog Select Register 3 (ADC_CWSELR3)	32	R/W	0000_0000h	35.24.39/ 1059
2C0	Channel Watchdog Select Register 4 (ADC_CWSELR4)	32	R/W	0000_0000h	35.24.40/ 1060
2D0	Channel Watchdog Select Register 8 (ADC_CWSELR8)	32	R/W	0000_0000h	35.24.41/ 1061
2D4	Channel Watchdog Select Register 9 (ADC_CWSELR9)	32	R/W	0000_0000h	35.24.42/ 1062
2D8	Channel Watchdog Select Register 10 (ADC_CWSELR10)	32	R/W	0000_0000h	35.24.43/ 1063
2DC	Channel Watchdog Select Register 11 (ADC_CWSELR11)	32	R/W	0000_0000h	35.24.44/ 1064
2E0	Channel Watchdog Enable Register 0 (ADC_CWENR0)	32	R/W	0000_0000h	35.24.45/ 1065
2E4	Channel Watchdog Enable Register 1 (ADC_CWENR1)	32	R/W	0000_0000h	35.24.46/ 1068

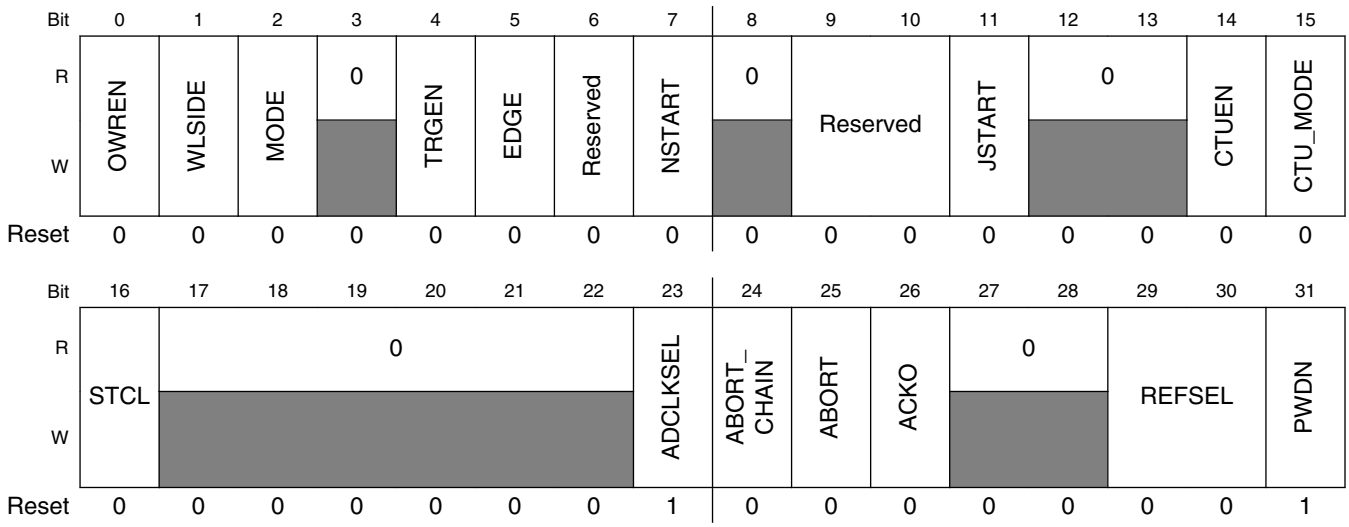
Table continues on the next page...

ADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2E8	Channel Watchdog Enable Register 2 (ADC_CWENR2)	32	R/W	0000_0000h	35.24.47/ 1069
2F0	Analog Watchdog Out of Range Register 0 (ADC_AWORR0)	32	R/W	0000_0000h	35.24.48/ 1072
2F4	Analog Watchdog Out of Range Register 1 (ADC_AWORR1)	32	R/W	0000_0000h	35.24.49/ 1075
2F8	Analog Watchdog Out of Range Register 2 (ADC_AWORR2)	32	R/W	0000_0000h	35.24.50/ 1077
340	Self Test Configuration Register 1 (ADC_STCR1)	32	R/W	1818_2507h	35.24.51/ 1079
344	Self Test Configuration Register 2 (ADC_STCR2)	32	R/W	0000_0005h	35.24.52/ 1080
348	Self Test Configuration Register 3 (ADC_STCR3)	32	R/W	0000_0300h	35.24.53/ 1083
34C	Self Test Baud Rate Register (ADC_STBRR)	32	R/W	0005_0000h	35.24.54/ 1084
350	Self Test Status Register 1 (ADC_STSR1)	32	R/W	0000_0000h	35.24.55/ 1085
354	Self Test Status Register 2 (ADC_STSR2)	32	R	0000_0000h	35.24.56/ 1087
358	Self Test Status Register 3 (ADC_STSR3)	32	R	0000_0000h	35.24.57/ 1088
35C	Self Test Status Register 4 (ADC_STSR4)	32	R	0000_0000h	35.24.58/ 1089
370	Self Test Data Register 1 (ADC_STDR1)	32	R	0000_0000h	35.24.59/ 1089
374	Self Test Data Register 2 (ADC_STDR2)	32	R	0000_0000h	35.24.60/ 1091
380	Self Test Analog Watchdog Register 0 (ADC_STAW0R)	32	R/W	0727_04C5h	35.24.61/ 1092
384	Self Test Analog Watchdog Register 1A (ADC_STAW1AR)	32	R/W	0003_0001h	35.24.62/ 1094
388	Self Test Analog Watchdog Register 1B (ADC_STAW1BR)	32	R/W	03E8_0ED0h	35.24.63/ 1095
38C	Self Test Analog Watchdog Register 2 (ADC_STAW2R)	32	R/W	0000_0FF9h	35.24.64/ 1096
394	Self Test Analog Watchdog Register 4 (ADC_STAW4R)	32	R/W	0010_0FF0h	35.24.65/ 1097
398	Self Test Analog Watchdog Register 5 (ADC_STAW5R)	32	R/W	0010_0FF0h	35.24.66/ 1098
3A0	Calibration, BIST Control and status Register (ADC_CALBISTREG)	32	R/W	C737_06F6h	35.24.67/ 1100
3A8	Offset and Gain User Register (ADC_OFSGNUSR)	32	R/W	0000_0000h	35.24.68/ 1102

35.24.1 Main Configuration Register (ADC_MCR)

Address: 0h base + 0h offset = 0h



ADC_MCR field descriptions

Field	Description
0 OWREN	Overwrite enable. 0 Older valid conversion data is not overwritten by newer conversion data. 1 Newer conversion result is always overwritten, irrespective of the validity of older conversion data.
1 WLSIDE	Write Left/Right aligned. 0 The conversion data is written right aligned from (32-resolution) to 31). 1 Data is left aligned (from 16 to (16 + resolution - 1).
2 MODE	One_Shot/Scan. 0 One_Shot Mode: configure the Normal conversion of one chain. 1 Scan Mode: configure continuous chain conversion mode; when the programmed chain conversion is finished it restarts immediately.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 TRGEN	External trigger enable. When set, enables the external trigger to start a conversion; if the external trigger feature is needed this bit must be set.
5 EDGE	Start trigger edge detection. If TRGEN is 1 this bit selects the falling (EDGE = zero) or rising (EDGE = one) edge for the external trigger.
6 Reserved	This field is reserved. **NOTE:** User must not overwrite default value of this field.

Table continues on the next page...

ADC_MCR field descriptions (continued)

Field	Description
7 NSTART	Start of Normal conversion. Setting this field starts a normal conversion. In One shot mode: This field is cleared as soon as conversion is started. Setting this field during any ongoing conversion keeps this bit high until the requested conversion is started. **NOTE:** This bit cannot be set (1) when PWDN = 1. This bit is always cleared when PWDN = 1.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–10 Reserved	This field is reserved.
11 JSTART	Injection start. Setting this field will start the configured injected conversion chain. Resetting this field has no effect, as the injected chain conversion cannot be interrupted. Setting this field during ongoing Injected conversion keeps this field high value until the requested conversion is started.
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 CTUEN	Crosstrigger Unit Enable. Enables the crosstriggering unit (BCTU). 0 The crosstriggering unit is disabled and the BCTU-triggered injected conversion cannot take place. 1 The crosstriggering unit is enabled and the BCTUtriggered injected conversion can take place.
15 CTU_MODE	Crosstrigger Unit MODE bit When both modes are present in an ADC instance, this bit sets the operating mode of the ADC with respect to the BCTU. Please see the chip-specific ADC section for whether a specific ADC instance has both modes. Otherwise, treat this bit as Reserved. 0 BCTU control mode is selected. 1 BCTU trigger mode is selected.
16 STCL	Self Testing Configuration Lock. 0 No lock 1 The self-testing configuration is locked i.e. STCR1, STCR2, STCR3, STBRR, STAW0R, STAW1AR, STAW1BR, STAW2R, STAW4R, STAW5R are write-protected. It can be used only in CPU and SCAN mode. The lock bit is cleared only by a peripheral reset.
17–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 ADCLKSEL	Analog Clock frequency Selector. This field can be written in Power Down mode only. Please check device specific additional restriction on programming of this field. 0 ADC clock frequency is half of bus clock 1 ADC clock frequency is equal to bus clock
24 ABORT_CHAIN	Abort Chain. If this bit is set the ongoing Chain Conversion is aborted. This bit gets reset by hardware as soon as a new conversion is requested.

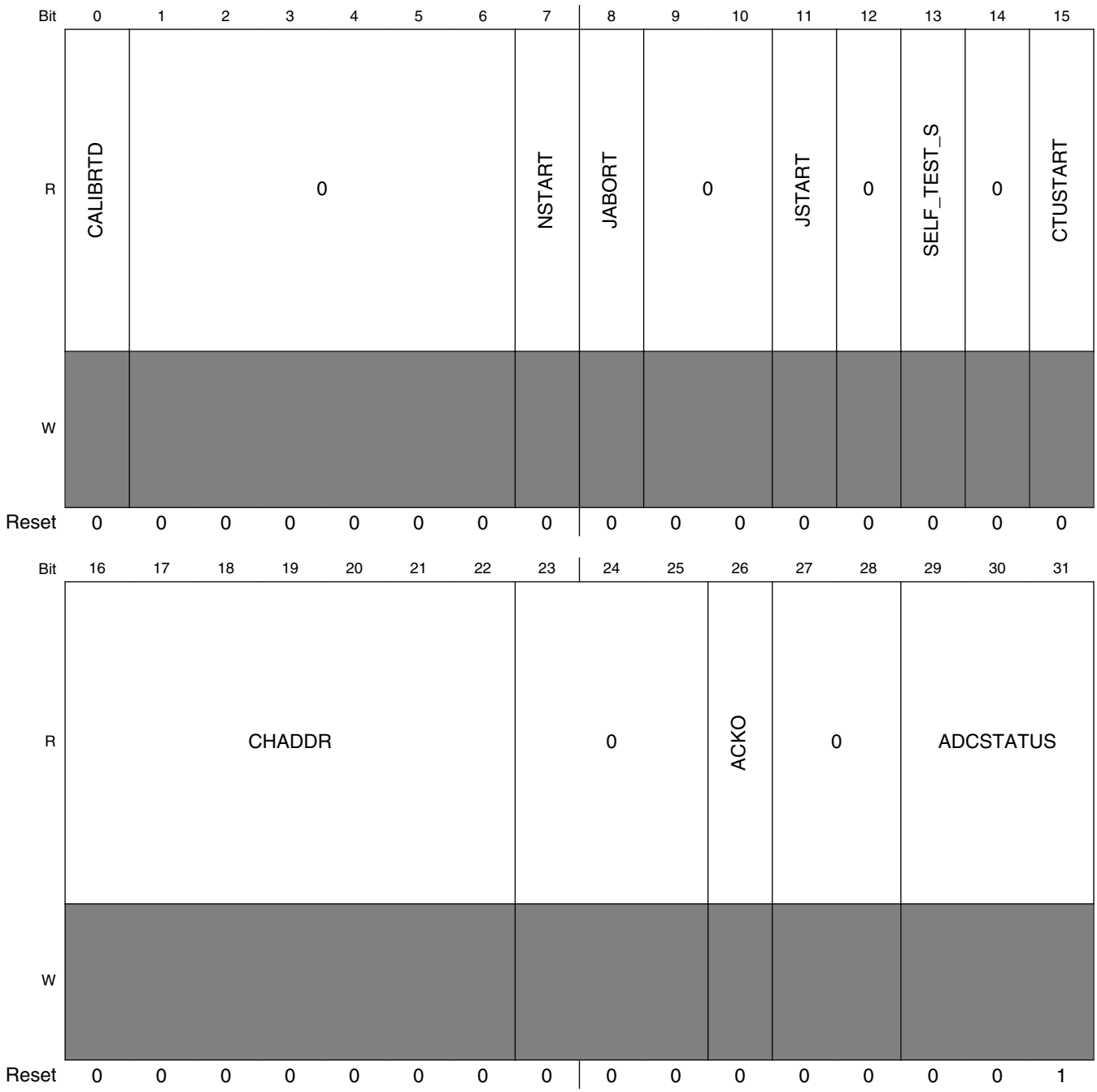
Table continues on the next page...

ADC_MCR field descriptions (continued)

Field	Description
	NOTE: Setting this bit in idle state (no normal/injected conversion is ongoing) has no effect. In this case it is reset immediately. During ongoing BCTU conversion this bit also cannot be programmed.
25 ABORT	Abort Conversion. If this bit is set, the ongoing channel conversion is aborted and the next channel conversion is invoked. This bit gets reset by hardware as soon as the new conversion is invoked. This bit cannot be written while self test conversion is ongoing. **NOTE:** Setting this bit in idle state (No Normal/injected conversion is ongoing) has no effect. In this case it is reset immediately. During ongoing BCTU conversion this bit cannot be programmed
26 ACKO	Auto clock off enable. If set, enables the Auto clock off feature.
27–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 REFSEL	Reference voltage selection for ADC analog part. 00 Select VREFH as reference voltage 01 Reserved 10 Reserved 11 Reserved
31 PWDN	Power-down enable. When this bit is set, the analog module is requested to enter Power-Down mode. When the ADC's status is PWDN, resetting this bit will start the ADC's transition to IDLE mode. In BCTU control mode MCR[CTUEN] must be reset before entering into power-down mode.

35.24.2 Main Status register (ADC_MSR)

Address: 0h base + 4h offset = 4h



ADC_MSR field descriptions

Field	Description
0 CALIBRTD	This bit indicates the ADC calibration status. This bit gets updated after running the High Accuracy Calibration mode.

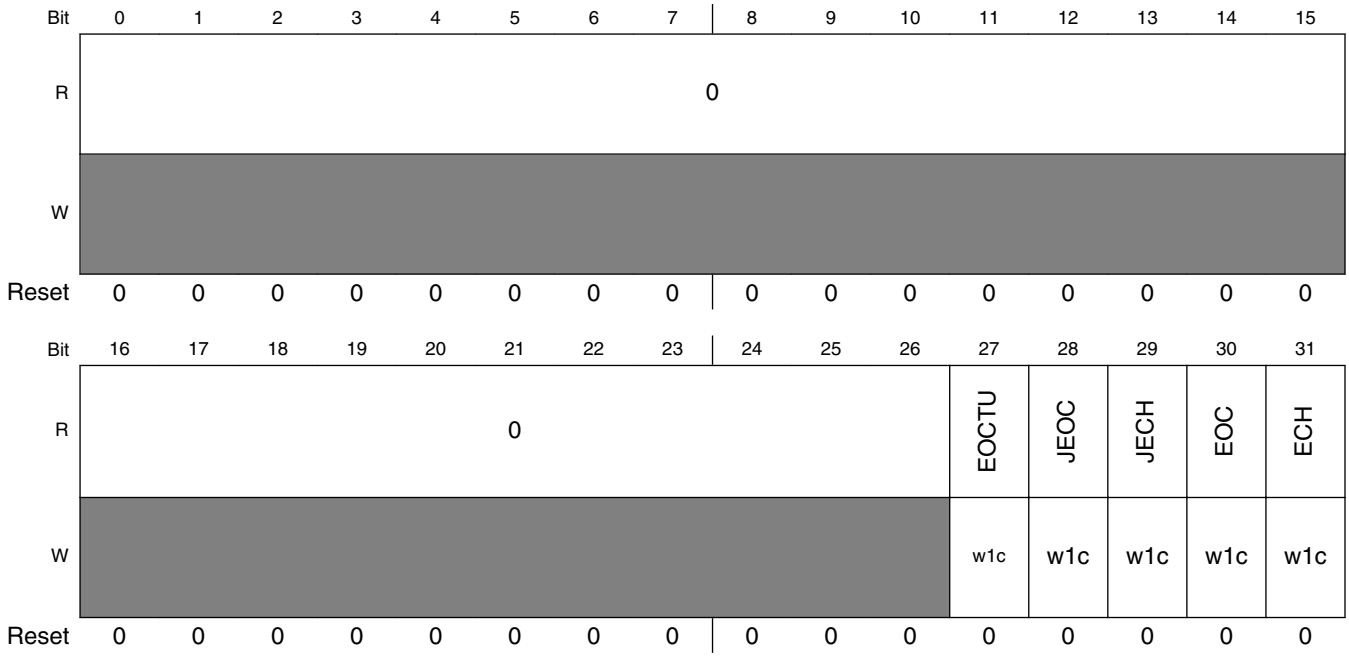
Table continues on the next page...

ADC_MSR field descriptions (continued)

Field	Description
	0 Uncalibrated or calibration unsuccessful 1 Calibration was run and it was successful
1–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 NSTART	This status bit is used to signal that a Normal conversion is ongoing.
8 JABORT	This status bit is used to signal that an Injected conversion has been aborted by BCTU. That bit is reset when a new injected conversion starts.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 JSTART	This status bit is used to signal that an Injected conversion is ongoing.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SELF_TEST_S	This status bit signals that self test conversion is ongoing.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 CTUSTART	This status bit is used to signal that a BCTU conversion is ongoing. This bit is set when a BCTU trigger pulse is received and the BCTU conversion starts.When BCTU trigger mode is enabled this bit is automatically reset when the conversion is completed.When Control Mode is enabled this bit is reset when the BCTU is disabled (CTUEN set to 0).
16–22 CHADDR	Channel under measure address. Status bits are used to signal which channel is under measure.
23–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 ACKO	Auto clock off enable. This status bit is used to signal if the Auto clock off feature is on.
27–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 ADCSTATUS	Status of the ADC. 000 Idle 001 Power Down 100 Sample 110 Conversion 010 Wait state (waiting to start conversion [external trigger]) The start of ADC conversion can be delayed (after a programmed duration in "DSDR" register) for a time after the application of trigger for External channels. During this wait period the ADCSTATUS is shown as "wait state". 011 Busy in Calibration

35.24.3 Interrupt Status Register (ADC_ISR)

Address: 0h base + 10h offset = 10h



ADC_ISR field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 EOCTU	End of BCTU conversion. This bit is set for a digital end of conversion on a the BCTU Channel; active when set.
28 JEOC	End of injected channel conversion. This bit is set for a digital end of conversion on an injected channel; active when set.
29 JECH	End of injected chain conversion. This bit is set for a digital end of chain conversion on an injected channel; active when set. **NOTE:** If there is no channel selected in the ADC_JCMR*n* register(s) and there is a start of conversion trigger, then ADC_ISR[JECH] is set and a JECH interrupt is immediately issued (if enabled). Similarly, if no channel is selected in the ADC_NCMR*n* register and a normal conversion is triggered, ADC_ISR[ECH] is set and an ECH interrupt is immediately issued, if enabled.
30 EOC	End of channel conversion. This bit is set for a digital end of conversion on a Normal Channel; active when set.
31 ECH	End of chain conversion. This bit is set for a digital end of chain conversion on a Normal Channel; active when set. **NOTE:** If there is no channel selected in the ADC_NCMR*n* register(s) and there is a start of conversion trigger, then ADC_ISR[ECH] is set and an ECH interrupt is immediately issued (if enabled).

Table continues on the next page...

ADC_ISR field descriptions (continued)

Field	Description
	Similarly, if no channel is selected in the ADC_JCMR*n* register and an injected conversion is triggered, ADC_ISR[JECH] is set and a JECH interrupt is immediately issued, if enabled.

35.24.4 Channel Pending register 0 (ADC_CEOCFR0)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EOCCH31	EOCCH30	EOCCH29	EOCCH28	EOCCH27	EOCCH26	EOCCH25	EOCCH24	EOCCH23	EOCCH22	EOCCH21	EOCCH20	EOCCH19	EOCCH18	EOCCH17	EOCCH16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOCCH15	EOCCH14	EOCCH13	EOCCH12	EOCCH11	EOCCH10	EOCCH9	EOCCH8	EOCCH7	EOCCH6	EOCCH5	EOCCH4	EOCCH3	EOCCH2	EOCCH1	EOCCH0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CEOCFR0 field descriptions

Field	Description
0 EOCCH31	EOC Channel 31. When set, the measure of channel 31 is completed.
1 EOCCH30	EOC Channel 30. When set, the measure of channel 30 is completed.

Table continues on the next page...

ADC_CEOCFR0 field descriptions (continued)

Field	Description
2 EOCCH29	EOC Channel 29. When set, the measure of channel 29 is completed.
3 EOCCH28	EOC Channel 28. When set, the measure of channel 28 is completed.
4 EOCCH27	EOC Channel 27. When set, the measure of channel 27 is completed.
5 EOCCH26	EOC Channel 26. When set, the measure of channel 26 is completed.
6 EOCCH25	EOC Channel 25. When set, the measure of channel 25 is completed.
7 EOCCH24	EOC Channel 24. When set, the measure of channel 24 is completed.
8 EOCCH23	EOC Channel 23. When set, the measure of channel 23 is completed.
9 EOCCH22	EOC Channel 22. When set, the measure of channel 22 is completed.
10 EOCCH21	EOC Channel 21. When set, the measure of channel 21 is completed.
11 EOCCH20	EOC Channel 20. When set, the measure of channel 20 is completed.
12 EOCCH19	EOC Channel 19. When set, the measure of channel 19 is completed.
13 EOCCH18	EOC Channel 18. When set, the measure of channel 18 is completed.
14 EOCCH17	EOC Channel 17. When set, the measure of channel 17 is completed.
15 EOCCH16	EOC Channel 16. When set, the measure of channel 16 is completed.
16 EOCCH15	EOC Channel 15. When set, the measure of channel 15 is completed.
17 EOCCH14	EOC Channel 14. When set, the measure of channel 14 is completed.
18 EOCCH13	EOC Channel 13. When set, the measure of channel 13 is completed.
19 EOCCH12	EOC Channel 12.

Table continues on the next page...

ADC_CEOCFR0 field descriptions (continued)

Field	Description
	When set, the measure of channel 12 is completed.
20 EOCCH11	EOC Channel 11. When set, the measure of channel 11 is completed.
21 EOCCH10	EOC Channel 10. When set, the measure of channel 10 is completed.
22 EOCCH9	EOC Channel 9. When set, the measure of channel 9 is completed.
23 EOCCH8	EOC Channel 8. When set, the measure of channel 8 is completed.
24 EOCCH7	EOC Channel 7. When set, the measure of channel 7 is completed.
25 EOCCH6	EOC Channel 6. When set, the measure of channel 6 is completed.
26 EOCCH5	EOC Channel 5. When set, the measure of channel 5 is completed.
27 EOCCH4	EOC Channel 4. When set, the measure of channel 4 is completed.
28 EOCCH3	EOC Channel 3. When set, the measure of channel 3 is completed.
29 EOCCH2	EOC Channel 2. When set, the measure of channel 2 is completed.
30 EOCCH1	EOC Channel 1. When set, the measure of channel 1 is completed.
31 EOCCH0	EOC Channel 0. When set, the measure of channel 0 is completed.

35.24.5 Channel Pending register 1 (ADC_CEOCFR1)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOCCH47	EOCCH46	EOCCH45	EOCCH44	EOCCH43	EOCCH42	EOCCH41	EOCCH40	EOCCH39	EOCCH38	EOCCH37	EOCCH36	EOCCH35	EOCCH34	EOCCH33	EOCCH32
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CEOCFR1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 EOCCH47	EOC Channel 47. When set, the measure of channel 47 is completed.
17 EOCCH46	EOC Channel 46. When set, the measure of channel 46 is completed.
18 EOCCH45	EOC Channel 45. When set, the measure of channel 45 is completed.

Table continues on the next page...

ADC_CEOCFR1 field descriptions (continued)

Field	Description
19 EOCCH44	EOC Channel 44. When set, the measure of channel 44 is completed.
20 EOCCH43	EOC Channel 43. When set, the measure of channel 43 is completed.
21 EOCCH42	EOC Channel 42. When set, the measure of channel 42 is completed.
22 EOCCH41	EOC Channel 41. When set, the measure of channel 41 is completed.
23 EOCCH40	EOC Channel 40. When set, the measure of channel 40 is completed.
24 EOCCH39	EOC Channel 39. When set, the measure of channel 39 is completed.
25 EOCCH38	EOC Channel 38. When set, the measure of channel 38 is completed.
26 EOCCH37	EOC Channel 37. When set, the measure of channel 37 is completed.
27 EOCCH36	EOC Channel 36. When set, the measure of channel 36 is completed.
28 EOCCH35	EOC Channel 35. When set, the measure of channel 35 is completed.
29 EOCCH34	EOC Channel 34. When set, the measure of channel 34 is completed.
30 EOCCH33	EOC Channel 33. When set, the measure of channel 33 is completed.
31 EOCCH32	EOC Channel 32. When set, the measure of channel 32 is completed.

35.24.6 Channel Pending register 2 (ADC_CEOCFR2)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EOCCH95	EOCCH94	EOCCH93	EOCCH92	EOCCH91	EOCCH90	EOCCH89	EOCCH88	EOCCH87	EOCCH86	EOCCH85	EOCCH84	EOCCH83	EOCCH82	EOCCH81	EOCCH80
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOCCH79	EOCCH78	EOCCH77	EOCCH76	EOCCH75	EOCCH74	EOCCH73	EOCCH72	EOCCH71	EOCCH70	EOCCH69	EOCCH68	EOCCH67	EOCCH66	EOCCH65	EOCCH64
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CEOCFR2 field descriptions

Field	Description
0 EOCCH95	EOC Channel 95. When set, the measure of channel 95 is completed.
1 EOCCH94	EOC Channel 94. When set, the measure of channel 94 is completed.
2 EOCCH93	EOC Channel 93. When set, the measure of channel 93 is completed.
3 EOCCH92	EOC Channel 92. When set, the measure of channel 92 is completed.

Table continues on the next page...

ADC_CEOCFR2 field descriptions (continued)

Field	Description
4 EOCCH91	EOC Channel 91. When set, the measure of channel 91 is completed.
5 EOCCH90	EOC Channel 90. When set, the measure of channel 90 is completed.
6 EOCCH89	EOC Channel 89. When set, the measure of channel 89 is completed.
7 EOCCH88	EOC Channel 88. When set, the measure of channel 88 is completed.
8 EOCCH87	EOC Channel 87. When set, the measure of channel 87 is completed.
9 EOCCH86	EOC Channel 86. When set, the measure of channel 86 is completed.
10 EOCCH85	EOC Channel 85. When set, the measure of channel 85 is completed.
11 EOCCH84	EOC Channel 84. When set, the measure of channel 84 is completed.
12 EOCCH83	EOC Channel 83. When set, the measure of channel 83 is completed.
13 EOCCH82	EOC Channel 82. When set, the measure of channel 82 is completed.
14 EOCCH81	EOC Channel 81. When set, the measure of channel 81 is completed.
15 EOCCH80	EOC Channel 80. When set, the measure of channel 80 is completed.
16 EOCCH79	EOC Channel 79. When set, the measure of channel 79 is completed.
17 EOCCH78	EOC Channel 78. When set, the measure of channel 78 is completed.
18 EOCCH77	EOC Channel 77. When set, the measure of channel 77 is completed.
19 EOCCH76	EOC Channel 76. When set, the measure of channel 76 is completed.
20 EOCCH75	EOC Channel 75. When set, the measure of channel 75 is completed.
21 EOCCH74	EOC Channel 74.

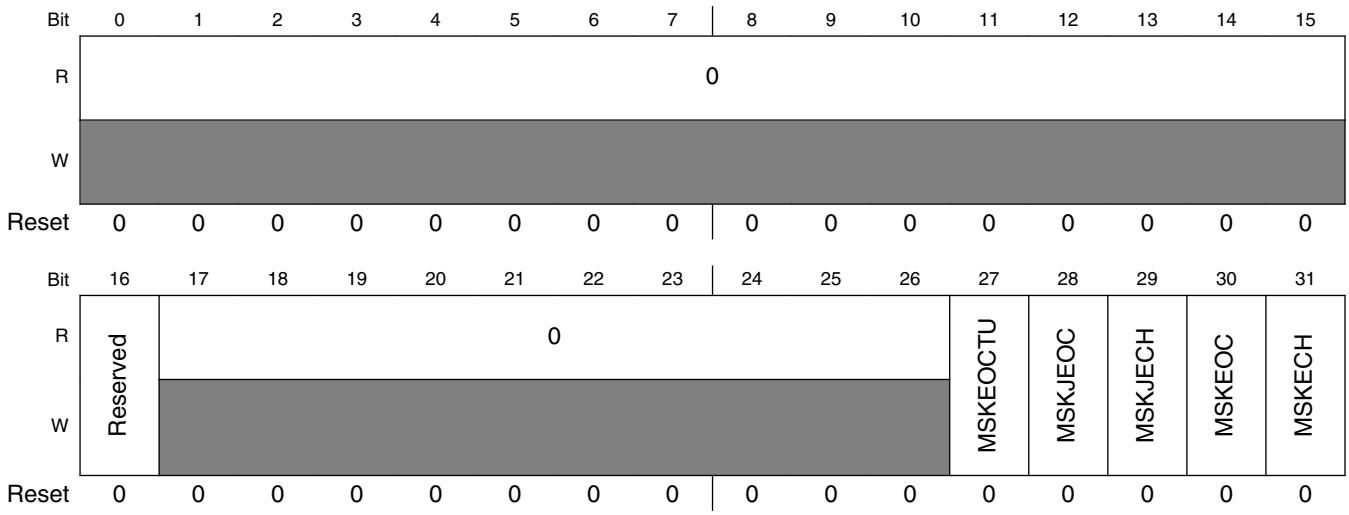
Table continues on the next page...

ADC_CEOCFR2 field descriptions (continued)

Field	Description
	When set, the measure of channel 74 is completed.
22 EOCCH73	EOC Channel 73. When set, the measure of channel 73 is completed.
23 EOCCH72	EOC Channel 72. When set, the measure of channel 72 is completed.
24 EOCCH71	EOC Channel 71. When set, the measure of channel 71 is completed.
25 EOCCH70	EOC Channel 70. When set, the measure of channel 70 is completed.
26 EOCCH69	EOC Channel 69. When set, the measure of channel 69 is completed.
27 EOCCH68	EOC Channel 68. When set, the measure of channel 68 is completed.
28 EOCCH67	EOC Channel 67. When set, the measure of channel 67 is completed.
29 EOCCH66	EOC Channel 66. When set, the measure of channel 66 is completed.
30 EOCCH65	EOC Channel 65. When set, the measure of channel 65 is completed.
31 EOCCH64	EOC Channel 64. When set, the measure of channel 64 is completed.

35.24.7 Interrupt Mask Register (ADC_IMR)

Address: 0h base + 20h offset = 20h



ADC_IMR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 Reserved	This field is reserved. Reserved
17–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 MSKEOCTU	Mask bit for EOCTU. When set, the interrupt is enabled.
28 MSKJEOC	Mask bit for JEOC. When set, the interrupt is enabled.
29 MSKJECH	Mask bit for JECH. When set, the interrupt is enabled.
30 MSKEOC	Mask bit for EOC. When set, the interrupt is enabled.
31 MSKECH	Mask bit for ECH. When set, the interrupt is enabled.

35.24.8 Channel Interrupt Mask Register 0 (ADC_CIMR0)

The fields in this register globally enable or disable end of conversion interrupts by channels within the range of Channel 0 through 31.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific details to find which channels are applicable.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CIM31	CIM30	CIM29	CIM28	CIM27	CIM26	CIM25	CIM24	CIM23	CIM22	CIM21	CIM20	CIM19	CIM18	CIM17	CIM16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CIM15	CIM14	CIM13	CIM12	CIM11	CIM10	CIM9	CIM8	CIM7	CIM6	CIM5	CIM4	CIM3	CIM2	CIM1	CIM0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CIMR0 field descriptions

Field	Description
0 CIM31	Channel 31 interrupt enable. When this field is set, the interrupt for channel 31 is enabled.
1 CIM30	Channel 30 interrupt enable. When this field is set, the interrupt for channel 30 is enabled.
2 CIM29	Channel 29 interrupt enable. When this field is set, the interrupt for channel 29 is enabled.
3 CIM28	Channel 28 interrupt enable. When this field is set, the interrupt for channel 28 is enabled.
4 CIM27	Channel 27 interrupt enable. When this field is set, the interrupt for channel 27 is enabled.
5 CIM26	Channel 26 interrupt enable. When this field is set, the interrupt for channel 26 is enabled.
6 CIM25	Channel 25 interrupt enable. When this field is set, the interrupt for channel 25 is enabled.
7 CIM24	Channel 24 interrupt enable. When this field is set, the interrupt for channel 24 is enabled.
8 CIM23	Channel 23 interrupt enable. When this field is set, the interrupt for channel 23 is enabled.

Table continues on the next page...

ADC_CIMR0 field descriptions (continued)

Field	Description
9 CIM22	Channel 22 interrupt enable. When this field is set, the interrupt for channel 22 is enabled.
10 CIM21	Channel 21 interrupt enable. When this field is set, the interrupt for channel 21 is enabled.
11 CIM20	Channel 20 interrupt enable. When this field is set, the interrupt for channel 20 is enabled.
12 CIM19	Channel 19 interrupt enable. When this field is set, the interrupt for channel 19 is enabled.
13 CIM18	Channel 18 interrupt enable. When this field is set, the interrupt for channel 18 is enabled.
14 CIM17	Channel 17 interrupt enable. When this field is set, the interrupt for channel 17 is enabled.
15 CIM16	Channel 16 interrupt enable. When this field is set, the interrupt for channel 16 is enabled.
16 CIM15	Channel 15 interrupt enable. When this field is set, the interrupt for channel 15 is enabled.
17 CIM14	Channel 14 interrupt enable. When this field is set, the interrupt for channel 14 is enabled.
18 CIM13	Channel 13 interrupt enable. When this field is set, the interrupt for channel 13 is enabled.
19 CIM12	Channel 12 interrupt enable. When this field is set, the interrupt for channel 12 is enabled.
20 CIM11	Channel 11 interrupt enable. When this field is set, the interrupt for channel 11 is enabled.
21 CIM10	Channel 10 interrupt enable. When this field is set, the interrupt for channel 10 is enabled.
22 CIM9	Channel 9 interrupt enable. When this field is set, the interrupt for channel 9 is enabled.
23 CIM8	Channel 8 interrupt enable. When this field is set, the interrupt for channel 8 is enabled.
24 CIM7	Channel 7 interrupt enable. When this field is set, the interrupt for channel 7 is enabled.
25 CIM6	Channel 6 interrupt enable. When this field is set, the interrupt for channel 6 is enabled.
26 CIM5	Channel 5 interrupt enable.

Table continues on the next page...

ADC_CIMR0 field descriptions (continued)

Field	Description
	When this field is set, the interrupt for channel 5 is enabled.
27 CIM4	Channel 4 interrupt enable. When this field is set, the interrupt for channel 4 is enabled.
28 CIM3	Channel 3 interrupt enable. When this field is set, the interrupt for channel 3 is enabled.
29 CIM2	Channel 2 interrupt enable. When this field is set, the interrupt for channel 2 is enabled.
30 CIM1	Channel 1 interrupt enable. When this field is set, the interrupt for channel 1 is enabled.
31 CIM0	Channel 0 interrupt enable. When this field is set, the interrupt for channel 0 is enabled.

35.24.9 Channel Interrupt Mask Register 1 (ADC_CIMR1)

The fields in this register globally enable or disable end of conversion interrupts by channels within the range of Channel 32 through 63.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific details to find which channels are applicable.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CIM47	CIM46	CIM45	CIM44	CIM43	CIM42	CIM41	CIM40	CIM39	CIM38	CIM37	CIM36	CIM35	CIM34	CIM33	CIM32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CIMR1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 CIM47	Channel 47 interrupt enable. When this field is set, the interrupt for channel 47 is enabled.
17 CIM46	Channel 46 interrupt enable. When this field is set, the interrupt for channel 46 is enabled.
18 CIM45	Channel 45 interrupt enable. When this field is set, the interrupt for channel 45 is enabled.
19 CIM44	Channel 44 interrupt enable. When this field is set, the interrupt for channel 44 is enabled.
20 CIM43	Channel 43 interrupt enable. When this field is set, the interrupt for channel 43 is enabled.
21 CIM42	Channel 42 interrupt enable. When this field is set, the interrupt for channel 42 is enabled.
22 CIM41	Channel 41 interrupt enable. When this field is set, the interrupt for channel 41 is enabled.
23 CIM40	Channel 40 interrupt enable. When this field is set, the interrupt for channel 40 is enabled.
24 CIM39	Channel 39 interrupt enable. When this field is set, the interrupt for channel 39 is enabled.
25 CIM38	Channel 38 interrupt enable. When this field is set, the interrupt for channel 38 is enabled.
26 CIM37	Channel 37 interrupt enable. When this field is set, the interrupt for channel 37 is enabled.
27 CIM36	Channel 36 interrupt enable. When this field is set, the interrupt for channel 36 is enabled.
28 CIM35	Channel 35 interrupt enable. When this field is set, the interrupt for channel 35 is enabled.
29 CIM34	Channel 34 interrupt enable. When this field is set, the interrupt for channel 34 is enabled.
30 CIM33	Channel 33 interrupt enable. When this field is set, the interrupt for channel 33 is enabled.
31 CIM32	Channel 32 interrupt enable. When this field is set, the interrupt for channel 32 is enabled.

35.24.10 Channel Interrupt Mask Register 2 (ADC_CIMR2)

The fields in this register globally enable or disable end of conversion interrupts by channels within the range of Channel 64 through 95.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific details to find which channels are applicable.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CIM95	CIM94	CIM93	CIM92	CIM91	CIM90	CIM89	CIM88	CIM87	CIM86	CIM85	CIM84	CIM83	CIM82	CIM81	CIM80
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CIM79	CIM78	CIM77	CIM76	CIM75	CIM74	CIM73	CIM72	CIM71	CIM70	CIM69	CIM68	CIM67	CIM66	CIM65	CIM64
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CIMR2 field descriptions

Field	Description
0 CIM95	Channel 95 interrupt enable. When this field is set, the interrupt for channel 95 is enabled.
1 CIM94	Channel 94 interrupt enable. When this field is set, the interrupt for channel 94 is enabled.
2 CIM93	Channel 93 interrupt enable. When this field is set, the interrupt for channel 93 is enabled.
3 CIM92	Channel 92 interrupt enable. When this field is set, the interrupt for channel 92 is enabled.
4 CIM91	Channel 91 interrupt enable. When this field is set, the interrupt for channel 91 is enabled.
5 CIM90	Channel 90 interrupt enable. When this field is set, the interrupt for channel 90 is enabled.
6 CIM89	Channel 89 interrupt enable. When this field is set, the interrupt for channel 89 is enabled.

Table continues on the next page...

ADC_CIMR2 field descriptions (continued)

Field	Description
7 CIM88	Channel 88 interrupt enable. When this field is set, the interrupt for channel 88 is enabled.
8 CIM87	Channel 87 interrupt enable. When this field is set, the interrupt for channel 87 is enabled.
9 CIM86	Channel 86 interrupt enable. When this field is set, the interrupt for channel 86 is enabled.
10 CIM85	Channel 85 interrupt enable. When this field is set, the interrupt for channel 85 is enabled.
11 CIM84	Channel 84 interrupt enable. When this field is set, the interrupt for channel 84 is enabled.
12 CIM83	Channel 83 interrupt enable. When this field is set, the interrupt for channel 83 is enabled.
13 CIM82	Channel 82 interrupt enable. When this field is set, the interrupt for channel 82 is enabled.
14 CIM81	Channel 81 interrupt enable. When this field is set, the interrupt for channel 81 is enabled.
15 CIM80	Channel 80 interrupt enable. When this field is set, the interrupt for channel 80 is enabled.
16 CIM79	Channel 79 interrupt enable. When this field is set, the interrupt for channel 79 is enabled.
17 CIM78	Channel 78 interrupt enable. When this field is set, the interrupt for channel 78 is enabled.
18 CIM77	Channel 77 interrupt enable. When this field is set, the interrupt for channel 77 is enabled.
19 CIM76	Channel 76 interrupt enable. When this field is set, the interrupt for channel 76 is enabled.
20 CIM75	Channel 75 interrupt enable. When this field is set, the interrupt for channel 75 is enabled.
21 CIM74	Channel 74 interrupt enable. When this field is set, the interrupt for channel 74 is enabled.
22 CIM73	Channel 73 interrupt enable. When this field is set, the interrupt for channel 73 is enabled.
23 CIM72	Channel 72 interrupt enable. When this field is set, the interrupt for channel 72 is enabled.
24 CIM71	Channel 71 interrupt enable.

Table continues on the next page...

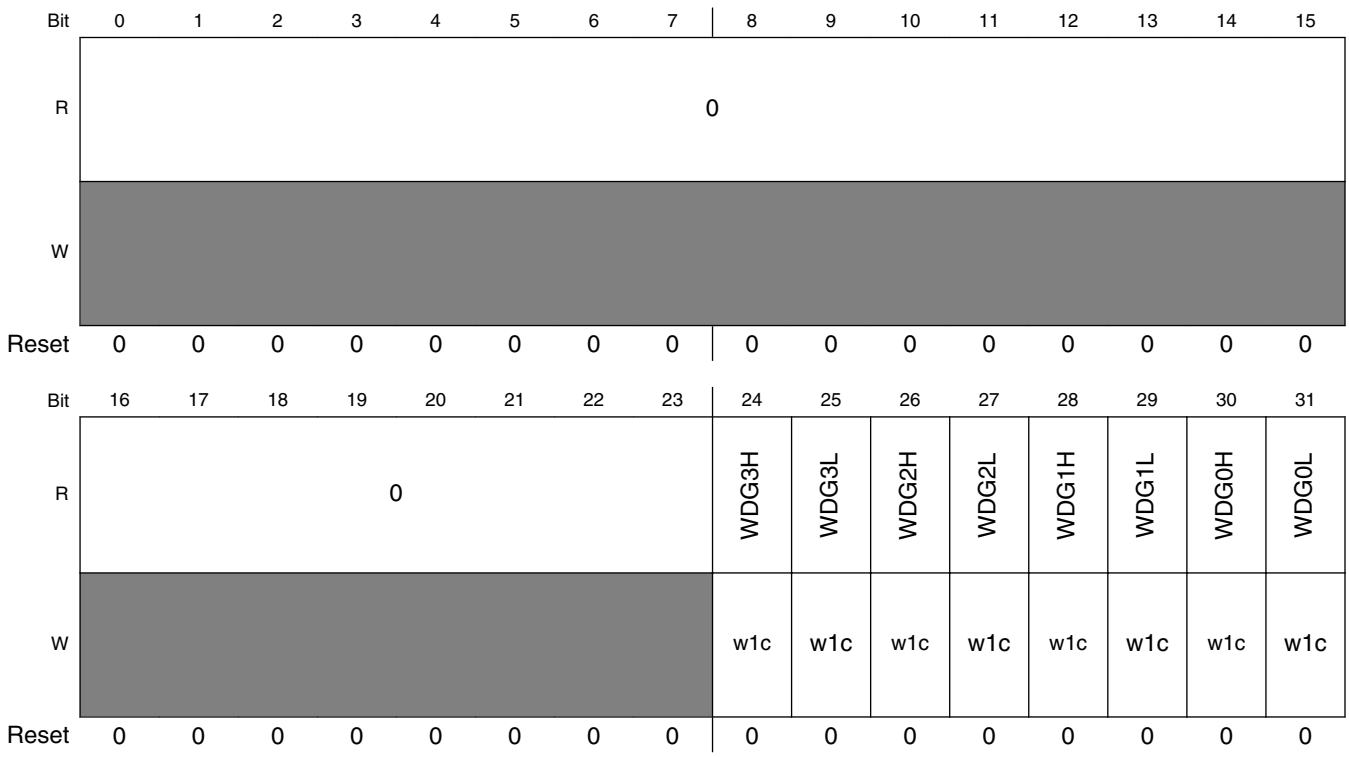
ADC_CIMR2 field descriptions (continued)

Field	Description
	When this field is set, the interrupt for channel 71 is enabled.
25 CIM70	Channel 70 interrupt enable. When this field is set, the interrupt for channel 70 is enabled.
26 CIM69	Channel 69 interrupt enable. When this field is set, the interrupt for channel 69 is enabled.
27 CIM68	Channel 68 interrupt enable. When this field is set, the interrupt for channel 68 is enabled.
28 CIM67	Channel 67 interrupt enable. When this field is set, the interrupt for channel 67 is enabled.
29 CIM66	Channel 66 interrupt enable. When this field is set, the interrupt for channel 66 is enabled.
30 CIM65	Channel 65 interrupt enable. When this field is set, the interrupt for channel 65 is enabled.
31 CIM64	Channel 64 interrupt enable. When this field is set, the interrupt for channel 64 is enabled.

35.24.11 Watchdog Threshold Interrupt Status Register (ADC_WTISR)

The fields in this register, when set, indicate either the upper or lower threshold value has been crossed for a specific watchdog.

Address: 0h base + 30h offset = 30h



ADC_WTISR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 WDG3H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 3 (WD3). **NOTE:** Write a '1' to this field to clear it.
25 WDG3L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 3 (WD3). **NOTE:** Write a '1' to this field to clear it.
26 WDG2H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 2 (WD2). **NOTE:** Write a '1' to this field to clear it.
27 WDG2L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 2 (WD2).

Table continues on the next page...

ADC_WTISR field descriptions (continued)

Field	Description
	NOTE: Write a '1' to this field to clear it.
28 WDG1H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 1 (WD1). **NOTE:** Write a '1' to this field to clear it.
29 WDG1L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 1 (WD1). **NOTE:** Write a '1' to this field to clear it.
30 WDG0H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 0 (WD0). **NOTE:** Write a '1' to this field to clear it.
31 WDG0L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 0 (WD0). **NOTE:** Write a '1' to this field to clear it.

35.24.12 Watchdog Threshold Interrupt Mask Register (ADC_WTIMR)

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_WTIMR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 MSKWDG3H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 3 (WD3) being higher than the programmed higher threshold. When set, the interrupt is enabled.

Table continues on the next page...

ADC_WTIMR field descriptions (continued)

Field	Description
25 MSKWDG3L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 3 (WD3) being lower than the programmed lower threshold. When set, the interrupt is enabled.
26 MSKWDG2H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 2 (WD2) being higher than the programmed higher threshold. When set, the interrupt is enabled.
27 MSKWDG2L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 2 (WD2) being lower than the programmed lower threshold. When set, the interrupt is enabled.
28 MSKWDG1H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 1 (WD1) being higher than the programmed higher threshold. When set, the interrupt is enabled.
29 MSKWDG1L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 1 (WD1) being lower than the programmed lower threshold. When set, the interrupt is enabled.
30 MSKWDG0H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 0 (WD0) being higher than the programmed higher threshold. When set, the interrupt is enabled.
31 MSKWDG0L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 0 (WD0) being lower than the programmed lower threshold. When set, the interrupt is enabled.

35.24.13 DMA Enable register (ADC_DMAE)

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0														DCLR	DMAEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_DMAE field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

ADC_DMAE field descriptions (continued)

Field	Description
30 DCLR	DMA Clear sequence enable. 0 DMA request cleared by Acknowledge from DMA controller 1 DMA request cleared on read of data registers **NOTE:** When DCLR is set, ADC DMA channels should only execute read accesses on ADC_CDR*n* registers.
31 DMAEN	DMA global enable. When this bit is set, the DMA feature is enabled.

35.24.14 DMA Channel Select Register 0 (ADC_DMAR0)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DMA31	DMA30	DMA29	DMA28	DMA27	DMA26	DMA25	DMA24	DMA23	DMA22	DMA21	DMA20	DMA19	DMA18	DMA17	DMA16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DMA15	DMA14	DMA13	DMA12	DMA11	DMA10	DMA9	DMA8	DMA7	DMA6	DMA5	DMA4	DMA3	DMA2	DMA1	DMA0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_DMAR0 field descriptions

Field	Description
0 DMA31	DMA enable for channel 31. When set, transferring data in DMA mode is enabled for channel 31.
1 DMA30	DMA enable for channel 30. When set, transferring data in DMA mode is enabled for channel 30.

Table continues on the next page...

ADC_DMAR0 field descriptions (continued)

Field	Description
2 DMA29	DMA enable for channel 29. When set, transferring data in DMA mode is enabled for channel 29.
3 DMA28	DMA enable for channel 28. When set, transferring data in DMA mode is enabled for channel 28.
4 DMA27	DMA enable for channel 27. When set, transferring data in DMA mode is enabled for channel 27.
5 DMA26	DMA enable for channel 26. When set, transferring data in DMA mode is enabled for channel 26.
6 DMA25	DMA enable for channel 25. When set, transferring data in DMA mode is enabled for channel 25.
7 DMA24	DMA enable for channel 24. When set, transferring data in DMA mode is enabled for channel 24.
8 DMA23	DMA enable for channel 23. When set, transferring data in DMA mode is enabled for channel 23.
9 DMA22	DMA enable for channel 22. When set, transferring data in DMA mode is enabled for channel 22.
10 DMA21	DMA enable for channel 21. When set, transferring data in DMA mode is enabled for channel 21.
11 DMA20	DMA enable for channel 20. When set, transferring data in DMA mode is enabled for channel 20.
12 DMA19	DMA enable for channel 19. When set, transferring data in DMA mode is enabled for channel 19.
13 DMA18	DMA enable for channel 18. When set, transferring data in DMA mode is enabled for channel 18.
14 DMA17	DMA enable for channel 17. When set, transferring data in DMA mode is enabled for channel 17.
15 DMA16	DMA enable for channel 16. When set, transferring data in DMA mode is enabled for channel 16.
16 DMA15	DMA enable for channel 15. When set, transferring data in DMA mode is enabled for channel 15.
17 DMA14	DMA enable for channel 14. When set, transferring data in DMA mode is enabled for channel 14.
18 DMA13	DMA enable for channel 13. When set, transferring data in DMA mode is enabled for channel 13.
19 DMA12	DMA enable for channel 12.

Table continues on the next page...

ADC_DMAR0 field descriptions (continued)

Field	Description
	When set, transferring data in DMA mode is enabled for channel 12.
20 DMA11	DMA enable for channel 11. When set, transferring data in DMA mode is enabled for channel 11.
21 DMA10	DMA enable for channel 10. When set, transferring data in DMA mode is enabled for channel 10.
22 DMA9	DMA enable for channel 9. When set, transferring data in DMA mode is enabled for channel 9.
23 DMA8	DMA enable for channel 8. When set, transferring data in DMA mode is enabled for channel 8.
24 DMA7	DMA enable for channel 7. When set, transferring data in DMA mode is enabled for channel 7.
25 DMA6	DMA enable for channel 6. When set, transferring data in DMA mode is enabled for channel 6.
26 DMA5	DMA enable for channel 5. When set, transferring data in DMA mode is enabled for channel 5.
27 DMA4	DMA enable for channel 4. When set, transferring data in DMA mode is enabled for channel 4.
28 DMA3	DMA enable for channel 3. When set, transferring data in DMA mode is enabled for channel 3.
29 DMA2	DMA enable for channel 2. When set, transferring data in DMA mode is enabled for channel 2.
30 DMA1	DMA enable for channel 1. When set, transferring data in DMA mode is enabled for channel 1.
31 DMA0	DMA enable for channel 0. When set, transferring data in DMA mode is enabled for channel 0.

35.24.15 DMA Channel Select Register 1 (ADC_DMAR1)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DMA47	DMA46	DMA45	DMA44	DMA43	DMA42	DMA41	DMA40	DMA39	DMA38	DMA37	DMA36	DMA35	DMA34	DMA33	DMA32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_DMAR1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 DMA47	DMA enable for channel 47. When set, transferring data in DMA mode is enabled for channel 47.
17 DMA46	DMA enable for channel 46. When set, transferring data in DMA mode is enabled for channel 46.
18 DMA45	DMA enable for channel 45. When set, transferring data in DMA mode is enabled for channel 45.
19 DMA44	DMA enable for channel 44. When set, transferring data in DMA mode is enabled for channel 44.
20 DMA43	DMA enable for channel 43. When set, transferring data in DMA mode is enabled for channel 43.
21 DMA42	DMA enable for channel 42. When set, transferring data in DMA mode is enabled for channel 42.
22 DMA41	DMA enable for channel 41.

Table continues on the next page...

ADC_DMAR1 field descriptions (continued)

Field	Description
	When set, transferring data in DMA mode is enabled for channel 41.
23 DMA40	DMA enable for channel 40. When set, transferring data in DMA mode is enabled for channel 40.
24 DMA39	DMA enable for channel 39. When set, transferring data in DMA mode is enabled for channel 39.
25 DMA38	DMA enable for channel 38. When set, transferring data in DMA mode is enabled for channel 38.
26 DMA37	DMA enable for channel 37. When set, transferring data in DMA mode is enabled for channel 37.
27 DMA36	DMA enable for channel 36. When set, transferring data in DMA mode is enabled for channel 36.
28 DMA35	DMA enable for channel 35. When set, transferring data in DMA mode is enabled for channel 35.
29 DMA34	DMA enable for channel 34. When set, transferring data in DMA mode is enabled for channel 34.
30 DMA33	DMA enable for channel 33. When set, transferring data in DMA mode is enabled for channel 33.
31 DMA32	DMA enable for channel 32. When set, transferring data in DMA mode is enabled for channel 32.

35.24.16 DMA Channel Select Register 2 (ADC_DMAR2)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 4Ch offset = 4Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DMA95	DMA94	DMA93	DMA92	DMA91	DMA90	DMA89	DMA88	DMA87	DMA86	DMA85	DMA84	DMA83	DMA82	DMA81	DMA80
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DMA79	DMA78	DMA77	DMA76	DMA75	DMA74	DMA73	DMA72	DMA71	DMA70	DMA69	DMA68	DMA67	DMA66	DMA65	DMA64
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_DMAR2 field descriptions

Field	Description
0 DMA95	DMA enable for channel 95. When set, transferring data in DMA mode is enabled for channel 95.
1 DMA94	DMA enable for channel 94. When set, transferring data in DMA mode is enabled for channel 94.
2 DMA93	DMA enable for channel 93. When set, transferring data in DMA mode is enabled for channel 93.
3 DMA92	DMA enable for channel 92. When set, transferring data in DMA mode is enabled for channel 92.
4 DMA91	DMA enable for channel 91. When set, transferring data in DMA mode is enabled for channel 91.
5 DMA90	DMA enable for channel 90. When set, transferring data in DMA mode is enabled for channel 90.
6 DMA89	DMA enable for channel 89. When set, transferring data in DMA mode is enabled for channel 89.

Table continues on the next page...

ADC_DMAR2 field descriptions (continued)

Field	Description
7 DMA88	DMA enable for channel 88. When set, transferring data in DMA mode is enabled for channel 88.
8 DMA87	DMA enable for channel 87. When set, transferring data in DMA mode is enabled for channel 87.
9 DMA86	DMA enable for channel 86. When set, transferring data in DMA mode is enabled for channel 86.
10 DMA85	DMA enable for channel 85. When set, transferring data in DMA mode is enabled for channel 85.
11 DMA84	DMA enable for channel 84. When set, transferring data in DMA mode is enabled for channel 84.
12 DMA83	DMA enable for channel 83. When set, transferring data in DMA mode is enabled for channel 83.
13 DMA82	DMA enable for channel 82. When set, transferring data in DMA mode is enabled for channel 82.
14 DMA81	DMA enable for channel 81. When set, transferring data in DMA mode is enabled for channel 81.
15 DMA80	DMA enable for channel 80. When set, transferring data in DMA mode is enabled for channel 80.
16 DMA79	DMA enable for channel 79. When set, transferring data in DMA mode is enabled for channel 79.
17 DMA78	DMA enable for channel 78. When set, transferring data in DMA mode is enabled for channel 78.
18 DMA77	DMA enable for channel 77. When set, transferring data in DMA mode is enabled for channel 77.
19 DMA76	DMA enable for channel 76. When set, transferring data in DMA mode is enabled for channel 76.
20 DMA75	DMA enable for channel 75. When set, transferring data in DMA mode is enabled for channel 75.
21 DMA74	DMA enable for channel 74. When set, transferring data in DMA mode is enabled for channel 74.
22 DMA73	DMA enable for channel 73. When set, transferring data in DMA mode is enabled for channel 73.
23 DMA72	DMA enable for channel 72. When set, transferring data in DMA mode is enabled for channel 72.
24 DMA71	DMA enable for channel 71.

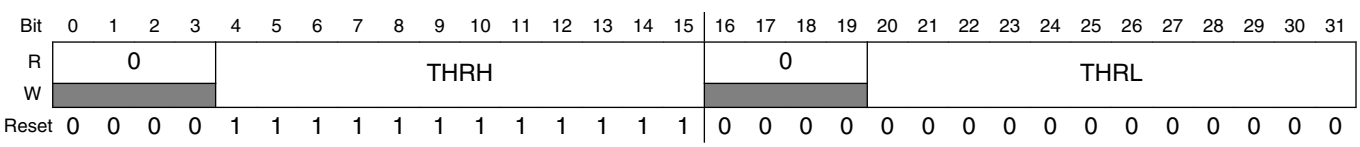
Table continues on the next page...

ADC_DMAR2 field descriptions (continued)

Field	Description
	When set, transferring data in DMA mode is enabled for channel 71.
25 DMA70	DMA enable for channel 70. When set, transferring data in DMA mode is enabled for channel 70.
26 DMA69	DMA enable for channel 69. When set, transferring data in DMA mode is enabled for channel 69.
27 DMA68	DMA enable for channel 68. When set, transferring data in DMA mode is enabled for channel 68.
28 DMA67	DMA enable for channel 67. When set, transferring data in DMA mode is enabled for channel 67.
29 DMA66	DMA enable for channel 66. When set, transferring data in DMA mode is enabled for channel 66.
30 DMA65	DMA enable for channel 65. When set, transferring data in DMA mode is enabled for channel 65.
31 DMA64	DMA enable for channel 64. When set, transferring data in DMA mode is enabled for channel 64.

35.24.17 Threshold Register (ADC_THRHLR*n*)

Address: 0h base + 60h offset + (4d × i), where i=0d to 3d



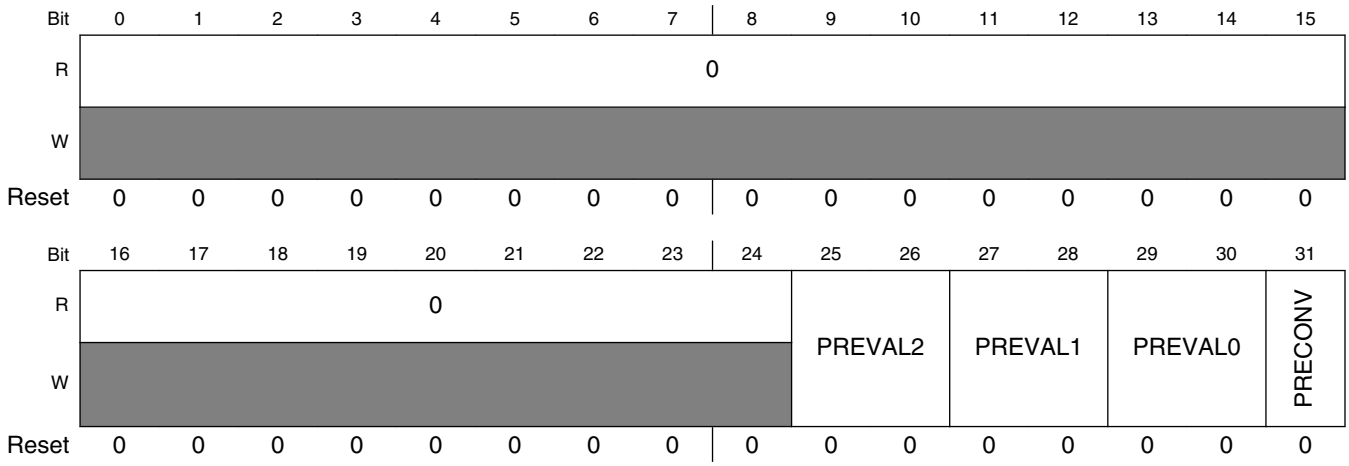
ADC_THRHLR*n* field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 THRH	High threshold value for watchdog n.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value for watchdog n.

35.24.18 Presampling Control Register (ADC_PSCR)

See the chip configuration details for the availability of this register.

Address: 0h base + 80h offset = 80h



ADC_PSCR field descriptions

Field	Description
0–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–26 PREVAL2	Internal voltage selection for Presampling Selects analog input voltage for presampling from the available four internal voltages for External channels (channel 64 to 95). Refer to Presampling channel enable.
27–28 PREVAL1	Internal voltage selection for Presampling. These two bits select analog input voltage for presampling from the available internal voltages using internal extended channels (channel 32 to 63). This is used for temperature sensor also. Refer to Presampling channel enable.
29–30 PREVAL0	Internal voltage selection for Presampling. These two bits select analog input voltage for presampling from the available four internal voltages for Internal precision channels (Channel 0 to 31). Refer to Presampling channel enable.
31 PRECONV	Convert Presampled value If bit PRECONV is set, presampling is followed by the conversion. Sampling will be bypassed and conversion of presampled data will be done. This bit has no effect on conversion of a channel when presampling on that channel is disabled (for examplei.e PSR*x*[PRES*y*] = 0 , where *x* = register index and *y* = channel number).

35.24.19 Presampling register 0 (ADC_PSR0)

See the chip configuration details for the availability of this register.

Address: 0h base + 84h offset = 84h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PRES31	PRES30	PRES29	PRES28	PRES27	PRES26	PRES25	PRES24	PRES23	PRES22	PRES21	PRES20	PRES19	PRES18	PRES17	PRES16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PRES15	PRES14	PRES13	PRES12	PRES11	PRES10	PRES9	PRES8	PRES7	PRES6	PRES5	PRES4	PRES3	PRES2	PRES1	PRES0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_PSR0 field descriptions

Field	Description
0 PRES31	Presampling enable for channel 31. When set, presampling is enabled for channel 31.
1 PRES30	Presampling enable for channel 30. When set, presampling is enabled for channel 30.
2 PRES29	Presampling enable for channel 29. When set, presampling is enabled for channel 29.
3 PRES28	Presampling enable for channel 28. When set, presampling is enabled for channel 28.
4 PRES27	Presampling enable for channel 27. When set, presampling is enabled for channel 27.
5 PRES26	Presampling enable for channel 26. When set, presampling is enabled for channel 26.
6 PRES25	Presampling enable for channel 25. When set, presampling is enabled for channel 25.
7 PRES24	Presampling enable for channel 24. When set, presampling is enabled for channel 24.
8 PRES23	Presampling enable for channel 23.

Table continues on the next page...

ADC_PSR0 field descriptions (continued)

Field	Description
	When set, presampling is enabled for channel 23.
9 PRES22	Presampling enable for channel 22. When set, presampling is enabled for channel 22.
10 PRES21	Presampling enable for channel 21. When set, presampling is enabled for channel 21.
11 PRES20	Presampling enable for channel 20. When set, presampling is enabled for channel 20.
12 PRES19	Presampling enable for channel 19. When set, presampling is enabled for channel 19.
13 PRES18	Presampling enable for channel 18. When set, presampling is enabled for channel 18.
14 PRES17	Presampling enable for channel 17. When set, presampling is enabled for channel 17.
15 PRES16	Presampling enable for channel 16. When set, presampling is enabled for channel 16.
16 PRES15	Presampling enable for channel 15. When set, presampling is enabled for channel 15.
17 PRES14	Presampling enable for channel 14. When set, presampling is enabled for channel 14.
18 PRES13	Presampling enable for channel 13. When set, presampling is enabled for channel 13.
19 PRES12	Presampling enable for channel 12. When set, presampling is enabled for channel 12.
20 PRES11	Presampling enable for channel 11. When set, presampling is enabled for channel 11.
21 PRES10	Presampling enable for channel 10. When set, presampling is enabled for channel 10.
22 PRES9	Presampling enable for channel 9. When set, presampling is enabled for channel 9.
23 PRES8	Presampling enable for channel 8. When set, presampling is enabled for channel 8.
24 PRES7	Presampling enable for channel 7. When set, presampling is enabled for channel 7.
25 PRES6	Presampling enable for channel 6. When set, presampling is enabled for channel 6.

Table continues on the next page...

ADC_PSR0 field descriptions (continued)

Field	Description
26 PRES5	Presampling enable for channel 5. When set, presampling is enabled for channel 5.
27 PRES4	Presampling enable for channel 4. When set, presampling is enabled for channel 4.
28 PRES3	Presampling enable for channel 3. When set, presampling is enabled for channel 3.
29 PRES2	Presampling enable for channel 2. When set, presampling is enabled for channel 2.
30 PRES1	Presampling enable for channel 1. When set, presampling is enabled for channel 1.
31 PRES0	Presampling enable for channel 0. When set, presampling is enabled for channel 0.

35.24.20 Presampling register 1 (ADC_PSR1)

Address: 0h base + 88h offset = 88h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PRES47	PRES46	PRES45	PRES44	PRES43	PRES42	PRES41	PRES40	PRES39	PRES38	PRES37	PRES36	PRES35	PRES34	PRES33	PRES32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_PSR1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 PRES47	Presampling enable for channel 47. When set, presampling is enabled for channel 47.
17 PRES46	Presampling enable for channel 46. When set, presampling is enabled for channel 46.

Table continues on the next page...

ADC_PSR1 field descriptions (continued)

Field	Description
18 PRES45	Presampling enable for channel 45. When set, presampling is enabled for channel 45.
19 PRES44	Presampling enable for channel 44. When set, presampling is enabled for channel 44.
20 PRES43	Presampling enable for channel 43. When set, presampling is enabled for channel 43.
21 PRES42	Presampling enable for channel 42. When set, presampling is enabled for channel 42.
22 PRES41	Presampling enable for channel 41. When set, presampling is enabled for channel 41.
23 PRES40	Presampling enable for channel 40. When set, presampling is enabled for channel 40.
24 PRES39	Presampling enable for channel 39. When set, presampling is enabled for channel 39.
25 PRES38	Presampling enable for channel 38. When set, presampling is enabled for channel 38.
26 PRES37	Presampling enable for channel 37. When set, presampling is enabled for channel 37.
27 PRES36	Presampling enable for channel 36. When set, presampling is enabled for channel 36.
28 PRES35	Presampling enable for channel 35. When set, presampling is enabled for channel 35.
29 PRES34	Presampling enable for channel 34. When set, presampling is enabled for channel 34.
30 PRES33	Presampling enable for channel 33. When set, presampling is enabled for channel 33.
31 PRES32	Presampling enable for channel 32. When set, presampling is enabled for channel 32.

35.24.21 Presampling register 2 (ADC_PSR2)

Address: 0h base + 8Ch offset = 8Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PRES95	PRES94	PRES93	PRES92	PRES91	PRES90	PRES89	PRES88	PRES87	PRES86	PRES85	PRES84	PRES83	PRES82	PRES81	PRES80
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PRES79	PRES78	PRES77	PRES76	PRES75	PRES74	PRES73	PRES72	PRES71	PRES70	PRES69	PRES68	PRES67	PRES66	PRES65	PRES64
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_PSR2 field descriptions

Field	Description
0 PRES95	Presampling enable for channel 95. When set, presampling is enabled for channel 95.
1 PRES94	Presampling enable for channel 94. When set, presampling is enabled for channel 94.
2 PRES93	Presampling enable for channel 93. When set, presampling is enabled for channel 93.
3 PRES92	Presampling enable for channel 92. When set, presampling is enabled for channel 92.
4 PRES91	Presampling enable for channel 91. When set, presampling is enabled for channel 91.
5 PRES90	Presampling enable for channel 90. When set, presampling is enabled for channel 90.
6 PRES89	Presampling enable for channel 89. When set, presampling is enabled for channel 89.
7 PRES88	Presampling enable for channel 88. When set, presampling is enabled for channel 88.
8 PRES87	Presampling enable for channel 87. When set, presampling is enabled for channel 87.
9 PRES86	Presampling enable for channel 86. When set, presampling is enabled for channel 86.

Table continues on the next page...

ADC_PSR2 field descriptions (continued)

Field	Description
10 PRES85	Presampling enable for channel 85. When set, presampling is enabled for channel 85.
11 PRES84	Presampling enable for channel 84. When set, presampling is enabled for channel 84.
12 PRES83	Presampling enable for channel 83. When set, presampling is enabled for channel 83.
13 PRES82	Presampling enable for channel 82. When set, presampling is enabled for channel 82.
14 PRES81	Presampling enable for channel 81. When set, presampling is enabled for channel 81.
15 PRES80	Presampling enable for channel 80. When set, presampling is enabled for channel 80.
16 PRES79	Presampling enable for channel 79. When set, presampling is enabled for channel 79.
17 PRES78	Presampling enable for channel 78. When set, presampling is enabled for channel 78.
18 PRES77	Presampling enable for channel 77. When set, presampling is enabled for channel 77.
19 PRES76	Presampling enable for channel 76. When set, presampling is enabled for channel 76.
20 PRES75	Presampling enable for channel 75. When set, presampling is enabled for channel 75.
21 PRES74	Presampling enable for channel 74. When set, presampling is enabled for channel 74.
22 PRES73	Presampling enable for channel 73. When set, presampling is enabled for channel 73.
23 PRES72	Presampling enable for channel 72. When set, presampling is enabled for channel 72.
24 PRES71	Presampling enable for channel 71. When set, presampling is enabled for channel 71.
25 PRES70	Presampling enable for channel 70. When set, presampling is enabled for channel 70.
26 PRES69	Presampling enable for channel 69. When set, presampling is enabled for channel 69.
27 PRES68	Presampling enable for channel 68.

Table continues on the next page...

ADC_PSR2 field descriptions (continued)

Field	Description
	When set, presampling is enabled for channel 68.
28 PRES67	Presampling enable for channel 67. When set, presampling is enabled for channel 67.
29 PRES66	Presampling enable for channel 66. When set, presampling is enabled for channel 66.
30 PRES65	Presampling enable for channel 65. When set, presampling is enabled for channel 65.
31 PRES64	Presampling enable for channel 64. When set, presampling is enabled for channel 64.

35.24.22 Conversion Timing Register 0 (ADC_CTR0)

Conversion timing register. CTR0 is associated with channels 0 to 31.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 94h offset = 94h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved	0	Reserved		0	Reserved		0	INPSAMP							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0

ADC_CTR0 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 Reserved	This field is reserved.

Table continues on the next page...

ADC_CTR0 field descriptions (continued)

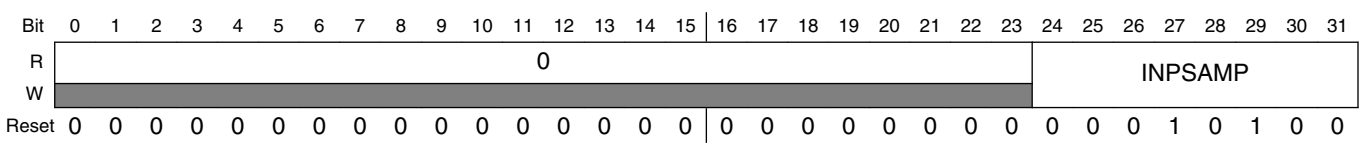
Field	Description
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–19 Reserved	This field is reserved.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–22 Reserved	This field is reserved.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 INPSAMP	Configuration of sampling phase duration. This value tells the sample period duration in terms of the SAR Controller operating clock. The minimum acceptable value is 8. Setting a lower value automatically takes 8 cycles. **NOTE:** The Bandgap voltage sampling time is controlled by CTR0[INPSAMP]. CTR0[INPSAMP] is also used for the external channel sample time. When the user software needs to sample the Bandgap voltage, it must change the CTR0[INPSAMP] value to allow for a longer time while the Bandgap is being sampled, then change its value back to the normal value for sampling external channels.

35.24.23 Conversion Timing Register 1 (ADC_CTR1)

CTR1 is associated with the temperature sensor and channels 32 to 63.

Please see the chip-specific configuration information as all channels may not be available in all ADC instances.

Address: 0h base + 98h offset = 98h



ADC_CTR1 field descriptions

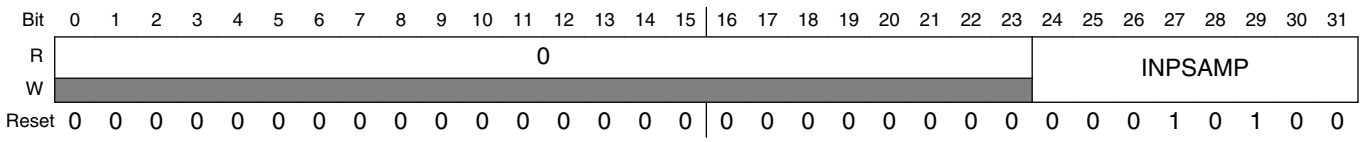
Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 INPSAMP	Configuration of sampling phase duration. This value tells the sample period duration in terms of the SAR Controller operating clock. The minimum acceptable value is 8. Setting a lower value automatically takes 8 cycles.

35.24.24 Conversion Timing Register 2 (ADC_CTR2)

CTR2 is associated to channels 64 to 95.

Please see the chip-specific configuration information as all channels may not be available in all ADC instances.

Address: 0h base + 9Ch offset = 9Ch



ADC_CTR2 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 INPSAMP	Configuration of sampling phase duration. This value tells the sample period duration in terms of the SAR Controller operating clock. The minimum acceptable value is 8. Setting a lower value automatically takes 8 cycles.

35.24.25 Normal Conversion Mask Register 0 (ADC_NCMR0)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + A4h offset = A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_NCMR0 field descriptions

Field	Description
0 CH31	Normal sampling enable for channel 31. When set, normal sampling is enabled for channel 31.
1 CH30	Normal sampling enable for channel 30. When set, normal sampling is enabled for channel 30.
2 CH29	Normal sampling enable for channel 29. When set, normal sampling is enabled for channel 29.
3 CH28	Normal sampling enable for channel 28. When set, normal sampling is enabled for channel 28.
4 CH27	Normal sampling enable for channel 27. When set, normal sampling is enabled for channel 27.
5 CH26	Normal sampling enable for channel 26. When set, normal sampling is enabled for channel 26.
6 CH25	Normal sampling enable for channel 25. When set, normal sampling is enabled for channel 25.

Table continues on the next page...

ADC_NCMR0 field descriptions (continued)

Field	Description
7 CH24	Normal sampling enable for channel 24. When set, normal sampling is enabled for channel 24.
8 CH23	Normal sampling enable for channel 23. When set, normal sampling is enabled for channel 23.
9 CH22	Normal sampling enable for channel 22. When set, normal sampling is enabled for channel 22.
10 CH21	Normal sampling enable for channel 21. When set, normal sampling is enabled for channel 21.
11 CH20	Normal sampling enable for channel 20. When set, normal sampling is enabled for channel 20.
12 CH19	Normal sampling enable for channel 19. When set, normal sampling is enabled for channel 19.
13 CH18	Normal sampling enable for channel 18. When set, normal sampling is enabled for channel 18.
14 CH17	Normal sampling enable for channel 17. When set, normal sampling is enabled for channel 17.
15 CH16	Normal sampling enable for channel 16. When set, normal sampling is enabled for channel 16.
16 CH15	Normal sampling enable for channel 15. When set, normal sampling is enabled for channel 15.
17 CH14	Normal sampling enable for channel 14. When set, normal sampling is enabled for channel 14.
18 CH13	Normal sampling enable for channel 13. When set, normal sampling is enabled for channel 13.
19 CH12	Normal sampling enable for channel 12. When set, normal sampling is enabled for channel 12.
20 CH11	Normal sampling enable for channel 11. When set, normal sampling is enabled for channel 11.
21 CH10	Normal sampling enable for channel 10. When set, normal sampling is enabled for channel 10.
22 CH9	Normal sampling enable for channel 9. When set, normal sampling is enabled for channel 9.
23 CH8	Normal sampling enable for channel 8. When set, normal sampling is enabled for channel 8.
24 CH7	Normal sampling enable for channel 7.

Table continues on the next page...

ADC_NCMR0 field descriptions (continued)

Field	Description
	When set, normal sampling is enabled for channel 7.
25 CH6	Normal sampling enable for channel 6. When set, normal sampling is enabled for channel 6.
26 CH5	Normal sampling enable for channel 5. When set, normal sampling is enabled for channel 5.
27 CH4	Normal sampling enable for channel 4. When set, normal sampling is enabled for channel 4.
28 CH3	Normal sampling enable for channel 3. When set, normal sampling is enabled for channel 3.
29 CH2	Normal sampling enable for channel 2. When set, normal sampling is enabled for channel 2.
30 CH1	Normal sampling enable for channel 1. When set, normal sampling is enabled for channel 1.
31 CH0	Normal sampling enable for channel 0. When set, normal sampling is enabled for channel 0.

35.24.26 Normal Conversion Mask Register 1 (ADC_NCMR1)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + A8h offset = A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CH47	CH46	CH45	CH44	CH43	CH42	CH41	CH40	CH39	CH38	CH37	CH36	CH35	CH34	CH33	CH32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_NCMR1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 CH47	Normal sampling enable for channel 47. When set, normal sampling is enabled for channel 47.
17 CH46	Normal sampling enable for channel 46. When set, normal sampling is enabled for channel 46.
18 CH45	Normal sampling enable for channel 45. When set, normal sampling is enabled for channel 45.
19 CH44	Normal sampling enable for channel 44. When set, normal sampling is enabled for channel 44.
20 CH43	Normal sampling enable for channel 43. When set, normal sampling is enabled for channel 43.
21 CH42	Normal sampling enable for channel 42. When set, normal sampling is enabled for channel 42.
22 CH41	Normal sampling enable for channel 41. When set, normal sampling is enabled for channel 41.
23 CH40	Normal sampling enable for channel 40. When set, normal sampling is enabled for channel 40.
24 CH39	Normal sampling enable for channel 39. When set, normal sampling is enabled for channel 39.
25 CH38	Normal sampling enable for channel 38. When set, normal sampling is enabled for channel 38.
26 CH37	Normal sampling enable for channel 37. When set, normal sampling is enabled for channel 37.
27 CH36	Normal sampling enable for channel 36. When set, normal sampling is enabled for channel 36.
28 CH35	Normal sampling enable for channel 35. When set, normal sampling is enabled for channel 35.
29 CH34	Normal sampling enable for channel 34. When set, normal sampling is enabled for channel 34.
30 CH33	Normal sampling enable for channel 33. When set, normal sampling is enabled for channel 33.
31 CH32	Normal sampling enable for channel 32. When set, normal sampling is enabled for channel 32.

35.24.27 Normal Conversion Mask Register 2 (ADC_NCMR2)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + ACh offset = ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CH95	CH94	CH93	CH92	CH91	CH90	CH89	CH88	CH87	CH86	CH85	CH84	CH83	CH82	CH81	CH80
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CH79	CH78	CH77	CH76	CH75	CH74	CH73	CH72	CH71	CH70	CH69	CH68	CH67	CH66	CH65	CH64
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_NCMR2 field descriptions

Field	Description
0 CH95	Normal sampling enable for channel 95. When set, normal sampling is enabled for channel 95.
1 CH94	Normal sampling enable for channel 94. When set, normal sampling is enabled for channel 94.
2 CH93	Normal sampling enable for channel 93. When set, normal sampling is enabled for channel 93.
3 CH92	Normal sampling enable for channel 92. When set, normal sampling is enabled for channel 92.
4 CH91	Normal sampling enable for channel 91. When set, normal sampling is enabled for channel 91.
5 CH90	Normal sampling enable for channel 90. When set, normal sampling is enabled for channel 90.
6 CH89	Normal sampling enable for channel 89. When set, normal sampling is enabled for channel 89.

Table continues on the next page...

ADC_NCMR2 field descriptions (continued)

Field	Description
7 CH88	Normal sampling enable for channel 88. When set, normal sampling is enabled for channel 88.
8 CH87	Normal sampling enable for channel 87. When set, normal sampling is enabled for channel 87.
9 CH86	Normal sampling enable for channel 86. When set, normal sampling is enabled for channel 86.
10 CH85	Normal sampling enable for channel 85. When set, normal sampling is enabled for channel 85.
11 CH84	Normal sampling enable for channel 84. When set, normal sampling is enabled for channel 84.
12 CH83	Normal sampling enable for channel 83. When set, normal sampling is enabled for channel 83.
13 CH82	Normal sampling enable for channel 82. When set, normal sampling is enabled for channel 82.
14 CH81	Normal sampling enable for channel 81. When set, normal sampling is enabled for channel 81.
15 CH80	Normal sampling enable for channel 80. When set, normal sampling is enabled for channel 80.
16 CH79	Normal sampling enable for channel 79. When set, normal sampling is enabled for channel 79.
17 CH78	Normal sampling enable for channel 78. When set, normal sampling is enabled for channel 78.
18 CH77	Normal sampling enable for channel 77. When set, normal sampling is enabled for channel 77.
19 CH76	Normal sampling enable for channel 76. When set, normal sampling is enabled for channel 76.
20 CH75	Normal sampling enable for channel 75. When set, normal sampling is enabled for channel 75.
21 CH74	Normal sampling enable for channel 74. When set, normal sampling is enabled for channel 74.
22 CH73	Normal sampling enable for channel 73. When set, normal sampling is enabled for channel 73.
23 CH72	Normal sampling enable for channel 72. When set, normal sampling is enabled for channel 72.
24 CH71	Normal sampling enable for channel 71.

Table continues on the next page...

ADC_NCMR2 field descriptions (continued)

Field	Description
	When set, normal sampling is enabled for channel 71.
25 CH70	Normal sampling enable for channel 70. When set, normal sampling is enabled for channel 70.
26 CH69	Normal sampling enable for channel 69. When set, normal sampling is enabled for channel 69.
27 CH68	Normal sampling enable for channel 68. When set, normal sampling is enabled for channel 68.
28 CH67	Normal sampling enable for channel 67. When set, normal sampling is enabled for channel 67.
29 CH66	Normal sampling enable for channel 66. When set, normal sampling is enabled for channel 66.
30 CH65	Normal sampling enable for channel 65. When set, normal sampling is enabled for channel 65.
31 CH64	Normal sampling enable for channel 64. When set, normal sampling is enabled for channel 64.

35.24.28 Injected Conversion Mask Register 0 (ADC_JCMR0)

Enable bits of Injected Sampling for channel 0 to 31.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + B4h offset = B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_JCMR0 field descriptions

Field	Description
0 CH31	Injected sampling enable for channel 31. When set, injected sampling is enabled for channel 31.
1 CH30	Injected sampling enable for channel 30. When set, injected sampling is enabled for channel 30.
2 CH29	Injected sampling enable for channel 29. When set, injected sampling is enabled for channel 29.
3 CH28	Injected sampling enable for channel 28. When set, injected sampling is enabled for channel 28.
4 CH27	Injected sampling enable for channel 27. When set, injected sampling is enabled for channel 27.
5 CH26	Injected sampling enable for channel 26. When set, injected sampling is enabled for channel 26.
6 CH25	Injected sampling enable for channel 25. When set, injected sampling is enabled for channel 25.
7 CH24	Injected sampling enable for channel 24. When set, injected sampling is enabled for channel 24.
8 CH23	Injected sampling enable for channel 23. When set, injected sampling is enabled for channel 23.
9 CH22	Injected sampling enable for channel 22. When set, injected sampling is enabled for channel 22.
10 CH21	Injected sampling enable for channel 21. When set, injected sampling is enabled for channel 21.
11 CH20	Injected sampling enable for channel 20. When set, injected sampling is enabled for channel 20.
12 CH19	Injected sampling enable for channel 19. When set, injected sampling is enabled for channel 19.
13 CH18	Injected sampling enable for channel 18. When set, injected sampling is enabled for channel 18.
14 CH17	Injected sampling enable for channel 17. When set, injected sampling is enabled for channel 17.
15 CH16	Injected sampling enable for channel 16. When set, injected sampling is enabled for channel 16.
16 CH15	Injected sampling enable for channel 15. When set, injected sampling is enabled for channel 15.
17 CH14	Injected sampling enable for channel 14.

Table continues on the next page...

ADC_JCMR0 field descriptions (continued)

Field	Description
	When set, injected sampling is enabled for channel 14.
18 CH13	Injected sampling enable for channel 13. When set, injected sampling is enabled for channel 13.
19 CH12	Injected sampling enable for channel 12. When set, injected sampling is enabled for channel 12.
20 CH11	Injected sampling enable for channel 11. When set, injected sampling is enabled for channel 11.
21 CH10	Injected sampling enable for channel 10. When set, injected sampling is enabled for channel 10.
22 CH9	Injected sampling enable for channel 9. When set, injected sampling is enabled for channel 9.
23 CH8	Injected sampling enable for channel 8. When set, injected sampling is enabled for channel 8.
24 CH7	Injected sampling enable for channel 7. When set, injected sampling is enabled for channel 7.
25 CH6	Injected sampling enable for channel 6. When set, injected sampling is enabled for channel 6.
26 CH5	Injected sampling enable for channel 5. When set, injected sampling is enabled for channel 5.
27 CH4	Injected sampling enable for channel 4. When set, injected sampling is enabled for channel 4.
28 CH3	Injected sampling enable for channel 3. When set, injected sampling is enabled for channel 3.
29 CH2	Injected sampling enable for channel 2. When set, injected sampling is enabled for channel 2.
30 CH1	Injected sampling enable for channel 1. When set, injected sampling is enabled for channel 1.
31 CH0	Injected sampling enable for channel 0. When set, injected sampling is enabled for channel 0.

35.24.29 Injected Conversion Mask Register 1 (ADC_JCMR1)

Enable bits of Injected Sampling for channel 32 to 63.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + B8h offset = B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CH47	CH46	CH45	CH44	CH43	CH42	CH41	CH40	CH39	CH38	CH37	CH36	CH35	CH34	CH33	CH32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_JCMR1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 CH47	Injected sampling enable for channel 47. When set, injected sampling is enabled for channel 47.
17 CH46	Injected sampling enable for channel 46. When set, injected sampling is enabled for channel 46.
18 CH45	Injected sampling enable for channel 45. When set, injected sampling is enabled for channel 45.
19 CH44	Injected sampling enable for channel 44. When set, injected sampling is enabled for channel 44.
20 CH43	Injected sampling enable for channel 43. When set, injected sampling is enabled for channel 43.
21 CH42	Injected sampling enable for channel 42. When set, injected sampling is enabled for channel 42.
22 CH41	Injected sampling enable for channel 41. When set, injected sampling is enabled for channel 41.
23 CH40	Injected sampling enable for channel 40. When set, injected sampling is enabled for channel 40.
24 CH39	Injected sampling enable for channel 39. When set, injected sampling is enabled for channel 39.

Table continues on the next page...

ADC_JCMR1 field descriptions (continued)

Field	Description
25 CH38	Injected sampling enable for channel 38. When set, injected sampling is enabled for channel 38.
26 CH37	Injected sampling enable for channel 37. When set, injected sampling is enabled for channel 37.
27 CH36	Injected sampling enable for channel 36. When set, injected sampling is enabled for channel 36.
28 CH35	Injected sampling enable for channel 35. When set, injected sampling is enabled for channel 35.
29 CH34	Injected sampling enable for channel 34. When set, injected sampling is enabled for channel 34.
30 CH33	Injected sampling enable for channel 33. When set, injected sampling is enabled for channel 33.
31 CH32	Injected sampling enable for channel 32. When set, injected sampling is enabled for channel 32.

35.24.30 Injected Conversion Mask Register 2 (ADC_JCMR2)

Enable bits of Injected Sampling for channel 64 to 95.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + BCh offset = BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CH95	CH94	CH93	CH92	CH91	CH90	CH89	CH88	CH87	CH86	CH85	CH84	CH83	CH82	CH81	CH80
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CH79	CH78	CH77	CH76	CH75	CH74	CH73	CH72	CH71	CH70	CH69	CH68	CH67	CH66	CH65	CH64
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_JCMR2 field descriptions

Field	Description
0 CH95	Injected sampling enable for channel 95. When set, injected sampling is enabled for channel 95.
1 CH94	Injected sampling enable for channel 94. When set, injected sampling is enabled for channel 94.
2 CH93	Injected sampling enable for channel 93. When set, injected sampling is enabled for channel 93.
3 CH92	Injected sampling enable for channel 92. When set, injected sampling is enabled for channel 92.
4 CH91	Injected sampling enable for channel 91. When set, injected sampling is enabled for channel 91.
5 CH90	Injected sampling enable for channel 90. When set, injected sampling is enabled for channel 90.
6 CH89	Injected sampling enable for channel 89. When set, injected sampling is enabled for channel 89.
7 CH88	Injected sampling enable for channel 88. When set, injected sampling is enabled for channel 88.
8 CH87	Injected sampling enable for channel 87. When set, injected sampling is enabled for channel 87.
9 CH86	Injected sampling enable for channel 86. When set, injected sampling is enabled for channel 86.
10 CH85	Injected sampling enable for channel 85. When set, injected sampling is enabled for channel 85.
11 CH84	Injected sampling enable for channel 84. When set, injected sampling is enabled for channel 84.
12 CH83	Injected sampling enable for channel 83. When set, injected sampling is enabled for channel 83.
13 CH82	Injected sampling enable for channel 82. When set, injected sampling is enabled for channel 82.
14 CH81	Injected sampling enable for channel 81. When set, injected sampling is enabled for channel 81.
15 CH80	Injected sampling enable for channel 80. When set, injected sampling is enabled for channel 80.
16 CH79	Injected sampling enable for channel 79. When set, injected sampling is enabled for channel 79.
17 CH78	Injected sampling enable for channel 78.

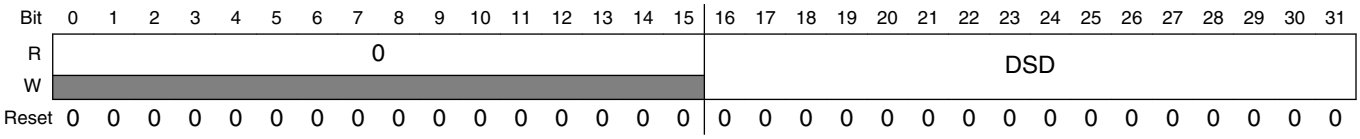
Table continues on the next page...

ADC_JCMR2 field descriptions (continued)

Field	Description
	When set, injected sampling is enabled for channel 78.
18 CH77	Injected sampling enable for channel 77. When set, injected sampling is enabled for channel 77.
19 CH76	Injected sampling enable for channel 76. When set, injected sampling is enabled for channel 76.
20 CH75	Injected sampling enable for channel 75. When set, injected sampling is enabled for channel 75.
21 CH74	Injected sampling enable for channel 74. When set, injected sampling is enabled for channel 74.
22 CH73	Injected sampling enable for channel 73. When set, injected sampling is enabled for channel 73.
23 CH72	Injected sampling enable for channel 72. When set, injected sampling is enabled for channel 72.
24 CH71	Injected sampling enable for channel 71. When set, injected sampling is enabled for channel 71.
25 CH70	Injected sampling enable for channel 70. When set, injected sampling is enabled for channel 70.
26 CH69	Injected sampling enable for channel 69. When set, injected sampling is enabled for channel 69.
27 CH68	Injected sampling enable for channel 68. When set, injected sampling is enabled for channel 68.
28 CH67	Injected sampling enable for channel 67. When set, injected sampling is enabled for channel 67.
29 CH66	Injected sampling enable for channel 66. When set, injected sampling is enabled for channel 66.
30 CH65	Injected sampling enable for channel 65. When set, injected sampling is enabled for channel 65.
31 CH64	Injected sampling enable for channel 64. When set, injected sampling is enabled for channel 64.

35.24.31 Decode Signals Delay Register (ADC_DSDR)

Address: 0h base + C4h offset = C4h

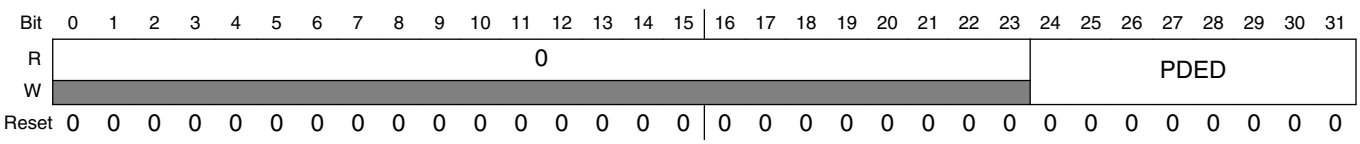


ADC_DSDR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 DSD	The delay between the external decode signals and the beginning of the sampling phase. It is used to take into account the settling time of the external mux. The decode signal delay is calculated as (DSD x 1/ Frequency of System Clock).

35.24.32 Power Down Exit Delay Register (ADC_PDEDR)

Address: 0h base + C8h offset = C8h



ADC_PDEDR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 PDED	The delay between the power down bit reset and the start of conversion. The power down delay is calculated as (PDED x 1/[ADC_clock_frequency]).

35.24.33 Channel Data Register *n* (Precision Channels) (ADC_CDR*n*)

Note

It is important to note that the content of the CDATA field is affected by the setting of the ADC_MCR[WLSIDE] field, which controls whether the data is left aligned or right aligned. When ADC_MCR[WLSIDE]=1 (data to be left aligned), the CDATA field is left aligned from bit 16 to bit (16 + resolution - 1). When ADC_MCR[WLSIDE]=0 (data to be right aligned), the CDATA field is right aligned from bit (32-resolution) to bit 31. All other fields are unaffected.

Address: 0h base + 100h offset + (4d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												VALID	OVERW	RESULT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CDR*n* field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 VALID	Used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read.

Table continues on the next page...

ADC_CDR*n* field descriptions (continued)

Field	Description
13 OVERW	Overwrite data Used to notify when a conversion data is overwritten by a newer result. The new data is written or discarded according to the OWREN bit of MCR register.
14–15 RESULT	These two bits reflect the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode 01 Data is a result of Injected conversion mode 10 Data is a result of BCTU conversion mode 11 Reserved
16–31 CDATA	Converted Data 11:0. **Note:** It is important to note that the content of the CDATA field is affected by the setting of the ADC_MCR[WLSIDE] field, which controls whether the data is left aligned or right aligned. When ADC_MCR[WLSIDE]=1 (data to be left aligned), the CDATA field is left aligned from bit 16 to bit (16 + resolution - 1). When ADC_MCR[WLSIDE]=0 (data to be right aligned), the CDATA field is right aligned from bit (32-resolution) to bit 31. All other fields are unaffected.

35.24.34 Channel Data Register *n* (Internal Channels) (ADC_CDR*n*)

Note

It is important to note that the content of the CDATA field is affected by the setting of the ADC_MCR[WLSIDE] field, which controls whether the data is left aligned or right aligned. When ADC_MCR[WLSIDE]=1 (data to be left aligned), the CDATA field is left aligned from bit 16 to bit (16 + resolution - 1). When ADC_MCR[WLSIDE]=0 (data to be right aligned), the CDATA field is right aligned from bit (32-resolution) to bit 31. All other fields are unaffected.

Address: 0h base + 180h offset + (4d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												VALID	OVERW	RESULT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CDR*n* field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 VALID	Used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read.

Table continues on the next page...

ADC_CDR*n* field descriptions (continued)

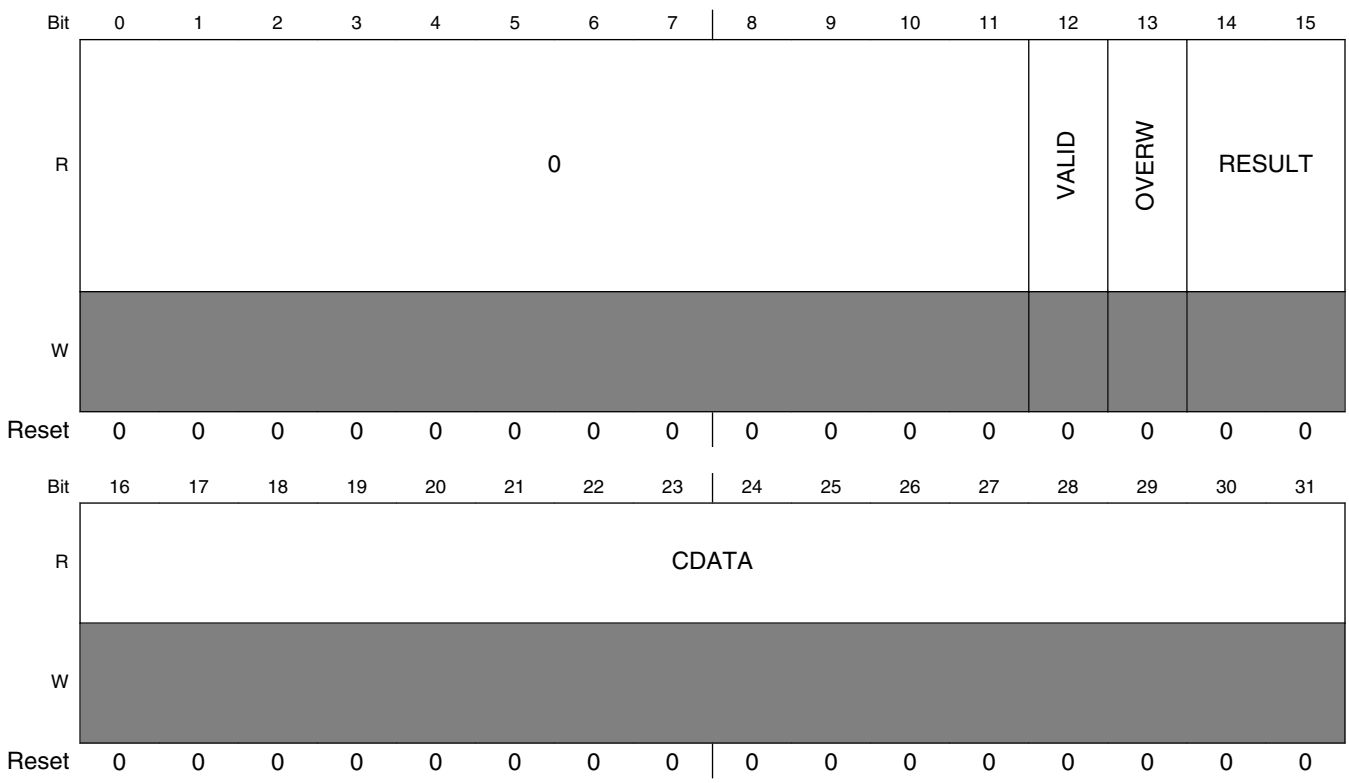
Field	Description
13 OVERW	Overwrite data Used to notify when a conversion data is overwritten by a newer result. The new data is written or discarded according to the OWREN bit of MCR register.
14–15 RESULT	These two bits reflect the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode 01 Data is a result of Injected conversion mode 10 Data is a result of BCTU conversion mode 11 Reserved
16–31 CDATA	Converted Data 11:0. **Note:** It is important to note that the content of the CDATA field is affected by the setting of the ADC_MCR[WLSIDE] field, which controls whether the data is left aligned or right aligned. When ADC_MCR[WLSIDE]=1 (data to be left aligned), the CDATA field is left aligned from bit 16 to bit (16 + resolution - 1). When ADC_MCR[WLSIDE]=0 (data to be right aligned), the CDATA field is right aligned from bit (32-resolution) to bit 31. All other fields are unaffected.

35.24.35 Channel Data Register *n* (External Channels) (ADC_CDR*n*)

Note

It is important to note that the content of the CDATA field is affected by the setting of the ADC_MCR[WLSIDE] field, which controls whether the data is left aligned or right aligned. When ADC_MCR[WLSIDE]=1 (data to be left aligned), the CDATA field is left aligned from bit 16 to bit (16 + resolution - 1). When ADC_MCR[WLSIDE]=0 (data to be right aligned), the CDATA field is right aligned from bit (32-resolution) to bit 31. All other fields are unaffected.

Address: 0h base + 200h offset + (4d × i), where i=0d to 31d



ADC_CDR*n* field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 VALID	Used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read.

Table continues on the next page...

ADC_CDR*n* field descriptions (continued)

Field	Description
13 OVERW	Overwrite data Used to notify when a conversion data is overwritten by a newer result. The new data is written or discarded according to the OWREN bit of MCR register.
14–15 RESULT	These two bits reflect the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode 01 Data is a result of Injected conversion mode 10 Data is a result of BCTU conversion mode 11 Reserved
16–31 CDATA	Converted Data 11:0. **Note:** It is important to note that the content of the CDATA field is affected by the setting of the ADC_MCR[WLSIDE] field, which controls whether the data is left aligned or right aligned. When ADC_MCR[WLSIDE]=1 (data to be left aligned), the CDATA field is left aligned from bit 16 to bit (16 + resolution - 1). When ADC_MCR[WLSIDE]=0 (data to be right aligned), the CDATA field is right aligned from bit (32-resolution) to bit 31. All other fields are unaffected.

35.24.36 Channel Watchdog Select Register 0 (ADC_CWSELR0)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2B0h offset = 2B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH7				WSEL_CH6				WSEL_CH5				WSEL_CH4				WSEL_CH3				WSEL_CH2				WSEL_CH1				WSEL_CH0			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWSELR0 field descriptions

Field	Description
0–3 WSEL_CH7	Channel Watchdog select for channel 7. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected

Table continues on the next page...

ADC_CWSELR0 field descriptions (continued)

Field	Description
	1111 : THRHLR15 register is selected
4–7 WSEL_CH6	Channel Watchdog select for channel 6. See WSEL_CH7.
8–11 WSEL_CH5	Channel Watchdog select for channel 5. See WSEL_CH7.
12–15 WSEL_CH4	Channel Watchdog select for channel 4. See WSEL_CH7.
16–19 WSEL_CH3	Channel Watchdog select for channel 3. See WSEL_CH7.
20–23 WSEL_CH2	Channel Watchdog select for channel 2. See WSEL_CH7.
24–27 WSEL_CH1	Channel Watchdog select for channel 1. See WSEL_CH7.
28–31 WSEL_CH0	Channel Watchdog select for channel 0. See WSEL_CH7.

35.24.37 Channel Watchdog Select Register 1 (ADC_CWSELR1)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2B4h offset = 2B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH15				WSEL_CH14				WSEL_CH13				WSEL_CH12				WSEL_CH11				WSEL_CH10				WSEL_CH9				WSEL_CH8			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWSELR1 field descriptions

Field	Description
0–3 WSEL_CH15	Channel Watchdog select for channel 15. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected

Table continues on the next page...

ADC_CWSELR1 field descriptions (continued)

Field	Description
	0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH14	Channel Watchdog select for channel 14. See WSEL_CH15.
8–11 WSEL_CH13	Channel Watchdog select for channel 13. See WSEL_CH15.
12–15 WSEL_CH12	Channel Watchdog select for channel 12. See WSEL_CH15.
16–19 WSEL_CH11	Channel Watchdog select for channel 11. See WSEL_CH15.
20–23 WSEL_CH10	Channel Watchdog select for channel 10. See WSEL_CH15.
24–27 WSEL_CH9	Channel Watchdog select for channel 9. See WSEL_CH15.
28–31 WSEL_CH8	Channel Watchdog select for channel 8. See WSEL_CH15.

35.24.38 Channel Watchdog Select Register 2 (ADC_CWSELR2)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2B8h offset = 2B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH23				WSEL_CH22				WSEL_CH21				WSEL_CH20				WSEL_CH19				WSEL_CH18				WSEL_CH17				WSEL_CH16			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWSELR2 field descriptions

Field	Description
0–3 WSEL_CH23	Channel Watchdog select for channel 23. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH22	Channel Watchdog select for channel 22. See WSEL_CH23.
8–11 WSEL_CH21	Channel Watchdog select for channel 21. See WSEL_CH23.
12–15 WSEL_CH20	Channel Watchdog select for channel 20. See WSEL_CH23.
16–19 WSEL_CH19	Channel Watchdog select for channel 19. See WSEL_CH23.
20–23 WSEL_CH18	Channel Watchdog select for channel 18. See WSEL_CH23.
24–27 WSEL_CH17	Channel Watchdog select for channel 17. See WSEL_CH23.
28–31 WSEL_CH16	Channel Watchdog select for channel 16. See WSEL_CH23.

35.24.39 Channel Watchdog Select Register 3 (ADC_CWSELR3)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2BCh offset = 2BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH31				WSEL_CH30				WSEL_CH29				WSEL_CH28				WSEL_CH27				WSEL_CH26				WSEL_CH25				WSEL_CH24			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWSELR3 field descriptions

Field	Description
0–3 WSEL_CH31	Channel Watchdog select for channel 31. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH30	Channel Watchdog select for channel 30. See WSEL_CH30.
8–11 WSEL_CH29	Channel Watchdog select for channel 29. See WSEL_CH31.
12–15 WSEL_CH28	Channel Watchdog select for channel 28. See WSEL_CH31.
16–19 WSEL_CH27	Channel Watchdog select for channel 27. See WSEL_CH31.
20–23 WSEL_CH26	Channel Watchdog select for channel 26. See WSEL_CH31.
24–27 WSEL_CH25	Channel Watchdog select for channel 25. See WSEL_CH31.
28–31 WSEL_CH24	Channel Watchdog select for channel 24. See WSEL_CH31.

35.24.40 Channel Watchdog Select Register 4 (ADC_CWSELR4)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2C0h offset = 2C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH39				WSEL_CH38				WSEL_CH37				WSEL_CH36				WSEL_CH35				WSEL_CH34				WSEL_CH33				WSEL_CH32			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWSELR4 field descriptions

Field	Description
0–3 WSEL_CH39	Channel Watchdog select for channel 39. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH38	Channel Watchdog select for channel 38. See WSEL_CH39.
8–11 WSEL_CH37	Channel Watchdog select for channel 37. See WSEL_CH39.
12–15 WSEL_CH36	Channel Watchdog select for channel 36. See WSEL_CH39.
16–19 WSEL_CH35	Channel Watchdog select for channel 35. See WSEL_CH39.
20–23 WSEL_CH34	Channel Watchdog select for channel 34. See WSEL_CH39.
24–27 WSEL_CH33	Channel Watchdog select for channel 33. See WSEL_CH39.
28–31 WSEL_CH32	Channel Watchdog select for channel 32. See WSEL_CH39.

35.24.41 Channel Watchdog Select Register 8 (ADC_CWSELR8)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2D0h offset = 2D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH71				WSEL_CH70				WSEL_CH69				WSEL_CH68				WSEL_CH67				WSEL_CH66				WSEL_CH65				WSEL_CH64			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWSELR8 field descriptions

Field	Description
0–3 WSEL_CH71	Channel Watchdog select for channel 71. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH70	Channel Watchdog select for channel 70. See WSEL_CH71.
8–11 WSEL_CH69	Channel Watchdog select for channel 69. See WSEL_CH71.
12–15 WSEL_CH68	Channel Watchdog select for channel 68. See WSEL_CH71.
16–19 WSEL_CH67	Channel Watchdog select for channel 67. See WSEL_CH71.
20–23 WSEL_CH66	Channel Watchdog select for channel 66. See WSEL_CH71.
24–27 WSEL_CH65	Channel Watchdog select for channel 65. See WSEL_CH71.
28–31 WSEL_CH64	Channel Watchdog select for channel 64. See WSEL_CH71.

35.24.42 Channel Watchdog Select Register 9 (ADC_CWSELR9)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2D4h offset = 2D4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH79				WSEL_CH78				WSEL_CH77				WSEL_CH76				WSEL_CH75				WSEL_CH74				WSEL_CH73				WSEL_CH72			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWSELR9 field descriptions

Field	Description
0–3 WSEL_CH79	Channel Watchdog select for channel 79. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH78	Channel Watchdog select for channel 78. See WSEL_CH79.
8–11 WSEL_CH77	Channel Watchdog select for channel 77. See WSEL_CH79.
12–15 WSEL_CH76	Channel Watchdog select for channel 76. See WSEL_CH79.
16–19 WSEL_CH75	Channel Watchdog select for channel 75. See WSEL_CH79.
20–23 WSEL_CH74	Channel Watchdog select for channel 74. See WSEL_CH79.
24–27 WSEL_CH73	Channel Watchdog select for channel 73. See WSEL_CH79.
28–31 WSEL_CH72	Channel Watchdog select for channel 72. See WSEL_CH79.

35.24.43 Channel Watchdog Select Register 10 (ADC_CWSELR10)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2D8h offset = 2D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH87				WSEL_CH86				WSEL_CH85				WSEL_CH84				WSEL_CH83				WSEL_CH82				WSEL_CH81				WSEL_CH80			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWSELR10 field descriptions

Field	Description
0–3 WSEL_CH87	Channel Watchdog select for channel 87. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH86	Channel Watchdog select for channel 86. See WSEL_CH87.
8–11 WSEL_CH85	Channel Watchdog select for channel 85. See WSEL_CH87.
12–15 WSEL_CH84	Channel Watchdog select for channel 84. See WSEL_CH87.
16–19 WSEL_CH83	Channel Watchdog select for channel 83. See WSEL_CH87.
20–23 WSEL_CH82	Channel Watchdog select for channel 82. See WSEL_CH87.
24–27 WSEL_CH81	Channel Watchdog select for channel 81. See WSEL_CH87.
28–31 WSEL_CH80	Channel Watchdog select for channel 80. See WSEL_CH87.

35.24.44 Channel Watchdog Select Register 11 (ADC_CWSELR11)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2DCh offset = 2DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH95				WSEL_CH94				WSEL_CH93				WSEL_CH92				WSEL_CH91				WSEL_CH90				WSEL_CH89				WSEL_CH88			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWSELR11 field descriptions

Field	Description
0–3 WSEL_CH95	Channel Watchdog select for channel 95. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH94	Channel Watchdog select for channel 94. See WSEL_CH95.
8–11 WSEL_CH93	Channel Watchdog select for channel 93. See WSEL_CH95.
12–15 WSEL_CH92	Channel Watchdog select for channel 92. See WSEL_CH95.
16–19 WSEL_CH91	Channel Watchdog select for channel 91. See WSEL_CH95.
20–23 WSEL_CH90	Channel Watchdog select for channel 90. See WSEL_CH95.
24–27 WSEL_CH89	Channel Watchdog select for channel 89. See WSEL_CH95.
28–31 WSEL_CH88	Channel Watchdog select for channel 88. See WSEL_CH95.

35.24.45 Channel Watchdog Enable Register 0 (ADC_CWENR0)

Controls whether the watchdog is enabled for channels 0 to 31.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2E0h offset = 2E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CWEN31	CWEN30	CWEN29	CWEN28	CWEN27	CWEN26	CWEN25	CWEN24	CWEN23	CWEN22	CWEN21	CWEN20	CWEN19	CWEN18	CWEN17	CWEN16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CWEN15	CWEN14	CWEN13	CWEN12	CWEN11	CWEN10	CWEN9	CWEN8	CWEN7	CWEN6	CWEN5	CWEN4	CWEN3	CWEN2	CWEN1	CWEN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWENR0 field descriptions

Field	Description
0 CWEN31	Watchdog enable for channel 31. When set, Watchdog feature is enabled for channel 31.
1 CWEN30	Watchdog enable for channel 30. When set, Watchdog feature is enabled for channel 30.
2 CWEN29	Watchdog enable for channel 29. When set, Watchdog feature is enabled for channel 29.
3 CWEN28	Watchdog enable for channel 28. When set, Watchdog feature is enabled for channel 28.
4 CWEN27	Watchdog enable for channel 27. When set, Watchdog feature is enabled for channel 27.
5 CWEN26	Watchdog enable for channel 26. When set, Watchdog feature is enabled for channel 26.
6 CWEN25	Watchdog enable for channel 25. When set, Watchdog feature is enabled for channel 25.
7 CWEN24	Watchdog enable for channel 24. When set, Watchdog feature is enabled for channel 24.
8 CWEN23	Watchdog enable for channel 23. When set, Watchdog feature is enabled for channel 23.
9 CWEN22	Watchdog enable for channel 22. When set, Watchdog feature is enabled for channel 22.
10 CWEN21	Watchdog enable for channel 21. When set, Watchdog feature is enabled for channel 21.
11 CWEN20	Watchdog enable for channel 20.

Table continues on the next page...

ADC_CWENR0 field descriptions (continued)

Field	Description
	When set, Watchdog feature is enabled for channel 20.
12 CWEN19	Watchdog enable for channel 19. When set, Watchdog feature is enabled for channel 19.
13 CWEN18	Watchdog enable for channel 18. When set, Watchdog feature is enabled for channel 18.
14 CWEN17	Watchdog enable for channel 17. When set, Watchdog feature is enabled for channel 17.
15 CWEN16	Watchdog enable for channel 16. When set, Watchdog feature is enabled for channel 16.
16 CWEN15	Watchdog enable for channel 15. When set, Watchdog feature is enabled for channel 15.
17 CWEN14	Watchdog enable for channel 14. When set, Watchdog feature is enabled for channel 14.
18 CWEN13	Watchdog enable for channel 13. When set, Watchdog feature is enabled for channel 13.
19 CWEN12	Watchdog enable for channel 12. When set, Watchdog feature is enabled for channel 12.
20 CWEN11	Watchdog enable for channel 11. When set, Watchdog feature is enabled for channel 11.
21 CWEN10	Watchdog enable for channel 10. When set, Watchdog feature is enabled for channel 10.
22 CWEN9	Watchdog enable for channel 9. When set, Watchdog feature is enabled for channel 9.
23 CWEN8	Watchdog enable for channel 8. When set, Watchdog feature is enabled for channel 8.
24 CWEN7	Watchdog enable for channel 7. When set, Watchdog feature is enabled for channel 7.
25 CWEN6	Watchdog enable for channel 6. When set, Watchdog feature is enabled for channel 6.
26 CWEN5	Watchdog enable for channel 5. When set, Watchdog feature is enabled for channel 5.
27 CWEN4	Watchdog enable for channel 4. When set, Watchdog feature is enabled for channel 4.
28 CWEN3	Watchdog enable for channel 3. When set, Watchdog feature is enabled for channel 3.

Table continues on the next page...

ADC_CWENR0 field descriptions (continued)

Field	Description
29 CWEN2	Watchdog enable for channel 2. When set, Watchdog feature is enabled for channel 2.
30 CWEN1	Watchdog enable for channel 1. When set, Watchdog feature is enabled for channel 1.
31 CWEN0	Watchdog enable for channel 0. When set, Watchdog feature is enabled for channel 0.

35.24.46 Channel Watchdog Enable Register 1 (ADC_CWENR1)

Controls whether the watchdog is enabled for channels 32 to 63.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2E4h offset = 2E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CWEN47	CWEN46	CWEN45	CWEN44	CWEN43	CWEN42	CWEN41	CWEN40	CWEN39	CWEN38	CWEN37	CWEN36	CWEN35	CWEN34	CWEN33	CWEN32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWENR1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 CWEN47	Watchdog enable for channel 47. When set, Watchdog feature is enabled for channel 47.
17 CWEN46	Watchdog enable for channel 46. When set, Watchdog feature is enabled for channel 46.

Table continues on the next page...

ADC_CWENR1 field descriptions (continued)

Field	Description
18 CWEN45	Watchdog enable for channel 45. When set, Watchdog feature is enabled for channel 45.
19 CWEN44	Watchdog enable for channel 44. When set, Watchdog feature is enabled for channel 44.
20 CWEN43	Watchdog enable for channel 43. When set, Watchdog feature is enabled for channel 43.
21 CWEN42	Watchdog enable for channel 42. When set, Watchdog feature is enabled for channel 42.
22 CWEN41	Watchdog enable for channel 41. When set, Watchdog feature is enabled for channel 41.
23 CWEN40	Watchdog enable for channel 40. When set, Watchdog feature is enabled for channel 40.
24 CWEN39	Watchdog enable for channel 39. When set, Watchdog feature is enabled for channel 39.
25 CWEN38	Watchdog enable for channel 38. When set, Watchdog feature is enabled for channel 38.
26 CWEN37	Watchdog enable for channel 37. When set, Watchdog feature is enabled for channel 37.
27 CWEN36	Watchdog enable for channel 36. When set, Watchdog feature is enabled for channel 36.
28 CWEN35	Watchdog enable for channel 35. When set, Watchdog feature is enabled for channel 35.
29 CWEN34	Watchdog enable for channel 34. When set, Watchdog feature is enabled for channel 34.
30 CWEN33	Watchdog enable for channel 33. When set, Watchdog feature is enabled for channel 33.
31 CWEN32	Watchdog enable for channel 32. When set, Watchdog feature is enabled for channel 32.

35.24.47 Channel Watchdog Enable Register 2 (ADC_CWENR2)

Controls whether the watchdog is enabled for channels 64 to 95.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2E8h offset = 2E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CWEN95	CWEN94	CWEN93	CWEN92	CWEN91	CWEN90	CWEN89	CWEN88	CWEN87	CWEN86	CWEN85	CWEN84	CWEN83	CWEN82	CWEN81	CWEN80
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CWEN79	CWEN78	CWEN77	CWEN76	CWEN75	CWEN74	CWEN73	CWEN72	CWEN71	CWEN70	CWEN69	CWEN68	CWEN67	CWEN66	CWEN65	CWEN64
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWENR2 field descriptions

Field	Description
0 CWEN95	Watchdog enable for channel 95. When set, Watchdog feature is enabled for channel 95.
1 CWEN94	Watchdog enable for channel 94. When set, Watchdog feature is enabled for channel 94.
2 CWEN93	Watchdog enable for channel 93. When set, Watchdog feature is enabled for channel 93.
3 CWEN92	Watchdog enable for channel 92. When set, Watchdog feature is enabled for channel 92.
4 CWEN91	Watchdog enable for channel 91. When set, Watchdog feature is enabled for channel 91.
5 CWEN90	Watchdog enable for channel 90. When set, Watchdog feature is enabled for channel 90.
6 CWEN89	Watchdog enable for channel 89. When set, Watchdog feature is enabled for channel 89.
7 CWEN88	Watchdog enable for channel 88. When set, Watchdog feature is enabled for channel 88.
8 CWEN87	Watchdog enable for channel 87. When set, Watchdog feature is enabled for channel 87.

Table continues on the next page...

ADC_CWENR2 field descriptions (continued)

Field	Description
9 CWEN86	Watchdog enable for channel 86. When set, Watchdog feature is enabled for channel 86.
10 CWEN85	Watchdog enable for channel 85. When set, Watchdog feature is enabled for channel 85.
11 CWEN84	Watchdog enable for channel 84. When set, Watchdog feature is enabled for channel 84.
12 CWEN83	Watchdog enable for channel 83. When set, Watchdog feature is enabled for channel 83.
13 CWEN82	Watchdog enable for channel 82. When set, Watchdog feature is enabled for channel 82.
14 CWEN81	Watchdog enable for channel 81. When set, Watchdog feature is enabled for channel 81.
15 CWEN80	Watchdog enable for channel 80. When set, Watchdog feature is enabled for channel 80.
16 CWEN79	Watchdog enable for channel 79. When set, Watchdog feature is enabled for channel 79.
17 CWEN78	Watchdog enable for channel 78. When set, Watchdog feature is enabled for channel 78.
18 CWEN77	Watchdog enable for channel 77. When set, Watchdog feature is enabled for channel 77.
19 CWEN76	Watchdog enable for channel 76. When set, Watchdog feature is enabled for channel 76.
20 CWEN75	Watchdog enable for channel 75. When set, Watchdog feature is enabled for channel 75.
21 CWEN74	Watchdog enable for channel 74. When set, Watchdog feature is enabled for channel 74.
22 CWEN73	Watchdog enable for channel 73. When set, Watchdog feature is enabled for channel 73.
23 CWEN72	Watchdog enable for channel 72. When set, Watchdog feature is enabled for channel 72.
24 CWEN71	Watchdog enable for channel 71. When set, Watchdog feature is enabled for channel 71.
25 CWEN70	Watchdog enable for channel 70. When set, Watchdog feature is enabled for channel 70.
26 CWEN69	Watchdog enable for channel 69.

Table continues on the next page...

ADC_CWENR2 field descriptions (continued)

Field	Description
	When set, Watchdog feature is enabled for channel 69.
27 CWEN68	Watchdog enable for channel 68. When set, Watchdog feature is enabled for channel 68.
28 CWEN67	Watchdog enable for channel 67. When set, Watchdog feature is enabled for channel 67.
29 CWEN66	Watchdog enable for channel 66. When set, Watchdog feature is enabled for channel 66.
30 CWEN65	Watchdog enable for channel 65. When set, Watchdog feature is enabled for channel 65.
31 CWEN64	Watchdog enable for channel 64. When set, Watchdog feature is enabled for channel 64.

35.24.48 Analog Watchdog Out of Range Register 0 (ADC_AWORR0)

Analog watchdog out of range register for channels 0 to 31.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2F0h offset = 2F0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AWOR_CH31	AWOR_CH30	AWOR_CH29	AWOR_CH28	AWOR_CH27	AWOR_CH26	AWOR_CH25	AWOR_CH24	AWOR_CH23	AWOR_CH22	AWOR_CH21	AWOR_CH20	AWOR_CH19	AWOR_CH18	AWOR_CH17	AWOR_CH16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AWOR_CH15	AWOR_CH14	AWOR_CH13	AWOR_CH12	AWOR_CH11	AWOR_CH10	AWOR_CH9	AWOR_CH8	AWOR_CH7	AWOR_CH6	AWOR_CH5	AWOR_CH4	AWOR_CH3	AWOR_CH2	AWOR_CH1	AWOR_CH0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_AWORR0 field descriptions

Field	Description
0 AWOR_CH31	Analog watchdog out of range for channel 31. When set, indicates the converted data is out of range for channel 31.
1 AWOR_CH30	Analog watchdog out of range for channel 30. When set, indicates the converted data is out of range for channel 30.
2 AWOR_CH29	Analog watchdog out of range for channel 29. When set, indicates the converted data is out of range for channel 29.
3 AWOR_CH28	Analog watchdog out of range for channel 28. When set, indicates the converted data is out of range for channel 28.
4 AWOR_CH27	Analog watchdog out of range for channel 27. When set, indicates the converted data is out of range for channel 27.
5 AWOR_CH26	Analog watchdog out of range for channel 26. When set, indicates the converted data is out of range for channel 26.
6 AWOR_CH25	Analog watchdog out of range for channel 25. When set, indicates the converted data is out of range for channel 25.
7 AWOR_CH24	Analog watchdog out of range for channel 24. When set, indicates the converted data is out of range for channel 24.
8 AWOR_CH23	Analog watchdog out of range for channel 23. When set, indicates the converted data is out of range for channel 23.
9 AWOR_CH22	Analog watchdog out of range for channel 22. When set, indicates the converted data is out of range for channel 22.
10 AWOR_CH21	Analog watchdog out of range for channel 21. When set, indicates the converted data is out of range for channel 21.
11 AWOR_CH20	Analog watchdog out of range for channel 20. When set, indicates the converted data is out of range for channel 20.
12 AWOR_CH19	Analog watchdog out of range for channel 19.

Table continues on the next page...

ADC_AWORR0 field descriptions (continued)

Field	Description
	When set, indicates the converted data is out of range for channel 19.
13 AWOR_CH18	Analog watchdog out of range for channel 18. When set, indicates the converted data is out of range for channel 18.
14 AWOR_CH17	Analog watchdog out of range for channel 17. When set, indicates the converted data is out of range for channel 17.
15 AWOR_CH16	Analog watchdog out of range for channel 16. When set, indicates the converted data is out of range for channel 16.
16 AWOR_CH15	Analog watchdog out of range for channel 15. When set, indicates the converted data is out of range for channel 15.
17 AWOR_CH14	Analog watchdog out of range for channel 14. When set, indicates the converted data is out of range for channel 14.
18 AWOR_CH13	Analog watchdog out of range for channel 13. When set, indicates the converted data is out of range for channel 13.
19 AWOR_CH12	Analog watchdog out of range for channel 12. When set, indicates the converted data is out of range for channel 12.
20 AWOR_CH11	Analog watchdog out of range for channel 11. When set, indicates the converted data is out of range for channel 11.
21 AWOR_CH10	Analog watchdog out of range for channel 10. When set, indicates the converted data is out of range for channel 10.
22 AWOR_CH9	Analog watchdog out of range for channel 9. When set, indicates the converted data is out of range for channel 9.
23 AWOR_CH8	Analog watchdog out of range for channel 8. When set, indicates the converted data is out of range for channel 8.
24 AWOR_CH7	Analog watchdog out of range for channel 7. When set, indicates the converted data is out of range for channel 7.
25 AWOR_CH6	Analog watchdog out of range for channel 6. When set, indicates the converted data is out of range for channel 6.
26 AWOR_CH5	Analog watchdog out of range for channel 5. When set, indicates the converted data is out of range for channel 5.
27 AWOR_CH4	Analog watchdog out of range for channel 4. When set, indicates the converted data is out of range for channel 4.
28 AWOR_CH3	Analog watchdog out of range for channel 3. When set, indicates the converted data is out of range for channel 3.
29 AWOR_CH2	Analog watchdog out of range for channel 2. When set, indicates the converted data is out of range for channel 2.

Table continues on the next page...

ADC_AWORR0 field descriptions (continued)

Field	Description
30 AWOR_CH1	Analog watchdog out of range for channel 1. When set, indicates the converted data is out of range for channel 1.
31 AWOR_CH0	Analog watchdog out of range for channel 0. When set, indicates the converted data is out of range for channel 0.

35.24.49 Analog Watchdog Out of Range Register 1 (ADC_AWORR1)

Analog watchdog out of range register for channels 32 to 63.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2F4h offset = 2F4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AWOR_CH47	AWOR_CH46	AWOR_CH45	AWOR_CH44	AWOR_CH43	AWOR_CH42	AWOR_CH41	AWOR_CH40	AWOR_CH39	AWOR_CH38	AWOR_CH37	AWOR_CH36	AWOR_CH35	AWOR_CH34	AWOR_CH33	AWOR_CH32
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_AWORR1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 AWOR_CH47	Analog watchdog out of range for channel 47. When set, indicates the converted data is out of range for channel 47.
17 AWOR_CH46	Analog watchdog out of range for channel 46. When set, indicates the converted data is out of range for channel 46.
18 AWOR_CH45	Analog watchdog out of range for channel 45. When set, indicates the converted data is out of range for channel 45.
19 AWOR_CH44	Analog watchdog out of range for channel 44. When set, indicates the converted data is out of range for channel 44.
20 AWOR_CH43	Analog watchdog out of range for channel 43. When set, indicates the converted data is out of range for channel 43.
21 AWOR_CH42	Analog watchdog out of range for channel 42. When set, indicates the converted data is out of range for channel 42.
22 AWOR_CH41	Analog watchdog out of range for channel 41. When set, indicates the converted data is out of range for channel 41.
23 AWOR_CH40	Analog watchdog out of range for channel 40. When set, indicates the converted data is out of range for channel 40.
24 AWOR_CH39	Analog watchdog out of range for channel 39. When set, indicates the converted data is out of range for channel 39.
25 AWOR_CH38	Analog watchdog out of range for channel 38. When set, indicates the converted data is out of range for channel 38.
26 AWOR_CH37	Analog watchdog out of range for channel 37. When set, indicates the converted data is out of range for channel 37.
27 AWOR_CH36	Analog watchdog out of range for channel 36. When set, indicates the converted data is out of range for channel 36.
28 AWOR_CH35	Analog watchdog out of range for channel 35. When set, indicates the converted data is out of range for channel 35.
29 AWOR_CH34	Analog watchdog out of range for channel 34. When set, indicates the converted data is out of range for channel 34.
30 AWOR_CH33	Analog watchdog out of range for channel 33. When set, indicates the converted data is out of range for channel 33.
31 AWOR_CH32	Analog watchdog out of range for channel 32. When set, indicates the converted data is out of range for channel 32.

35.24.50 Analog Watchdog Out of Range Register 2 (ADC_AWORR2)

Analog watchdog out of range register for channels 64 to 95.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2F8h offset = 2F8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AWOR_CH95	AWOR_CH94	AWOR_CH93	AWOR_CH92	AWOR_CH91	AWOR_CH90	AWOR_CH89	AWOR_CH88	AWOR_CH87	AWOR_CH86	AWOR_CH85	AWOR_CH84	AWOR_CH83	AWOR_CH82	AWOR_CH81	AWOR_CH80
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AWOR_CH79	AWOR_CH78	AWOR_CH77	AWOR_CH76	AWOR_CH75	AWOR_CH74	AWOR_CH73	AWOR_CH72	AWOR_CH71	AWOR_CH70	AWOR_CH69	AWOR_CH68	AWOR_CH67	AWOR_CH66	AWOR_CH65	AWOR_CH64
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_AWORR2 field descriptions

Field	Description
0 AWOR_CH95	Analog watchdog out of range for channel 95. When set, indicates the converted data is out of range for channel 95.
1 AWOR_CH94	Analog watchdog out of range for channel 94. When set, indicates the converted data is out of range for channel 94.
2 AWOR_CH93	Analog watchdog out of range for channel 93. When set, indicates the converted data is out of range for channel 93.

Table continues on the next page...

ADC_AWORR2 field descriptions (continued)

Field	Description
3 AWOR_CH92	Analog watchdog out of range for channel 92. When set, indicates the converted data is out of range for channel 92.
4 AWOR_CH91	Analog watchdog out of range for channel 91. When set, indicates the converted data is out of range for channel 91.
5 AWOR_CH90	Analog watchdog out of range for channel 90. When set, indicates the converted data is out of range for channel 90.
6 AWOR_CH89	Analog watchdog out of range for channel 89. When set, indicates the converted data is out of range for channel 89.
7 AWOR_CH88	Analog watchdog out of range for channel 88. When set, indicates the converted data is out of range for channel 88.
8 AWOR_CH87	Analog watchdog out of range for channel 87. When set, indicates the converted data is out of range for channel 87.
9 AWOR_CH86	Analog watchdog out of range for channel 86. When set, indicates the converted data is out of range for channel 86.
10 AWOR_CH85	Analog watchdog out of range for channel 85. When set, indicates the converted data is out of range for channel 85.
11 AWOR_CH84	Analog watchdog out of range for channel 84. When set, indicates the converted data is out of range for channel 84.
12 AWOR_CH83	Analog watchdog out of range for channel 83. When set, indicates the converted data is out of range for channel 83.
13 AWOR_CH82	Analog watchdog out of range for channel 82. When set, indicates the converted data is out of range for channel 82.
14 AWOR_CH81	Analog watchdog out of range for channel 81. When set, indicates the converted data is out of range for channel 81.
15 AWOR_CH80	Analog watchdog out of range for channel 80. When set, indicates the converted data is out of range for channel 80.
16 AWOR_CH79	Analog watchdog out of range for channel 79. When set, indicates the converted data is out of range for channel 79.
17 AWOR_CH78	Analog watchdog out of range for channel 78. When set, indicates the converted data is out of range for channel 78.
18 AWOR_CH77	Analog watchdog out of range for channel 77. When set, indicates the converted data is out of range for channel 77.
19 AWOR_CH76	Analog watchdog out of range for channel 76. When set, indicates the converted data is out of range for channel 76.
20 AWOR_CH75	Analog watchdog out of range for channel 75.

Table continues on the next page...

ADC_AWORR2 field descriptions (continued)

Field	Description
	When set, indicates the converted data is out of range for channel 75.
21 AWOR_CH74	Analog watchdog out of range for channel 74. When set, indicates the converted data is out of range for channel 74.
22 AWOR_CH73	Analog watchdog out of range for channel 73. When set, indicates the converted data is out of range for channel 73.
23 AWOR_CH72	Analog watchdog out of range for channel 72. When set, indicates the converted data is out of range for channel 72.
24 AWOR_CH71	Analog watchdog out of range for channel 71. When set, indicates the converted data is out of range for channel 71.
25 AWOR_CH70	Analog watchdog out of range for channel 70. When set, indicates the converted data is out of range for channel 70.
26 AWOR_CH69	Analog watchdog out of range for channel 69. When set, indicates the converted data is out of range for channel 69.
27 AWOR_CH68	Analog watchdog out of range for channel 68. When set, indicates the converted data is out of range for channel 68.
28 AWOR_CH67	Analog watchdog out of range for channel 67. When set, indicates the converted data is out of range for channel 67.
29 AWOR_CH66	Analog watchdog out of range for channel 66. When set, indicates the converted data is out of range for channel 66.
30 AWOR_CH65	Analog watchdog out of range for channel 65. When set, indicates the converted data is out of range for channel 65.
31 AWOR_CH64	Analog watchdog out of range for channel 64. When set, indicates the converted data is out of range for channel 64.

35.24.51 Self Test Configuration Register 1 (ADC_STCR1)

Address: 0h base + 340h offset = 340h

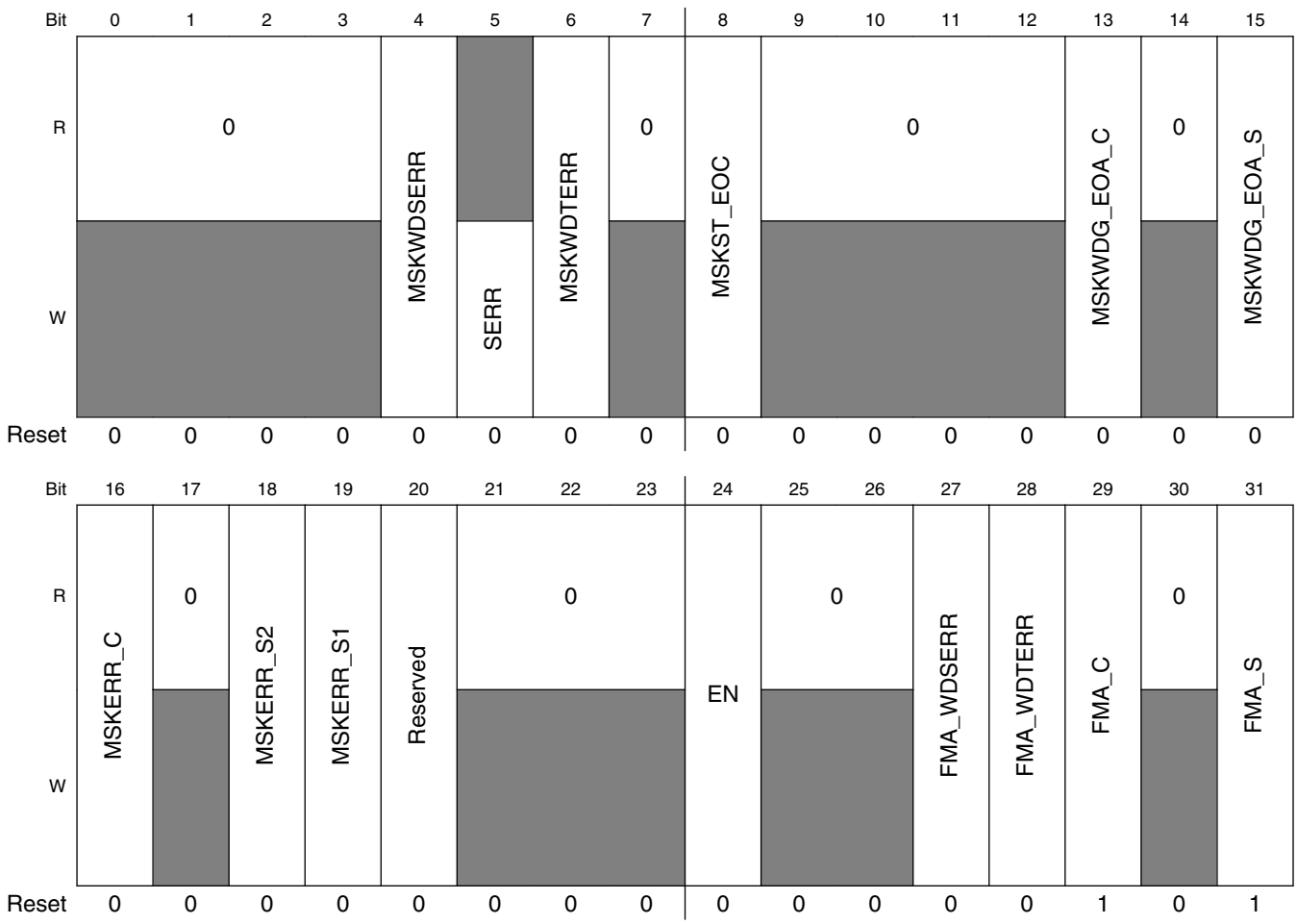
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INPSAMP_C								Reserved								INPSAMP_S								Reserved							
W																																
Reset	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	1	1	1

ADC_STCR1 field descriptions

Field	Description
0–7 INPSAMP_C	Sampling phase duration for the test conversions related to self test algorithm C. Minimum value should be set as per default (18h). Maximum can be FFh.
8–15 Reserved	This field is reserved.
16–23 INPSAMP_S	Sampling phase duration for the test conversions related to self test algorithm S. Minimum value should be set as per default (25h). Maximum can be FFh.
24–31 Reserved	This field is reserved.

35.24.52 Self Test Configuration Register 2 (ADC_STCR2)

Address: 0h base + 344h offset = 344h



ADC_STCR2 field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 MSKWDSERR	Interrupt enable (STSR1[WDSERR] status bit) 0 Interrupt disabled 1 Enables the STSR1[WDSERR] status bit to generate an interrupt indication.
5 SERR	Error fault injection bit (write only). Writing 1 to this bit sets the STSR1[ERRx] status bits once only. Reading always returns 0. Setting this bit does not affect any of the self test data registers.
6 MSKWDTERR	Interrupt enable (STSR[WDTERR] status bit) 0 Interrupt disabled 1 Enables the STSR1[WDTERR] status bit to generate an interrupt indication.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 MSKST_EOC	Interrupt Enable bit for STSR2[ST_EOC] 0 Interrupt disabled 1 Enables the STSR1[ST_EOC] status bit to generate an interrupt indication. (IMR[MSKEOC] must also be enabled.)
9–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 MSKWDG_EOA_C	Interrupt enable (WDG_EOA_C status bit) 0 Interrupt disabled 1 Enables the STSR1[WDG_EOA_C] status bit to generate an interrupt indication.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 MSKWDG_EOA_S	Interrupt enable (WDG_EOA_S status bit) 0 Interrupt disabled 1 Enables the STSR1[WDG_EOA_S] status bit to generate an interrupt indication.
16 MSKERR_C	Interrupt enable (ERR_C status bit) 0 Interrupt disabled 1 Enables the STSR1[ERR_C] status bit to generate an interrupt indication.
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 MSKERR_S2	Interrupt enable (ERR_S2 status bit) 0 Interrupt disabled 1 Enables the STSR1[ERR_S2] status bit to generate an interrupt indication.
19 MSKERR_S1	Interrupt enable (ERR_S1 status bit) 0 Interrupt disabled 1 Enables the STSR1[ERR_S1] status bit to generate an interrupt indication.
20 Reserved	This field is reserved.

Table continues on the next page...

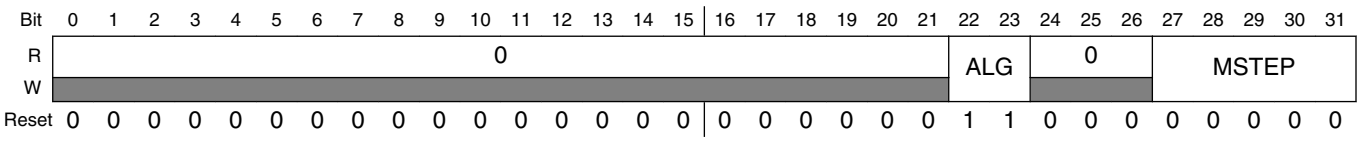
ADC_STCR2 field descriptions (continued)

Field	Description
21–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 EN	Self testing channel enable. The TEST conversions are enabled. Enables the TEST channel only in CPU mode. This bit should be set before starting the normal conversion and should not be changed while conversion is ongoing. This bit should be reset only after end of conversion for the last self test channel has been received. 0 Disabled 1 Enabled
25–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 FMA_WDSERR	Fault mapping for the Watchdog Sequence error. By default a failure on the watchdog sequence is mapped on the NCF fault line. 0 NCF mapping 1 CF mapping
28 FMA_WDTERR	Fault mapping for the Watchdog Timer error. By default a failure on the watchdog sequence is mapped on the NCF fault line. 0 NCF mapping 1 CF mapping
29 FMA_C	Fault mapping for self test algorithm C. By default a failure on the self test C algorithm is mapped on the CF fault line. 0 NCF mapping 1 CF mapping
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 FMA_S	Fault mapping for the self test algorithm BGAP. By default a failure on the Supply algorithm is mapped on the CF fault line. 0 NCF mapping 1 CF mapping

35.24.53 Self Test Configuration Register 3 (ADC_STCR3)

This register defines ADC self testing capabilities.

Address: 0h base + 348h offset = 348h



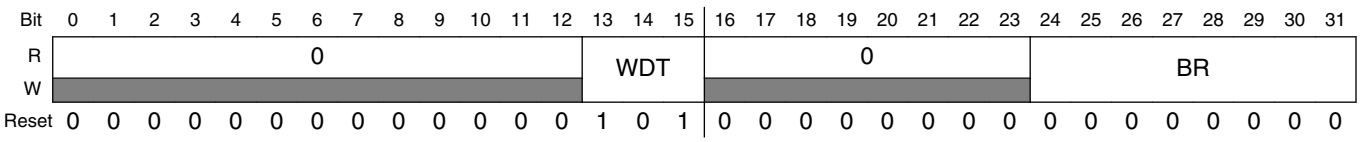
ADC_STCR3 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 ALG	Algorithm scheduling. ONE-SHOT mode algorithm scheduling: • 00: Algorithm S (single step=MSTEP) • 01: Reserved • 10: Algorithm C (single step=MSTEP) • 11: Algorithm S (default) For test/debug purposes. SCAN mode algorithm scheduling: • 00: Algorithm S • 01: Reserved • 10: Algorithm C • 11: Algorithm S + Algorithm C (default) The baud rate for the execution of the selected algorithm is defined by the STBRR register.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 MSTEP	One shot mode: Current step for Algorithm S or Algorithm C. • MSTEP = 0 to (NUM_SUPPLY_STEPS-1) for algorithm S • MSTEP = 0 to (NUM_C_STEPS-1) for algorithm C Scan mode: This field is not used in scan mode as always single step execution (interleaved mode) is performed. It should be programmed to zero in scan mode. **NOTE:** All unused valued are RESERVED and should not be used.

35.24.54 Self Test Baud Rate Register (ADC_STBRR)

Used to set the baud rate for the selected baud rate in SCAN mode as well as the watchdog timer value.

Address: 0h base + 34Ch offset = 34Ch

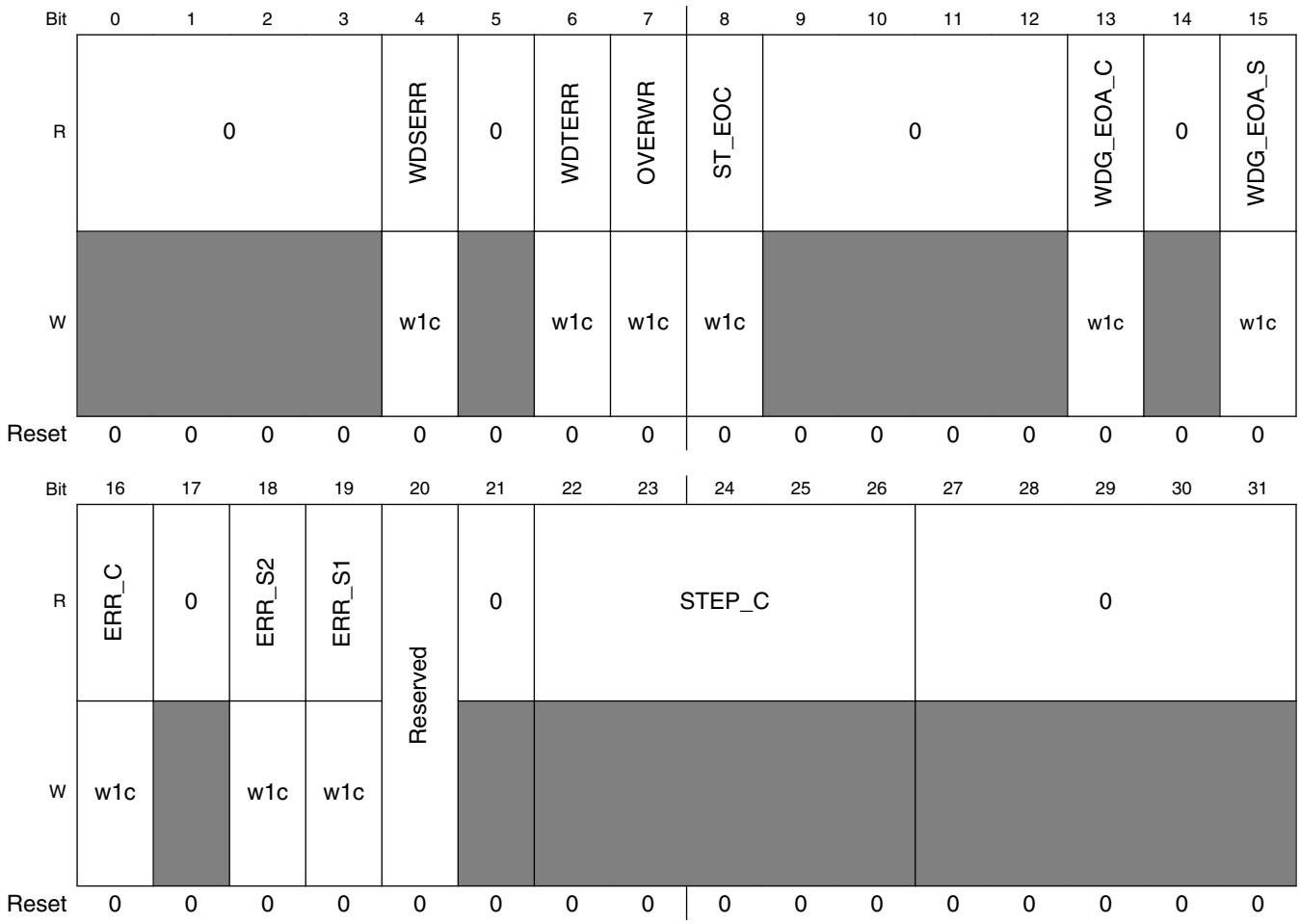


ADC_STBRR field descriptions

Field	Description
0–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 WDT	The watchdog timer value is used to monitor that the algorithm sequence is correctly executed within the safe time period. The self testing watchdog is enabled setting the STAWxR[WDTE] control bits. Default value is 10 ms. A fixed pre-scaler runs on the ADCdig clock (80 MHz). (Prescalar value is fixed at 1024). 000 0.1 ms ((0008h * Prescalar) cycles at 80 MHz) 001 0.5 ms ((0027h * Prescalar) cycles at 80 MHz) 010 1 ms ((004Eh * Prescalar) cycles at 80 MHz) 011 2 ms ((009Ch * Prescalar) cycles at 80 MHz) 100 5 ms ((0187h * Prescalar) cycles at 80 MHz) 101 10 ms ((030Dh * Prescalar) cycles at 80 MHz) 110 20 ms ((061Ah * Prescalar) cycles at 80 MHz) 111 50 ms ((0F42h * Prescalar) cycles at 80 MHz)
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 BR	Baud rate for the selected algorithm in SCAN mode (MCR[MODE] = 1). 00h Max scheduling rate (NOMINAL rate) ... FFh Min scheduling rate (NOMINAL rate scaled by 255) This field should be programmed when ST_EN is zero i.e. before enabling self test channel.

35.24.55 Self Test Status Register 1 (ADC_STSR1)

Address: 0h base + 350h offset = 350h



ADC_STSR1 field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 WDSERR	Watchdog sequence error of the ADC sub-system (check for algorithm step sequence). The WDSERR status bit is cleared writing 1. A 0 write is an invariant operation. It generates an interrupt if enabled (STCR2[MSKWDSERR] = 1). It provides the fault indication to an outside module such as the FCCU, asserting the Critical type or Non-Critical type lines according to the STCR2[FMA_WDSERR] mapping. **NOTE:** This bit is set only if STAWxR[WDTE] = '1'. 0 No failure 1 Failure occurred
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

ADC_STSR1 field descriptions (continued)

Field	Description
6 WDTERR	Watchdog timer error of the ADC sub-system (algorithm check for completion within safe time). The WDTERR status bit is cleared writing 1. A 0 write is an invariant operation. It generates an interrupt if enabled (STCR2[MSKWDTERR] = 1). It provides the fault indication to an outside module such as the FCCU, asserting the Critical type or Non-Critical type lines according to the STCR2[FMA_WDTERR] mapping. 0 No failure 1 Failure occurred
7 OVERWR	Overwrite error. Used to notify when the STSR1[ERRx] bit is overwritten by a newer one. The OVERWR status bit is cleared writing 1. A 0 write is an invariant operation. The new error status is written or discarded according to the MCR[OWREN] bit value. To avoid OVERWR indication, the ERRx status bit must be cleared (via software).
8 ST_EOC	Self Test EOC Bit. This bit is set along with EOC bit when end_of_conversion signal is received from ADC analog for self test channel. The ST_EOC status bit is cleared by writing 1b. A 0b write is an invariant operation. It generates an interrupt if enabled by STCR2[MSKST_EOC]. (Also, IMR[MSKEOC] has to be enabled.)
9–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 WDG_EOA_C	This bit indicates that Algorithm C has been completed. This bit is generated after the last STEP of algorithm C is executed. The WDG_EOA_C status bit is cleared by writing 1. A 0 write is an invariant operation. It generates an interrupt if enabled (STCR2[MSKWDG_EOA_C] = 1). This bit is set only if STAW4R[WDTE] = 1.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 WDG_EOA_S	This bit indicates that Algorithm S has been completed. This bit is generated after the last STEP of algorithm S is executed. The WDG_EOA_S status bit is cleared writing 1. A 0 write is an invariant operation. It generates an interrupt if enabled (STCR2[MSKWDG_EOA_S] = 1). This bit is set only if STAW0R[WDTE] = ‘1’.
16 ERR_C	Indicates an error on the self testing channel (algorithm C). The ERR_C status bit is cleared writing 1b. A 0b write is an invariant operation. It generates an interrupt if enabled (STCR2[MSKERR_C] = 1). It provides the fault indication to an outside module such as the FCCU, asserting the programmed fault line (STCR2[FMAx]). The ERR_C bit can be also set via SW (fault injection) writing 1b into the STCR2[SERR] bit. In this case the Critical type or Non-Critical type lines are asserted according to the STCR2[FMAx] mapping. 0 No C-algorithm ERROR 1 C-algorithm ERROR occurred
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 ERR_S2	Indicates an error on the self testing channel (algorithm SUPPLY, step2). The ERR_S2 status bit is cleared by writing a 1b. A 0b write is an invariant operation. It generates an interrupt (“ipi_int_er” line) if enabled (STCR2[MSKERR_S2] = 1). It provides the fault indication to an outside module such as the FCCU, asserting the programmed fault line (STCR2[FMAx]). The ERR_S2 bit can be also set via SW (fault injection) writing 1b into the STCR2[SERR] bit. In this case the Critical / Non-Critical lines are asserted according to the STCR2[FMAx] mapping. 0 No ERROR occurred on the sampled signal 1 ERROR occurred on the sampled signal

Table continues on the next page...

ADC_STSR1 field descriptions (continued)

Field	Description
19 ERR_S1	Indicates an error on the self testing channel (algorithm SUPPLY, step1). The ERR_S1 status bit is cleared writing 1. A 0 write is an invariant operation. It generates an interrupt if enabled (STCR2[MSKERR_S1] = 1). It provides the fault indication to an outside module such as the FCCU, asserting the programmed fault line (STCR2[FMAx]). The ERR_S1 bit can be also set via SW (fault injection) writing 1b into the STCR2[SERR] bit. In this case the Critical / Non-Critical lines are asserted according to the STCR2[FMAx] mapping. 0 No VDD ERROR 1 VDD ERROR occurred
20 Reserved	This field is reserved.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–26 STEP_C	Step of the algorithm C when an ERR_C has occurred. 0.. (NUM_C_STEPS-1) => algorithm C
27–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

35.24.56 Self Test Status Register 2 (ADC_STSR2)

If MCR[OVERWR] bit is set, registers STSR2-4 are overwritten if another error occurs.

Address: 0h base + 354h offset = 354h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OVFL	0			DATA1											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				DATA0											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_STSR2 field descriptions

Field	Description
0 OVFL	Overflow Bit This bit is set when the divisor is zero. Also, the STSR1[ERR_S1] bit is set if overflow occurs.
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 DATA1	Test channel converted data when the ERR_S1 has occurred. Algorithm S (step1) => fractional part of the ratio TEST(step1)/TEST (step0) = VDD/VBGAP
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 DATA0	Test channel converted data when the ERR_S1 has occurred. Algorithm S (step1) => integer part of the ratio TEST(step1)/TEST (step0) = VDD/VBGAP

35.24.57 Self Test Status Register 3 (ADC_STSR3)

If MCR[OVERWR] bit is set, registers STSR2-4 are overwritten if another error occurs.

Address: 0h base + 358h offset = 358h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				DATA1												0				0											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_STSR3 field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 DATA1	Test channel converted data when the ERR_S2 has occurred. Algorithm S (step2) => TEST channel data = VREF/VREF
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

35.24.58 Self Test Status Register 4 (ADC_STSR4)

If MCR[OVERWR] bit is set, registers STSR2-4 are overwritten if another error occurs.

Address: 0h base + 35Ch offset = 35Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				DATA1												0				0											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_STSR4 field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 DATA1	Test channel converted data when the ERR_C has occurred. Algorithm C => TEST channel data
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

35.24.59 Self Test Data Register 1 (ADC_STDR1)

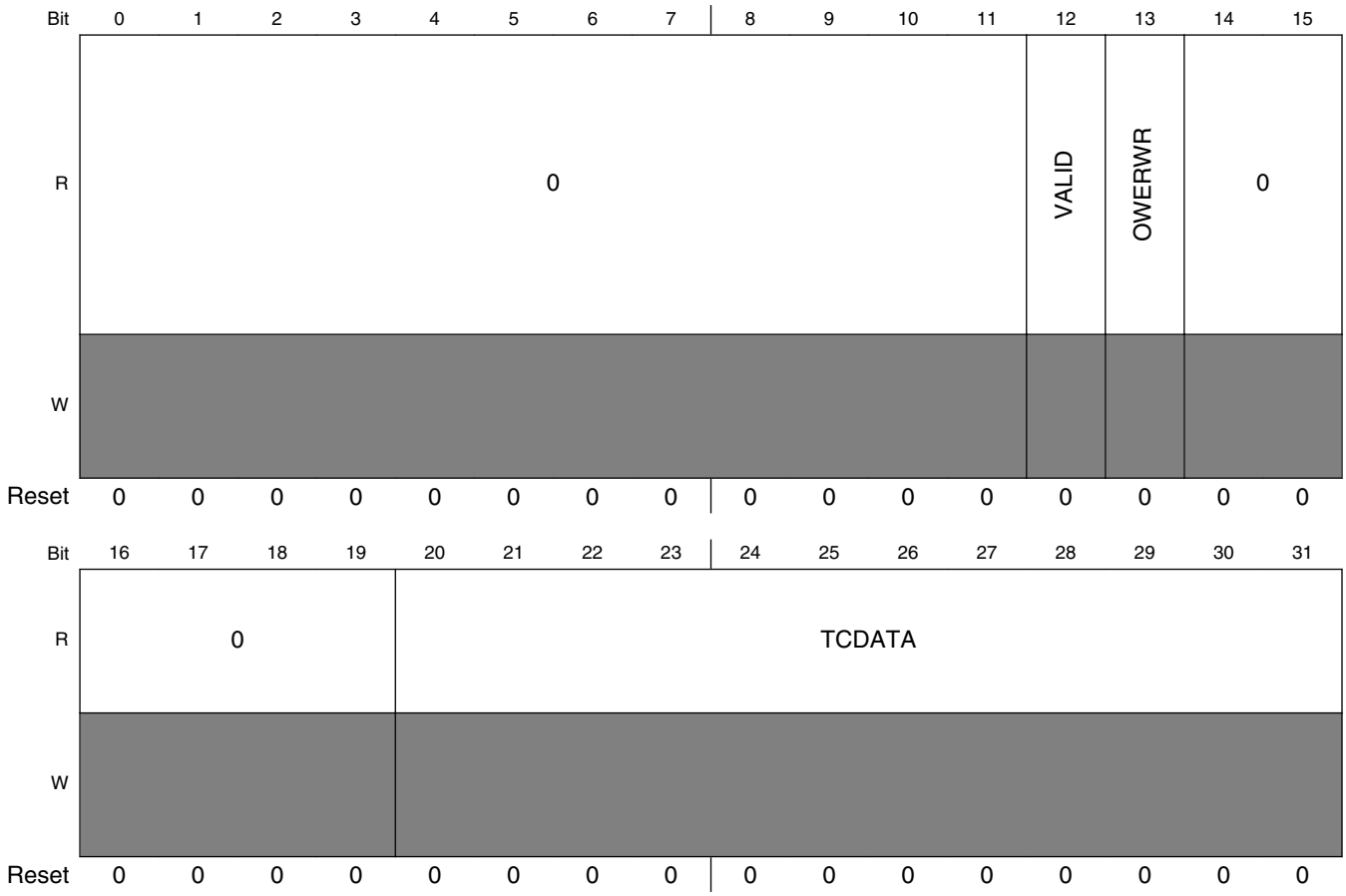
The ADC_STDR1 register contains the most current result data for conversions executed during ADC self test. Additionally, the register contains status bits to indicate whether result data has been read and whether existing conversion data has been overwritten by a newer result.

NOTE

1. For Algorithm S step 1, the result data contained in this register is not the final step 1 result. The step 1 data is the conversion of the analog supply (VDDA). This VDDA conversion result is divided by the Algorithm S step 0 conversion data and the result is written to the ADC_STDR2 register.
2. The least significant 3 bits of the analog supply (VDDA) conversion executed during Algorithm S step 1 processing are always '0'. The conversion performed in Algorithm S Step 1 is not on the full scale analog supply. Instead, the voltage sampled is VDDA / 8. The prescaling is done to avoid issues that can arise in the analog portion of the ADC

when VREFH = VDDA = 3.3 V, or VDDA is slightly higher than VREFH. The conversion result is multiplied by 8 before being written to this register to scale the result to the expected level, which results in the 3 least significant bits having a value of '0'.

Address: 0h base + 370h offset = 370h



ADC_STDR1 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 VALID	Valid data. Used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read.
13 OWERWR	Overwrite data. Used to notify when a conversion data is overwritten by a newer result. The new data is written or discarded according to the MCR[OWREN] bit value.
14–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

ADC_STDR1 field descriptions (continued)

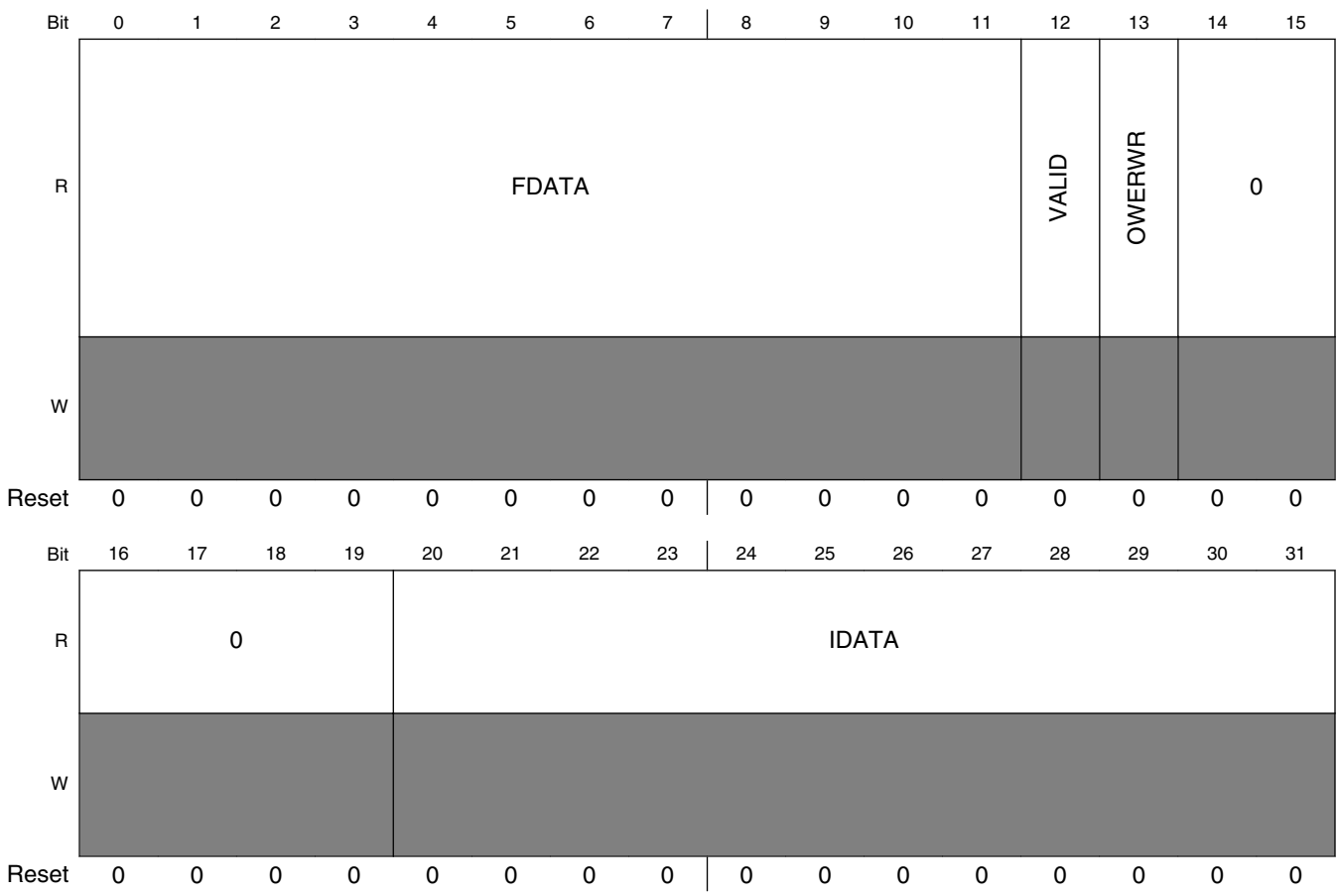
Field	Description
20–31 TCDATA	Test channel converted data. - Unsigned format for Supply Self test - 2's complement for Capacitive Self test

35.24.60 Self Test Data Register 2 (ADC_STDR2)

NOTE

If overflow error occurs for STEP1 of Algorithm-S, STDR2 is not written.

Address: 0h base + 374h offset = 374h



ADC_STDR2 field descriptions

Field	Description
0–11 FDATA	Fractional part of the ratio TEST(step1)/TEST (step0) = VDD/VBGAP for the algorithm S.

Table continues on the next page...

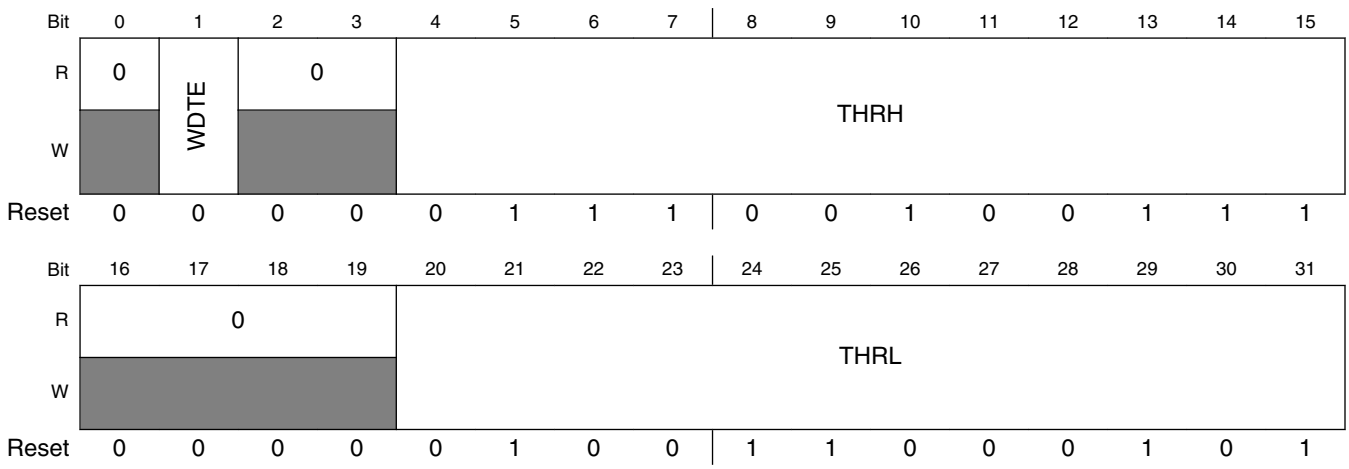
ADC_STDR2 field descriptions (continued)

Field	Description
12 VALID	Valid data. Used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read.
13 OWERWR	Overwrite data. Used to notify when a conversion data is overwritten by a newer result. The new data is written or discarded according to the MCR[OWREN] bit value.
14–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 IDATA	Integer part of the ratio TEST(step1)/TEST (step0) = VDD/VBGAP for the algorithm S.

35.24.61 Self Test Analog Watchdog Register 0 (ADC_STAW0R)

This register provides control values associated with the ADC's supply self test step 0, which samples bandgap voltage.

Address: 0h base + 380h offset = 380h



ADC_STAW0R field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 WDTE	Watchdog timer enable (related to the algorithm S) Enables/disables the watchdog timer monitoring function for all ADC supply self test steps. The watchdog timer verifies: • Correct sequence of the algorithm (step sequence) • Execution of the algorithm within the safe time period as defined by STBRR[WDT]

Table continues on the next page...

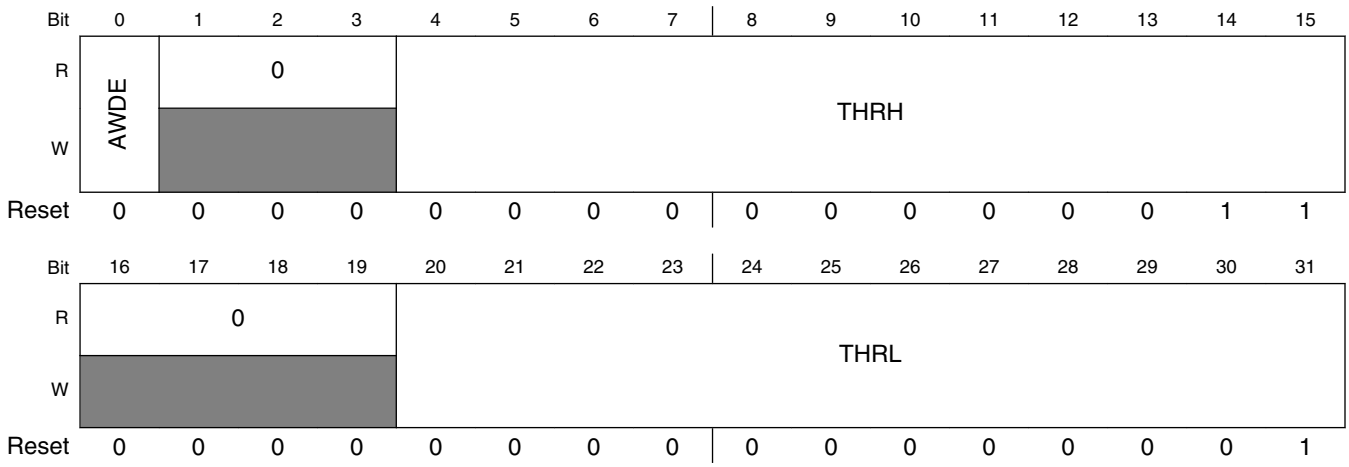
ADC_STAW0R field descriptions (continued)

Field	Description
	As soon as the watchdog timer is enabled the algorithm execution must be detected within the safe time period. The watchdog timer is reset each time the algorithm restarts. This bit should be set only in scan mode. 0 Disabled 1 Enabled
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 THRH	High threshold value for Algorithm S step 0 If the analog watchdog is enabled, the STSR1[ERRx] status bit is set if STDR1[TCDATA]) > THRH. **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value for Algorithm S step 0 If the analog watchdog is enabled, the STSR1[ERRx] status bit is set if STDR1[TCDATA]) < THRL. **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.

35.24.62 Self Test Analog Watchdog Register 1A (ADC_STAW1AR)

This register contains values associated with the ADC's supply self test step 1, in which the analog supply (VDDA) is sampled and a fixed point division is performed, dividing the conversion result by the result from step 0 (bandgap voltage). The integer portion of the result is handled separately from the fractional part, i.e., they have separate data registers and threshold fields. Thresholds in this register are applied to the integer portion of the result. Thresholds in the ADC_STAW1BR register are applied to the fractional portion of the result.

Address: 0h base + 384h offset = 384h



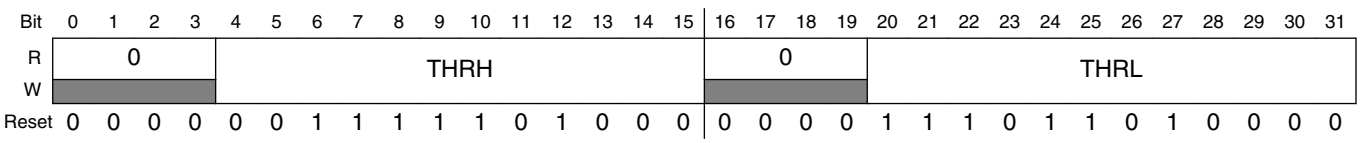
ADC_STAW1AR field descriptions

Field	Description
0 AWDE	Analog watchdog enable related to the algorithm S (step1).
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 THRH	High threshold value (integer part) for test channel for algorithm S (step 1) (unsigned coding). **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value (integer part) for test channel for algorithm S (step 1) (unsigned coding). **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.

35.24.63 Self Test Analog Watchdog Register 1B (ADC_STAW1BR)

This register contains values associated with the ADC's supply self test step 1, in which the analog supply (VDDA) is sampled and a fixed point division is performed, dividing the conversion result by the result from step 0 (bandgap voltage). The integer portion of the result is handled separately from the fractional part, i.e., they have separate data registers and threshold fields. Thresholds in this register are applied to the fractional portion of the result. Thresholds in the ADC_STAW1AR register are applied to the integer portion of the result.

Address: 0h base + 388h offset = 388h



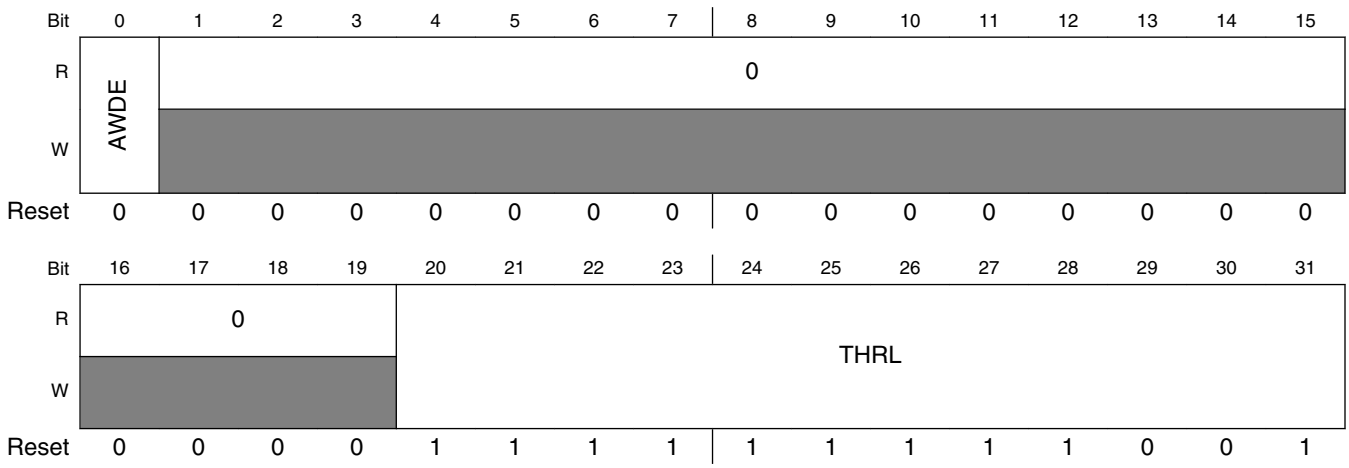
ADC_STAW1BR field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 THRH	High threshold value (fractional part) for test channel for algorithm S (step 1)(unsigned coding). **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value (fractional part) for test channel for algorithm S (step 1) (unsigned coding). **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.

35.24.64 Self Test Analog Watchdog Register 2 (ADC_STAW2R)

This register contains control values associated with the ADC's supply self test step 2, in which the ADC's high reference voltage is sampled and converted. Note that there is no upper threshold value in this register since the conversion result should ideally be FFFh (12-bit) or 3FF (10-bit).

Address: 0h base + 38Ch offset = 38Ch



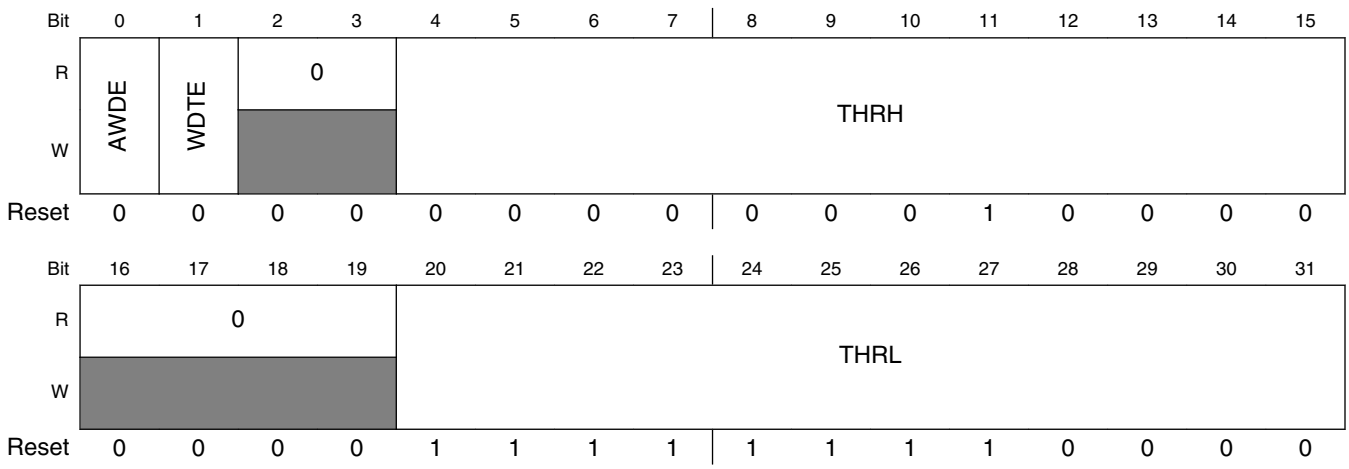
ADC_STAW2R field descriptions

Field	Description
0 AWDE	Analog watchdog enable related to the algorithm S (step2).
1–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value for channel x (unsigned coding). If the analog watchdog is enabled, the STSR1[ERR_S2] status bit is set if STDR1[TCDATA] < THRL. **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment, i.e., introduction of noise from outside of the device minimized. The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.

35.24.65 Self Test Analog Watchdog Register 4 (ADC_STAW4R)

This register contains ADC self test configuration controls associated with the ADC's capacitive self test (also referred to as "Algorithm C") sequence, which samples voltages from the ADC's internal capacitor matrix. The watchdog enable fields (AWDE and WDTE) affect all Algorithm C steps; the threshold fields (THRH and THRL) fields are specific to Algorithm C Step 0.

Address: 0h base + 394h offset = 394h



ADC_STAW4R field descriptions

Field	Description
0 AWDE	Analog watchdog enable (related to the algorithm C) Enables/disables the analog watchdog monitoring function for all ADC capacitive self test steps. 0 Disabled 1 Enabled
1 WDTE	Watchdog timer enable (related to the algorithm C). Enables/disables the watchdog timer monitoring function for all ADC capacitive self test steps. The watchdog timer verifies: • Correct sequence of the algorithm (step sequence) • Execution of the algorithm within the safe time period as defined by STBRR[WDT] As soon as the watchdog timer is enabled, execution must be detected within the safe time period. The watchdog timer is reset each time the algorithm restarts. This bit should be set only in scan mode. 0 Disabled 1 Enabled
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 THRH	High threshold value for step 0 of C algorithm.

Table continues on the next page...

ADC_STAW4R field descriptions (continued)

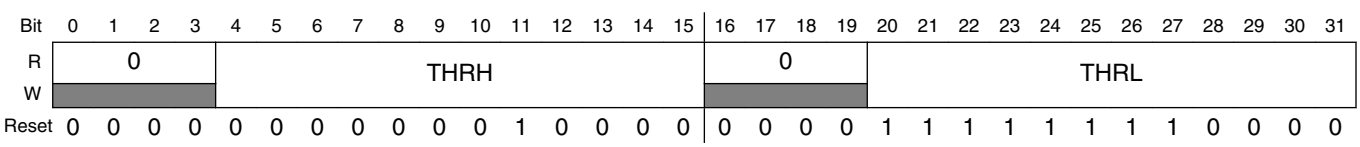
Field	Description
	This value is read by the ADC as twos-complement, but should be written positive. If the watchdog is enabled (STAW4R[AWDE] = 1) and STDR1[TCDATA] > STAW4R[THRH], then STSR1[ERR_C] is set. **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment (for example, introduction of noise from outside of the device minimized). The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value for step 0 of C algorithm. This value is read by the ADC as twos-complement, and it should be the negative of STAW4R[THRH]. If the watchdog is enabled (STAW4R[AWDE] = 1) and STDR1[TCDATA] < STAW4R[THRL], then STSR1[ERR_C] is set. **NOTE:** The settings in this register are applicable only to step 0 of the ADC's capacitive self test. Refer to the ADC_STAW5R register for settings applicable to other capacitive self test steps. The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment (for example, introduction of noise from outside of the device minimized). The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.

35.24.66 Self Test Analog Watchdog Register 5 (ADC_STAW5R)

This register contains ADC self test threshold values associated with all capacitive self test (also referred to as "Algorithm C") steps other than Step 0, which is controlled by fields in the ADC_STAW4R register.

Note that the threshold values in this register are distinctly different than those in the ADC_STAW4R register. The threshold values in this register do not represent a voltage conversion value relative to the ADC high reference voltage, but instead represent the upper and lower thresholds for the absolute difference between the voltage sampled and converted in an Algorithm C step and the voltage sampled and converted in Algorithm C Step 0. Ideally, the difference will be zero.

Address: 0h base + 398h offset = 398h



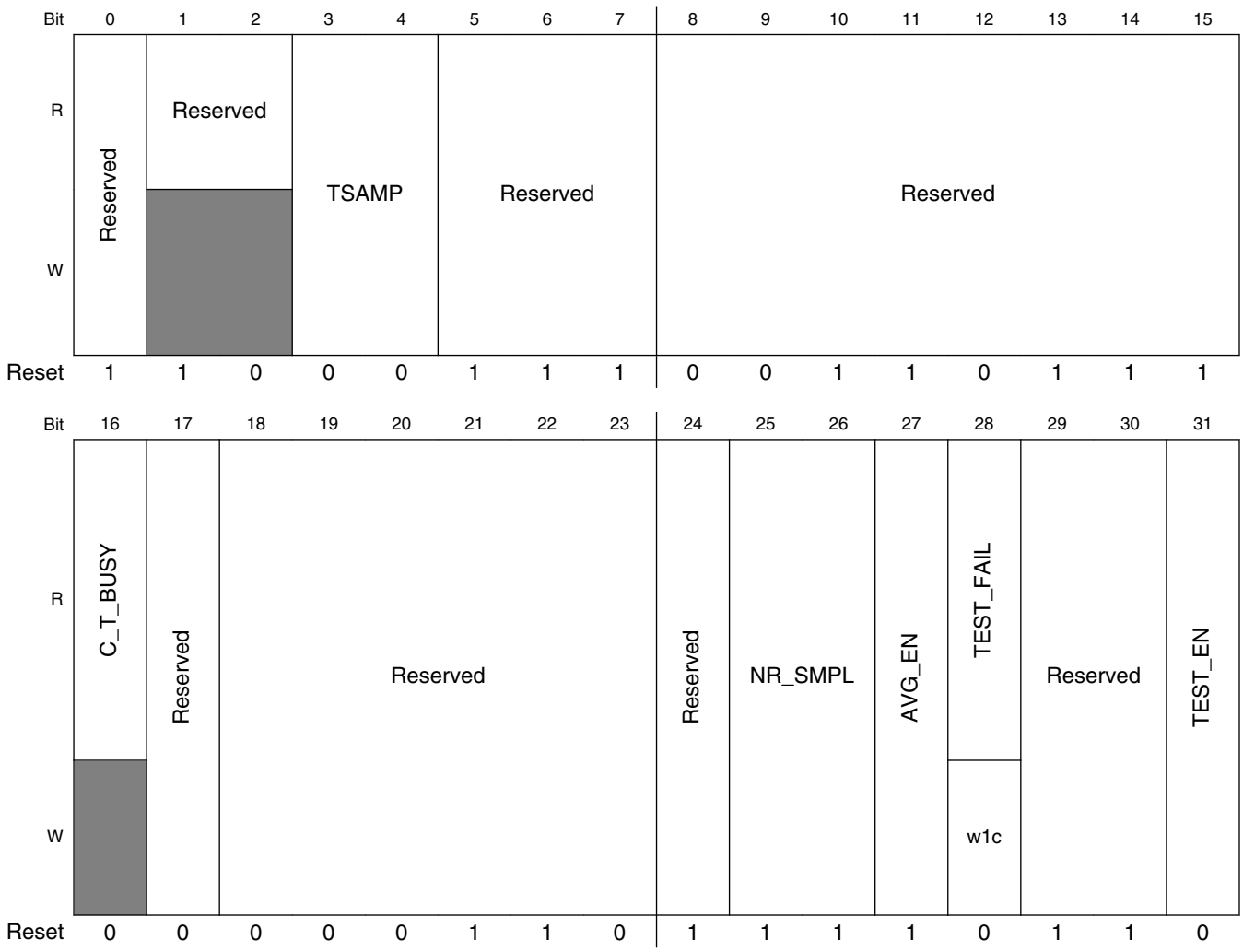
ADC_STAW5R field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 THRH	High threshold value for step N of C algorithm (N = 1 to CS-1). This value is read by the ADC as twos-complement, but should be written positive. If the watchdog is enabled (STAW4R[AWDE] = 1) and (STDR1[TCDATA] (step-N) – STDR1[TCDATA] (step-0)) > STAW5R[THRH], then STSR1[ERR_C] is set. **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment (for example, introduction of noise from outside of the device minimized). The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value for step 0 of C algorithm. This value is read by the ADC as twos-complement, and it should be the negative of STAW5R[THRH]. If the watchdog is enabled (STAW4R[AWDE = 1]) and (STDR1[TCDATA] (step-N) – STDR1[TCDATA] (step-0)) < STAW5R[THRL], then STSR1[ERR_C] is set. **NOTE:** The recommended watchdog threshold values are set according to expected and tested results in a noise-controlled environment (for example, introduction of noise from outside of the device minimized). The setting may need to be adjusted to prevent a given threshold value from falsely reporting fault detections in applications operating in noisier environments.

35.24.67 Calibration, BIST Control and status Register (ADC_CALBISTREG)

This register controls the Offset calculation, Calibration and BIST functionality with Averaging option. The averaging function has no effect in other conversion modes.

Address: 0h base + 3A0h offset = 3A0h



ADC_CALBISTREG field descriptions

Field	Description
0 Reserved	This field is reserved. When writing to this register, do not change the default value of this field.
1–2 Reserved	This field is reserved. When writing to this register, do not change the default value of this field.
3–4 TSAMP	Test Sample period in Calibration, BIST and Offset calculation process. 00 22 cycles of ADC clk

Table continues on the next page...

ADC_CALBISTREG field descriptions (continued)

Field	Description
	01 8 cycles of ADC clk 10 16 cycle of ADC clk 11 32 cycle of ADC clk
5–7 Reserved	This field is reserved.
8–15 Reserved	This field is reserved. When writing to this register, do not change the default value of this field.
16 C_T_BUSY	This bit indicates that ADC is doing internal operations for BIST or Calibration. This is set when the TEST_EN bit is set. 0 ADC is ready for use 1 ADC is busy in calibration or test operation
17 Reserved	This field is reserved.
18–23 Reserved	This field is reserved. When writing to this register, do not change the default value of this field.
24 Reserved	This field is reserved. The default value must not change. Always write the default value to this field.
25–26 NR_SMPL	Number of Samples for averaging. **NOTE:** The 16 samples setting is used only when the application needs quicker calibration (reduced time) to trade off with accuracy. For the best accuracy, it is necessary to use 512 samples. The minimum recommended number of samples is 32 samples. 00 16 samples 01 32 samples 10 128 samples 11 512 samples
27 AVG_EN	Average Enable (for Calibration only). 0 Disable 1 Enable
28 TEST_FAIL	Test Fail. This bit indicates the status of Calibration output. This bit sets automatically if tests are aborted prematurely by any normal conversion or tests complete but one of them fails. This bit is cleared by writing 1 to it or on restarting of test by writing 1 to TEST_EN 0 Test passed 1 Test failed
29–30 Reserved	This field is reserved.
31 TEST_EN	Enable the test. 0 Disable the test 1 Enable the test (self clearing) (once set it cannot be overwritten till it resets)

35.24.68 Offset and Gain User Register (ADC_OFSGNUSR)

This register contains the user configured offset and Gain values used by ADC for data processing and have all the bits valid (MSB through LSB).

OFFSET and Gain Calibration values can be either positive or negative, so they need to be in 2's complement format with MSB as sign bit.

Address: 0h base + 3A8h offset = 3A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						GAIN_USER										0								OFFSET_USER							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_OFSGNUSR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 GAIN_USER	Gain User (2's complement format)
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 OFFSET_USER	Offset User (2's complement format)

35.2.5 Functional description

The following sections detail the ADCD interfaces as well as the functionality of this module.

35.2.5.1 Conversion

After power-up or reset, the ADC is in power-down mode until the MCR[PWDN] field is written. There are some configurations, listed below, available only in power-down mode, that must be handled before exiting power down.

- Clock selection for SAR Controller (SAR_CLK): MCR [ADCLKSEL]
- Operating mode of ADC for BCTU (Trigger / Control): MCR [CTU_MODE] (if available)

After the above configurations, ADC need to be taken out from power-down mode by writing '0' into MCR [PWDN], Conversions can be initiated now in the following modes:

- Normal

- Injected
- BCTU Triggered

35.2.5.1.1 Normal conversion

The conversion mode generally used to convert channels is called normal conversion. Programming the available Normal Conversion Mask Register(s) (ADC_NCMR*n*) selects the channel(s). Each channel can be individually enabled by setting a bit in the corresponding field of the appropriate NCMR*n* register.

- NCMR0 serves *precision* channels:
  - 0 - 31 (ADC0)
  - 0 - 31 (ADC1)
  - 0 - 31 (ADC2)
  - 0 - 20 (ADC3)
- NCMR1 serves *internal* channels:
  - 32 - 39 (ADC0)
  - 32 - 38 (ADC1)
  - 32 - 36 (ADC2)
  - 32 - 38 (ADC3)
- NCMR2 serves *external* channels:
  - none (ADC0)
  - none (ADC1)
  - none (ADC2)
  - 64 to 95 (ADC3)

The mask register(s) must be programmed before starting the conversion and cannot be changed until the conversion of all the selected channels ends. Conversion always starts from the lowest channel number selected and sequentially goes to the highest number.

35.2.5.1.1.1 Starting a normal conversion

After programming the MCR and NCMR0*n* register(s), a normal conversion is started via *software* by setting the MCR[NSTART] bit.

To start a normal conversion via *external trigger*, enable external trigger by setting the (MCR[TRGEN] bit. A programmed event (rising/falling edge depending on MCR[EDGE] bit) on the normal trigger input starts the conversion.

Detection of an active edge defined by MCR[EDGE] bit (rising=1, falling=0) on the external trigger will start the normal conversion.

35.2.5.1.2 Normal conversion operating modes

Two operating modes are available for normal conversion:

- One-Shot mode
- Scan mode

The MCR[MODE] bit determines which mode is used during a normal conversion. The first phase of the conversion process involves sampling the analog channel. The next phase is the conversion phase, during which the sampled analog value is converted to digital as shown in the figure below.



Figure 35-156. Normal conversion flow

35.2.5.1.2.1 One shot mode

In one shot mode (MCR[MODE] = 0) a sequential conversion specified in the following mask register(s) is performed only once:

- NCMR0
- NCMR1
- NCMR2

At the end of each conversion, the digital result of the conversion is stored into the corresponding data register(CDRx, x = channel number).

For example: Channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in One-Shot mode. Conversion starts with channel B, followed by the conversion of channels D and E. At the end of conversion of channel E, the scanning of channels will stop.

The MSR[NSTART] status bit is automatically set when the normal conversion starts. At the same time, MCR[NSTART]is reset by hardware, allowing the software to program a new start of conversion in advance. In this case, the newly requested conversion starts after completing the running/current conversion. However, for the sake of simplicity and straightforwardness, it is advisable to program the NSTART bit for the new conversion only after the completion of the running conversion. To know the end of conversion of running chain, application will either wait for an End of Chain (ECH) Interrupt or check the status of MSR[NSTART] prior to setting MCR[NSTART] again.

If the conversion chain is started by the external trigger, the trigger input will not be observed by the ADC block until the current conversion chain is finished. After the chain is over, the next chain can be invoked/triggered by another hardware trigger edge.

35.2.5.1.2.2 Scan mode

In scan mode (MCR[MODE] = 1), a sequential conversion of *n* channels specified in the NCMR*n* register(s) is continuously performed. At the end of each conversion, the converted result is stored into the corresponding data registers, as in One-Shot mode.

The MSR[NSTART] status bit is automatically set when the normal conversion starts. Unlike One-Shot mode, the MCR[NSTART] bit is not reset automatically in Scan mode. It can be reset by software when you need to stop Scan mode. In that case, the ADC completes the current chain and, after the last conversion, resets MSR[NSTART].

For example: Channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in Scan mode. MCR[MODE] = 1 is set for Scan mode. Conversion starts from channel B, followed by the conversion of channels D and E. At the end of the conversion of channel E, the conversion of channel B starts again, followed by the conversion of channels D and E. This sequence repeats itself until MCR[NSTART] is reset by software.

This scan chain can be stopped by writing 0 to MCR[NSTART]. The conversion stops when the ongoing chain is finished.

NOTE

If the conversion is started by an external trigger, MCR[NSTART] is not set. As a consequence, once started, the only way to stop a Scan mode conversion started by an external trigger is to set the MCR[MODE] bit to 0.

35.2.5.1.2.3 End of conversion

In both modes, at the end of each conversion an end-of-conversion (EOC) interrupt is issued if enabled by the corresponding mask bit in the following register(s):

- CIMR0
- CIMR1
- CIMR2

Additionally, the EOC interrupt must enabled in the IMR register. After the conversion of all the channels selected in the NCMR*n* register(s) is completed and the conversion operation is considered finished, ISR[ECH] is set and an end-of-chain (ECH) interrupt is issued (if enabled in IMR[MSKECH]). The corresponding channel bit within the appropriate CEOCFR*n* register is updated to indicate that data is available on the data

register (CDR*n*) of the respective channel. If there is no channel selected in the NCMR*n* register(s) and there is a start of conversion trigger, then ISR[ECH] is set and an ECH interrupt is immediately issued (if enabled).

NOTE

To add or remove channels to/from the ongoing chain:

1. Clear the MCR[NSTART] bit
2. Set/clear mask bit for the needed channels in NCMR*n*
3. Set the MCR[NSTART] bit

35.2.5.1.3 Injected conversion

An injected conversion is a conversion chain that can be injected at any time, including into an ongoing normal conversion chain. To do so, configure the injected mask register(s):

- JCMR0
- JCMR1
- JCMR2

As in a normal conversion, each internal or external channel can be individually selected for injected conversion.

- JCMR0 controls *precision* channels:
  - 0 - 31 (ADC0)
  - 0 - 31 (ADC1)
  - 0 - 31 (ADC2)
  - 0 - 20 (ADC3)
- JCMR1 controls *internal* channels:
  - 32 - 39 (ADC0)
  - 32 - 38 (ADC1)
  - 32 - 36 (ADC2)
  - 32 - 38 (ADC3)
- JCMR2 controls *external* channels:
  - none (ADC0)
  - none (ADC1)
  - none (ADC2)
  - 64 to 95 (ADC3)

Injected conversion can only be run in One-Shot mode, and can only interrupt a normal conversion. Please note, that a BCTU conversion cannot be interrupted by an injected conversion. In this case, the injected conversion is discarded. When an injected chain is inserted into an ongoing normal chain, the ongoing normal channel is suspended and the

injected channel request is processed. After completing the last channel in the injected chain, normal conversion resumes from the channel at which it was suspended as shown in the following figure.

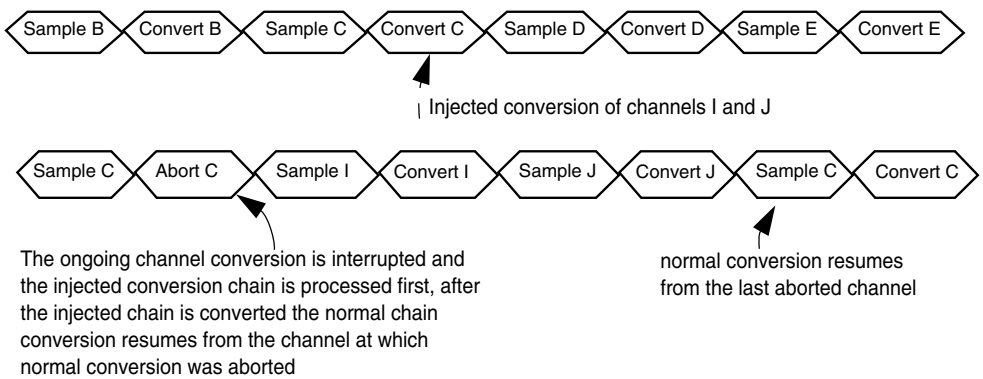


Figure 35-157. Injected sample/conversion sequence

35.2.5.1.3.1 Starting injected conversion

The injected conversion can be started via *software* by setting the MCR[JSTART] bit to start an injected conversion; the current normal conversion is suspended and the injected chain is converted.

The MSR[JSTART] status bit is automatically set when the injected conversion starts. At the same time, the MCR[JSTART] bit is reset, allowing software to program a new start of conversion in advance. In that case, the new requested conversion starts after the running conversion is completed.

At the end of each conversion, an End Of Injected Conversion (JEOC) interrupt is issued (if enabled by the corresponding mask bit in IMR) and at the end of the chain an End Of Injected Chain (JECH) interrupt is issued (if enabled by the corresponding mask bit in IMR). The status bit ISR[JECH] is also set.

If the content of all the injected conversion mask registers (JCMRn) is zero (i.e., no channel is selected) the interrupt JECH is immediately issued to resume normal operation or conversion mode.

The corresponding channel bit in the appropriate CEOCFR*n* register is updated to indicate that data is available in the data register (CDR) of the respective channel.

Once started, injected chain conversion cannot be interrupted by Normal or another Injected conversion. A BCTU trigger has the higher priority; injected conversion has a lower priority. Therefore, only a BCTU trigger can interrupt an ongoing injected conversion. In this case, the injected conversion will be discarded.

35.2.5.1.4 Aborting conversions

Two different abort functions are provided:

- A single channel abort
- A chain abort

You can abort the ongoing conversion by setting the MCR[ABORT] bit. The current conversion is aborted and the conversion of the next channel of the chain is immediately started. The abort action may take 1 to 4 cycles of the bus clock to abort the current conversion depending on the state of the conversion. At the start of any conversion and at the end of a particular conversion, when internal state counters are changing, then abort is delayed by a maximum of 3 cycles to take all states in stable condition. Abort also should not be programmed along with and within 3 cycles of programming the MCR[NSTART] or MCR[JSTART]. This may not have a significant impact from software point of view since on any SoC the next write access comes only after a minimum of 3 clock cycles.

NOTE

Programming Abort or Abort chain with NSTART or JSTART in idle condition has no effect.

For a Normal conversion chain having only one channel:

1. Abort re-runs the conversion again from the Abort point
2. If Interrupted by BCTU (in Trigger mode) Normal conversion runs twice instead of just once

In the case of an abort operation, the MSR[NSTART/JSTART] bit remains set, if not in last channel of chain, and the MCR[ABORT] bit is reset as soon as the channel is aborted. The EOC corresponding to the aborted channel is not generated. This behavior is true for Normal or Injected conversion modes. If the last channel of a chain is aborted, the end of chain is reported by generating an ECH interrupt.

It is also possible to abort the current chain of conversions by setting the MCR[ABORTCHAIN] bit. In that case, the behavior of the ADC depends on the MCR[MODE] bit (One-Shot/Scan) for Normal conversion. In One-Shot mode, MSR[NSTART] is automatically reset together with the MCR[ABORTCHAIN]. Otherwise, in Scan mode, a new chain is started, and the current chain is aborted. The EOC of the current aborted conversion is not generated, but an ECH interrupt is generated to signal the end of the chain. For Injected conversion, the behavior of ABORTCHAIN is same as for One-shot Normal conversion. In this case, MSR[JSTART] is automatically reset together with the MCR[ABORTCHAIN].

When an ABORTCHAIN is requested while an injected chain is running over a suspended normal chain, both injected and normal chains are aborted; both MSR[NSTART] and MSR[JSTART] are reset. ABORT and ABORTCHAIN requests are ignored during BCTU conversion. ABORT requests are also discarded during Self-Test conversion.

35.2.5.2 Crosstriggering Unit interface

The Crosstriggering Unit (BCTU) interface enhances the injected conversion capability of the ADC. The BCTU contains multiple event inputs that can be used to select the channels to be converted from the appropriate event configuration register. The BCTU/ ADC interface is shown in the figure below. The BCTU generates a trigger and a channel number to be converted. A single channel is converted for each request. After performing the conversion, the ADC returns the result.

The conversion result is also saved in the corresponding data register and it is compared with watchdog thresholds if requested.DMA and interrupt feature of ADC can also be used during CTU conversion.

The BCTU and ADCD can work on asynchronous clock domain. The signals ctu_trigger, ctu_nextcmd, ctu_push are toggle based, meaning any toggle (high-to-low or low-to-high) means one active command. The ADCD internally synchronizes the toggle and generates a trigger. The channel number needs to be fixed until the next active trigger. Similarly, the data remains valid from one toggle to the next toggle of the push command. The speed is marginally slower in asynchronous operation than synchronous due to synchronization overhead.

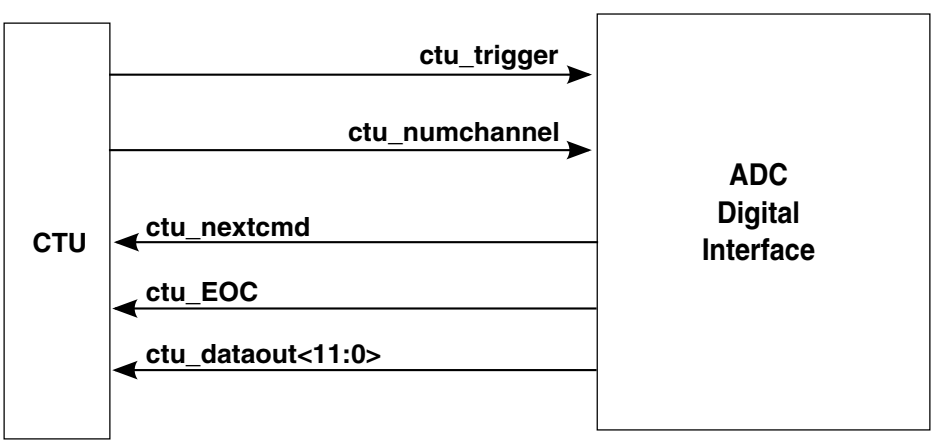


Figure 35-158. ADC Crosstriggering Unit

The BCTU interface has two modes of operation: Trigger and Control.

The BCTU interface is enabled by programming the MCR[CTUEN] bit. The BCTU interface operating mode is programmed by MCR[CTU_MODE] bit.

35.2.5.2.1 BCTU Trigger mode

In trigger mode, normal and injected conversions are enabled along with BCTU conversion, allowing all kinds of conversions to run.

When a trigger is received, the channel number is taken as the channel value and the BCTU conversion starts. The MSR[CTUSTART] bit is set automatically at this point and it is also automatically reset when the BCTU conversion is completed.

BCTU conversions should be requested after calibration is over. If a BCTU trigger is received during calibration, the calibration fails and BCTU conversion is started.

If a BCTU trigger is received during an ongoing injected conversion, then the injected chain is aborted immediately and only the BCTU conversion proceeds. The MSR[JSTART] bit is reset. Aborting of the injected conversion is indicated through the status bit of MSR[JABORT].

If a BCTU trigger is received during a ongoing normal conversion, the ongoing normal channel is suspended and the BCTU conversion is processed. Normal conversion resumes from the suspended channel after completion of BCTU triggered conversion.

If another BCTU trigger is received before ctu_nextcmd signal, that triggered conversion request is discarded.

If a Normal conversion is requested during BCTU conversion (MSR[CTUSTART] bit equal to ‘1’), the Normal conversion starts after the BCTU conversion is completed (MSR[CTUSTART] bit reset).

Injected conversion requests are discarded during a BCTU conversion, and the MCR[JSTART] bit is immediately reset.

35.2.5.2.2 BCTU Control mode

In BCTU Control mode, if enabled via MCR[CTUEN], the CPU is able to write in the ADC registers, but it cannot start a conversion. Conversion requests can be generated only by a BCTU trigger. If a normal or injected conversion is requested when the BCTU is enabled (MCR[CTUEN]), it will be automatically discarded.

Along with BCTU trigger the ctu_numchannel is taken as channel and BCTU conversion starts. MSR[CTUSTART]is set automatically at this point and it remains set unless the BCTU is disabled by resetting MCR[CTUEN].

BCTU conversions must be requested when calibration is over. If a BCTU trigger is received during calibration, the calibration stops immediately in order to satisfy the BCTU request. Calibration will fail in this case.

If another BCTU trigger is received before the ctu_nextcmd signal, the new request is discarded.

35.2.5.3 ADC clock prescaler and sample time settings

The clock (SAR_CLK) provided to the ADC's SAR controller, which controls the ADCA, must satisfy particular conditions of frequency and duty cycle. The maximum acceptable frequency is 80 MHz with a duty cycle equal to 50% (±5%).

The SAR_CLK frequency can be scaled by programming the MCR[ADCLKSEL] bit. If this bit is set, SAR_CLK frequency is the same as the bus clock. Otherwise, SAR_CLK frequency is half of the bus clock. MCR[ADCLKSEL] can only be written in power-down, when MCR[PWDN] = 1.

Sampling times are controlled by settings in the Conversion Timing Register (CTR*n*). A CTR register is provided for each channel type.

- CTR0 controls sampling time for *precision* channels:
  - 0 - 31 (ADC0)
  - 0 - 31 (ADC1)
  - 0 - 31 (ADC2)
  - 0 - 20 (ADC3)
- CTR1 controls sampling time for *internal* channels:
  - 32 - 39 (ADC0)
  - 32 - 38 (ADC1)
  - 32 - 36 (ADC2)
  - 32 - 38 (ADC3)
- CTR2 controls sampling time for *external* channels:
  - none (ADC0)
  - none (ADC1)
  - none (ADC2)
  - 64 to 95 (ADC3)

The exception is the Temperature Sensor channel, which always uses the CTR1 value. See Conversion Timing Register 1 (ADC_CTR1) for details.

35.2.5.4 Presampling

Presampling allows the ADC internal capacitor to precharge or discharge to a defined level before it starts the actual sampling/conversion of the analog input coming from the pads. This is useful for resetting information (history effect/offset) regarding the last converted data. During presampling, the ADC samples the internally generated voltage while, during sampling, the ADC samples the analog input coming from the pads.

Presampling can be enabled/disabled on an individual channel by setting the corresponding bit in the applicable PSR*n* register.

- PSR0 controls precision channels
- PSR1 controls internal channels
- PSR2 controls external channels

After enabling the presampling for a channel, the normal sequence of operation is presampling + sampling + evaluation for that channel. Sampling of the channel can be bypassed by setting PSCR[PRECONV]. When sampling of a channel is bypassed, the presampled voltage will be converted.

See Conversion time for the conversion's timing equation.

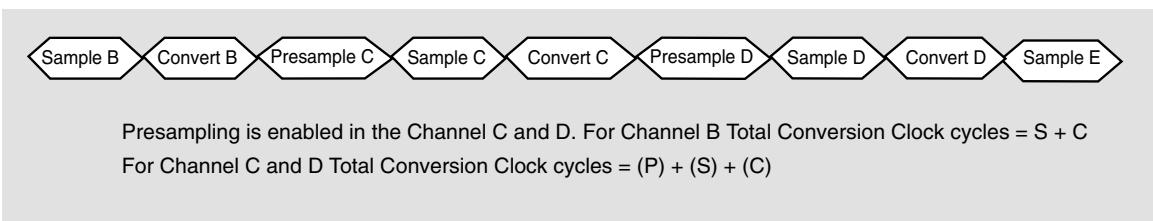


Figure 35-159. Presampling sequence

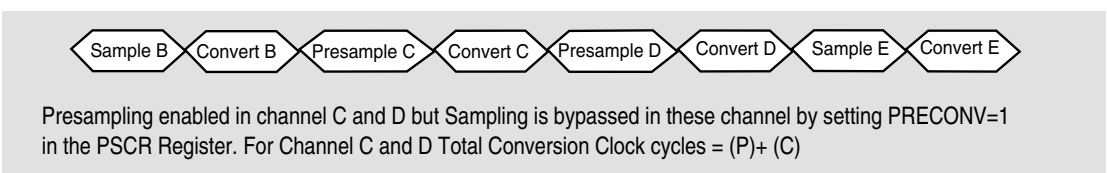


Figure 35-160. Presampling sequence with PRECONV = 1

35.2.5.4.1 Presampling channel enable

The presampling channels are selected by programming the presampling register (Presampling Control Register (ADC_PSCR)). These are used to enable analog switches that samples an internally generated voltage. It is possible to select presample voltage from four internally available voltages depending on the value of PREVAL*x* [*x* = 0,1,2] fields in the PSCR register as given in Table 35-159.

Table 35-159. Presample voltage selection

PREVAL*n*	Presampling voltage
00b	Presample voltage 1: VSS_HV_ADV_SAR
01b	Presample voltage 2: VDD_HV_ADV_SAR/8
10b	Presample voltage 3: VSS_HV_ADR_SAR
11b	Presample voltage 4: VDD_HV_ADR_SAR

The PREVALn fields are associated with specific channels, as shown in Table 35-160.

Table 35-160. PREVAL*n* channel association

Field	Channels
PREVAL0	• 0 - 31 (ADC0) • 0 - 31 (ADC1) • 0 - 31 (ADC2) • 0 - 20 (ADC3)
PREVAL1	• 32 - 39 (ADC0) • 32 - 38 (ADC1) • 32 - 36 (ADC2) • 32 - 38 (ADC3)
PREVAL2	• none (ADC0) • none (ADC1) • none (ADC2) • 64 to 95 (ADC3)

The temperature sensor channel is an exception. It can be mapped with any of the channels, but it uses the configuration of extended internal channel. So when this is selected, ADC takes PREVAL1 value to select the presample voltage because its physical properties may be different than the other internal precision channels.

35.2.5.5 Programmable analog watchdog

NOTE

The number of analog watchdogs and channels may vary depending on the ADC instance on your chip. Please see the chip-specific section for how many are available on each instance.

The analog watchdogs are used for determining whether the result of a conversion lies within a given guard area (as shown in Figure 35-161) specified by an upper and a lower threshold value named THRH and THRL, respectively.

After the conversion of the selected channel, a comparison is performed between the converted value and the threshold values. If the converted value lies outside the guard area, then the corresponding threshold violation interrupt is generated.

The comparison result is stored as WTISR[WDGxH] and WTISR[WDGxL], as explained in the following table. Depending on the WTIMR[MSKWDGxL] and WTIMR[MSKWDGxH] mask bits, an interrupt is generated upon threshold violation.

Table 35-161. Values of WDGxH and WDGxL fields

WDGxH	WDGxL	Converted data
1	0	converted data > THRH
0	1	converted data < THRL
0	0	THRH ≤ converted data ≤ THRL

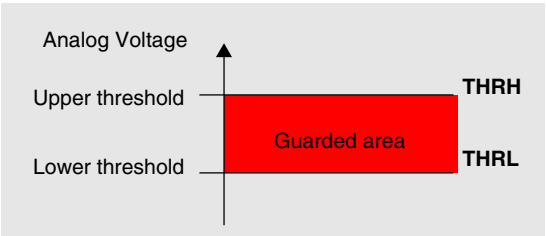


Figure 35-161. Guarded Area

35.2.5.5.1 Analog watchdog setting

For each watchdog there is one Threshold value (Low and High) Register (THRHLR).

The analog watchdog for each channel can be enabled independently by programming CWENR*n* register bits.

- CWENR0 controls *precision* channels:
  - 0 - 31 (ADC0)
  - 0 - 31 (ADC1)
  - 0 - 31 (ADC2)
  - 0 - 20 (ADC3)
- CWENR1 controls *internal* channels:
  - 32 - 39 (ADC0)
  - 32 - 38 (ADC1)
  - 32 - 36 (ADC2)
  - 32 - 38 (ADC3)
- CWENR2 controls *external* channels:
  - none (ADC0)
  - none (ADC1)
  - none (ADC2)
  - 64 to 95 (ADC3)

The availability of these registers depends on device configuration.

Threshold values for each channel can be selected independently from a maximum of 16 threshold registers (THRHLRx,(x=0..15)) by the WSEL_CHx field of CWSELRx (x=0..11) register.

Each CWSELRx register holds selection for 8 consecutive channels. CWSELR0 holds 8 WSEL_CH fields for channel 0 to 7. Likewise CWSELR1 is for channel 8 to 15 and so on. The availability of fields and registers depends on device configuration.

If the converted value for a particular channel lies outside the range specified by threshold values, then the corresponding bit is set in the Analog Watchdog Out of Range Register (AWORR).

For example, if channel number 12 is to be monitored with the threshold values in register THRHLR3, then CWSELR1[WSEL_CH4] needs to be programmed with 3. The enabling will be done by setting the bit corresponding to channel 12 in the CWENR0 register.

A set of threshold values (THRHLRx) can be linked to several ADC channels. The threshold values to be selected for a channel needs be programmed only once in the CWSELRx register.

35.2.5.6 DMA functionality

A Direct Memory Access (DMA) request can be programmed after the conversion of every channel by setting the respective masking bit in the following register(s):

- DMAR0
- DMAR1
- DMAR2

The DMA masking registers must be programmed before starting any conversion.

The DMA transfers can be enabled using the ADC_DMAE[DMAEN] bit. When the DMAE[DCLR] bit is set, the DMA request is cleared on the reading of the register for which DMA transfer has been enabled.

35.2.5.7 Interrupts

ADC generates the following maskable interrupt signals:

- End-of-conversion interrupt (EOC) request
- End-of-chain interrupt (ECH) request
- End-of-injected conversion interrupt (JEOC) request

- End-of-injected chain interrupt (JECH) request
- End-of-CTU conversion interrupt (EOCTU) request
- Watchdog threshold interrupt (WDGxL and WDGxH) requests

Interrupts are generated during the conversion process to signal events, like the end of a conversion, as explained in the register description for the Interrupt Status Register (ISR). The ISR and the Interrupt Mask Register (IMR) allow you to check and enable the interrupt request.

Interrupts can be individually enabled on a channel-by-channel basis by programming the Channel Interrupt Mask Register (CIMR).

Three Channel Pending Registers (CEOCFR*n*) are also provided in order to signal which of the channels' measurement has been completed.

Analog Watchdog interrupts are enabled by Watchdog Threshold Interrupt Mask Register (WTIMR) and the status can be checked at Watchdog Threshold Interrupt Status Register (WTISR). The watchdog interrupt source sets two fields, WDGxH and WDGxL, for each channel monitored in the WTISR.

In particular, the EOC, ECH, EOCTU, JEOC, and JECH interrupt lines are combined (OR-ed) on the same line. The EOC interrupt request is cleared by clearing either ISR[EOC] or all bits of CEOCFR. The JEOC interrupt request is cleared by clearing either ISR[JEOC] or all bits of CEOCFR. The EOCTU interrupt request is cleared by clearing either ISR[EOCTU] or all bits of CEOCFR. Also, the self-test end of algorithm interrupts STSR1[WDG_EOA_S] and STSR1[WDG_EOA_C] are combined.

All WDGxH and WDGxL requests are combined on the second line.

Another interrupt on the third line is a combination of WDSERR, WDTERR, ERR_S0, ERR_S1, ERR_S2, and ERR_C in STSR1.

The ISR register contains the interrupt pending request status. If you want to clear a particular interrupt event status, then writing a 1 to the corresponding status bit will clear the pending interrupt flag. (At this write operation, all the other bits of the ISR register must be programmed to 0.)

35.2.5.8 External Decode signals Delay

The ADC provides three external decode signals to be used to select one channel out of eight channels in external Analog multiplexers. There can be a maximum of four such multiplexers that are used for the 32 external channels to be converted. In order to take

into account the control of switching time of the external analog mux, a Decode Signals Delay (DSD) register is provided. This register allows you to program the delay between the decoding signal selection and the actual start of conversion.

If presampling is enabled then the DSD delay starts after the presample phase (presampling time is same as sample time).

35.2.5.9 Power-Down mode

The analog part of the ADC can be put in Power-Down mode by setting the MCR[PWDN] bit, which shuts off the ADCA and stops the clock in the SAR controller. After power-on or device reset, the ADCA is kept in Power-Down mode by default, so this state must be exited before starting any operation by resetting the MCR[PWDN] bit. In Power-Down mode, no conversion and calibration can be started. Trigger from BCTU during power down is discarded. In the ADC power-down mode, the BCTU must not start any conversion.

The MCR[PWDN] bit may be programmed high at any time. However, if a conversion is ongoing, then the ADCA cannot be sent immediately into Power-down mode. In fact, the ADC enters Power-down mode only after completing the ongoing conversion. Otherwise, the ongoing operation should be aborted manually by resetting the NSTART bit and using the ABORTCHAIN function.

The ADC status in MSR[ADCSTATUS] field shows Power-down status only when ADC enters in Power-down mode.

If BCTU is enabled and MSR[CTUSTART] is 1, then the MCR[PWDN] bit cannot be set. When BCTU Control mode is enabled, the application needs to reset the CTUEN bit before entering Power-Down mode.

After the power-down phase is complete, the process ongoing before the power-down phase must be restarted manually .

NOTE

Resetting the MCR[PWDN] bit and setting the MCR[NSTART] or MCR[JSTART] bit during the same cycle is forbidden.

35.2.5.10 Auto-Clock-Off mode

To reduce power consumption in Idle mode (without going into Power-Down mode), an auto-clock-off feature can be enabled by setting MCR[ACKO]. When enabled, the analog clock is automatically switched off when no operation is ongoing (that is, when no conversion has been programmed).

NOTE

This feature must remain off during calibration.

35.2.5.11 Calibration and Self Test

The ADC is calibrated and tested runtime through the same set of test conversions. The test result is used for computation and stored during the calibration process and only checked in self test. The two modes of running tests are as follows:

1. High-Accuracy mode/Calibration. Needs to be run after power-on reset and whenever required in runtime operation. Calibrates the ADC and updates calibration-related registers.
2. Quick-Check mode/Self-Test. Used for runtime functional self-tests to improve reliability. Catches any runtime fault or accuracy shift from previously calibrated values. Please see the chip-specific configuration information as self-test may not be available in all ADC instances.

The following sections describe both and the steps for executing them.

35.2.5.11.1 Calibration (High Accuracy mode)

The functioning of the analog block of ADC depends on the accuracy of values of its various parts and parameters, such as the value of capacitors, the gain of the amplifier, and so on. Due to variations in manufacturing and various runtime environmental effects, the actual values may differ from the ideal values for which the ADC is targeted.

Because of these deviations, the ADC's converted values contain errors. To eliminate these errors, the ADC must be calibrated prior to any conversion.

In the calibration process, a fixed known reference voltage (VrefH) is sampled many times and converted under various controlled conditions to check for deviations between these converted values and predefined values.

The deviations, known as offsets or modified values, are used to eliminate errors during the data processing of normal conversions.

Note

The ADC must be recalibrated if the operating conditions (particularly VrefH) change significantly. Calibration should also be performed again if the Self-Test indicates that a recalibration is needed.

Recommended settings:

- Clock frequency = 40 MHz
- Averging – Maximum (512)

The calibration process will take ~12ms of time with the above recommended settings.

NOTE

If the ADC clock frequency is > 40MHz during calibration, the accuracy of the calibration process will be compromised, hence the performance of the ADC. Thus, it is strongly recommended to run the calibration with 40 MHz clock.

As calibration is a time consuming process, it can be performed after a long interval – probably several hours of operation – but it is not necessary to recalibrate frequently. The goal is to maximize the possible accuracy delivered from conversion and to catch any structural fault to flag the error.

The calibration routine sets the fail flag (CALBISTREG[TEST_FAIL]) to indicate:

- A fault in ADC
- A premature abort of the calibration

Calibration is aborted if a normal conversion is initiated during calibration; this is called *premature termination*. The terminating conversion and the immediate next one to it will have invalid result in this case.

CALBISTREG[TEST_FAIL] = 1 indicates that the ADC health is not good and that it will not meet specifications.

To execute this routine, follow these steps:

1. Configure the ADC to 40 MHz.
   - Put the ADC in power down mode: set MCR[PWDN]
   - Configure clock divider selection: reset MCR[ADCLKSEL]
   - Power-up the ADC: reset MCR[PWDN]
2. Configure the Calibration, BIST Control, and Status Register (CALBISTREG). The default values are set for maximum accuracy (recommended).
3. Start calibration: set CALBISTREG[TEST_EN].

4. Wait for calibration to complete. Check MSR[ADCSTATUS] or till CALBISTREG[C_T_BUSY] becomes 0.
5. Check the value of MSR[CALIBRTD] to determine whether calibration was successful.
6. If MSR[CALIBRTD] = 1, calibration was successful; otherwise, it failed, and CALBISTREG[TEST_FAIL] = 1.
7. If required, reconfigure the ADC clock frequency for normal operation.

35.2.5.11.2 Self test (Quick-Check/Safety mode)

For devices used for very critical applications requiring high reliability it is important to check at regular intervals if the ADC is functioning correctly. For this purpose, Self Testing feature (Quick Check) has been incorporated inside ADC.

The following testing algorithms are available for checking:

- Supply Self test: algorithm S. It includes the conversion of the supply and VREF voltages. It includes a sequence of three test conversions (steps). The supply test conversions must be an atomic operation (no functional conversions interleaved).
- Capacitive Self test: algorithm C. It includes a sequence of 12 test conversions (steps) to check the capacitive array.

The capacitive test of ADC requires only 12 steps to test itself.

The capacitive array is used as DA converter to generate from V_{REFH} 12 different voltages by distributing and partially discharging portions of the capacity array. These 12 voltages are sampled using some redundant capacities. C algorithm may be compared to providing 12 different voltages to ADC and validate the conversion result.

Two kinds of tests are performed in this mode, as shown in the following table.

Table 35-162. Quick-Check mode tests

STCR3 [ALG]	STCR3 [MSTEP]	Description	Outcome	Comment
00 supply test	0	Supply self-test (Reserved)	Reserved	—
	1	Supply self-test (analog supply)	Measures the analog supply voltage (VDDA) and divides the result by the bandgap voltage (result from Algorithm S step 0). The VDDA conversion result is written to ADC_STDR1[TCDATA]. The final result (after division) consists of an integer portion and a fractional portion, both of which are written to the ADC_STDR2 register.	The least significant 3 bits of the analog supply (VDDA) conversion executed during Algorithm S step 1 processing and written to ADC_STDR1[TCDATA]) are always '0'. The conversion performed in Algorithm S Step 1 is not the full scale analog supply. Instead, the voltage sampled is VDDA / 8. The prescaling is done to avoid issues that can arise in the analog portion of the ADC when VREFH = VDDA = 3.3 V, or VDDA is slightly higher than VREFH. The conversion result is multiplied by 8 within the digital part of the ADC to scale the result to the expected level, which results in the 3 least significant bits having a value of '0'.
	2	Supply self-test (reference voltage high)	Measures the high reference voltage of the ADC	—
01	Reserved	—	—	—
10 capacitive test	0–11	One of the ADC-BIST steps is being run	The difference or error of individual value from the previously calibrated value is being returned (unsigned) as an ADC result that is compared with a programmed value in the analog watchdog register(s) to detect any fault. The difference or error may be caused by any runtime fault, runtime accuracy shift, or device noise.	Ideally, the result value is expected to be 0. If the returned value is higher, this indicates runtime fault or accuracy shift. This indicates that ADC should be recalibrated in High-Accuracy mode to check whether the reported error can be handled by recalibrating or whether permanent damage to ADC has occurred.
11	Supply test followed by Capacitive test in same sequence as shown above	—	—	—

This mode is described in the following sections.

35.2.5.11.2.1 Self test initialization and application information

ADC samples the internal V_{REFH} for the C algorithm and samples different supplies (V_{DD} and V_{REFH}) for the supply algorithm. It also provides signals for the following:

- Scheduling self-testing algorithms using configuration registers
- Monitoring the converted data using analog watchdog registers
- Flagging the error to FCCU in case a failure occurs in any of the algorithms

35.2.5.11.2.2 CPU mode

In this case, the test channel works similarly to a normal conversion. The test channel is enabled by setting STCR2[EN].

Self-test conversions are executed by interleaving the Normal conversions. The sequencing of steps of the selected algorithm for test channel depends on the operating mode of normal conversion, selected by MCR[MODE].

In One-Shot mode, if the test channel is enabled, only one step of the selected self-testing algorithm is executed at the end of the chain. The step number and algorithm to be executed are programmed in the STCR3 register. So, in One-Shot mode, the sequence is as follows:

1. Program NCMR*n* register(s) to select channels to be converted for normal conversion.
2. Modify the Sample period of normal conversion to 20 d cycles, write ADC_CTRx[INPSAMP] = 20 d (for 1 MSPS operation)
3. Program MCR[MODE] = 0 to select One-Shot mode.
4. Program sampling duration for self test conversion in STCR1[INPSAMPx].
5. Select the self-testing algorithm in STCR3[ALG]. The default is algorithm S.
6. Enable the self-testing channel by setting STCR2[EN].
7. Start the normal conversion by setting MCR[NSTART].
8. All normal conversions are executed as usual.
9. At the end of allnormal conversions, the step number programmed in STCR3[MSTEP] of the self-testing algorithm selected by STCR3[ALG] is executed similar to a normal functional channel.
10. On receiving an end-of-conversion for the test channel, the digital result is written in STDR1[TCDATA] andSTDR1[VALID] is set. Also, ISR[EOC], ISR[ECH], and STSR1[ST_EOC] are set. See the register descriptions (Self Test Data Register 1 (ADC_STDR1), Interrupt Status Register (ADC_ISR), and Self Test Status Register 1 (ADC_STSR1)) for details of settings for each test.
11. The state machine returns to the Idle state.

For example: in a device with channels A-B-C-D-E-F-G-H, channels B-D-E are to be converted in One-Shot mode. MODE = 0 is set for One-Shot mode. Refer to Conversion. Conversion for channels B-D-E are done. After channel E test channel conversion is done and ISR[ECH] and ISR[EOC] are set.

MSR[NSTART] is automatically set when the normal conversion starts; it is reset at the end of conversion for the test channel.

In Scan mode, the consecutive steps of the selected self test algorithm are converted continuously at the end of each chain of normal conversions. The number of channels converted at the end of each chain is 1 (except for algorithm S, in which all the steps are performed at once without any functional conversion interleaved). So, in Scan mode the sequence is as follows:

1. Program the NCMR*n* registers to select channels to be converted for normal conversion.
2. Modify the Sample period of normal conversion to 20 d cycles, write ADC_CTRx[INPSAMP] = 20 d (for 1 MSPS operation)
3. Program MCR[MODE] = 1 to select Scan mode.
4. Program sampling duration for self test conversion in STCR1[INPSAMPx].
5. Select the self-testing algorithm in STCR3[ALG]. By default, all algorithms are selected (that is, all algorithms are executed step-by-step one after the other).
6. Enable the self-testing channel by setting STCR2[EN].
7. Start the normal conversion by setting MCR[NSTART].
8. All normal conversions are executed as usual.
9. At the end of chain forthe normal conversions (assuming a default value of STCR3[ALG]), all steps of algorithm S are performed (as algorithm S is always atomic). MSR[SELF_TEST_S] is set.
10. At the end of conversion of test channel for the last step of algorithm S, the digital result is written in STDR1[TCDATA] and STDR1[VALID] is set. At the same time, MSR[SELF_TEST_S] is reset. For step 1, the integral part and fractional part are written in STDR2[IDATA] and STDR2[FDATA]. Also, ISR[EOC], ISR[ECH], and STSR2[ST_EOC] are set.
11. The next normal conversion chain starts.
12. At the end of the normal conversion chain, Step0 of the C algorithm is executed.
13. At the end of conversion of test channel for Step0 of the C algorithm, the digital result is stored in STDR1[TCDATA] and STDR1[VALID] is set (if MCR[OVERWR] is set). Also, ISR[EOC], ISR[ECH], andSTSR2[ST_EOC] are set.
14. The next normal conversion chain starts.
15. At the end of the normal conversion chain, step1 of C algorithm is executed.
16. This process continues for all steps of all algorithms.
17. The state machine returns to the Idle state when MCR[NSTART] is written to 0.

NOTE

- Instead of starting a normal conversion with software (by setting MCR[NSTART]), if it is started by external trigger, the test channel behavior remains the same.
- Test channel conversion is not performed during injected conversions. It is performed only during normal conversions.
- If, during a test channel conversion, injection conversion arrives, then the test channel is aborted (just as a normal functional channel) and injected conversions are done. After injected conversions are completed, the test channel resumes from the step at which it was aborted. In this case, MSR.SELF_TEST_S remains high during the injected conversion.
- For self testing, MCR[MODE]should be programmed at least one cycle *before* setting MCR[NSTART]. It should not be changed thereafter until a conversion is ongoing.

35.2.5.11.2.3 Abort and Abort Chain for self-test channel

Setting MCR[ABORT] during self-test channel has no effect.

In One-Shot mode, if MCR[ABORTCHAIN] is set when the test channel conversion is ongoing, the test channel is aborted and ECH is set. In this case, EOC for the test channel is not generated.

In Scan mode, if MCR[ABORTCHAIN] is set when test channel STEP N is ongoing, then the test channel STEP N is aborted and the next chain conversion starts. At the end of this chain, STEP N conversion is performed again. (If algorithm S is ongoing, the full algorithm is executed again.)

35.2.5.11.2.4 Self-test analog watchdog

ADCD also provides a monitor for the values returned by the ADC analog macro for self-test algorithms. The analog watchdogs are used to determine whether the result of conversion for self-test algorithms lie in a particular guard area. For this purpose, separate Self-Test Analog Watchdog Registers are provided for each algorithm.

If the analog watchdog feature is enabled (STAWxR[AWDE] = 1), a comparison is performed between the converted value and the threshold values after the conversion of each step of an algorithm. If the converted value does not lie between the upper and

lower threshold values specified by the Analog Watchdog Register of the particular algorithm, the corresponding error bit (STSR1[ERR_x]) is set and the step number during which the error occurred is updated in STSR1[STEP_x] (in case of the C algorithm) and erroneous data is written in STSR4[DATAx]. STSR1[ERR_x] generates an interrupt if enabled by the corresponding mask bit in STCR2. The fault indication is also given to the FCCU via the Critical and Non-Critical fault lines, so that necessary action can be taken at chip level.

The analog watchdog feature works differently in the case of algorithm S. As already mentioned, algorithm S is always an atomic operation. So, separate error bits are provided in STSR1 for each step of algorithm S to avoid an overwrite in case an error occurs in more than one step. Hence, there are separate mask bits for each step in STCR1. For the same reason, separate fields in the status registers (STSR2 and STSR3) exist to store erroneous data for each step.

In step 1 of supply algorithm, ADC measures the Analog supply (Vdda) for the ADC, and then a fixed point division is performed with the result of step 1 (ADC supply voltage) and step 0 (band gap voltage). It takes around 26 cycles after EOC for step 1 to get a divided value, and it is the divided value on which analog watchdog checks are applied. So, the value to be compared for step 1 contains the integer as well as fractional part; two registers (STAW1AR and STAW1BR) are provided for this. The comparison is done first for the integer part using the threshold values programmed in STAW1AR. If the integer part lies in the range, the fractional part comparison is skipped; otherwise, it is compared with the values programmed in STAW1BR. The following table summarizes this feature for step 1.

Table 35-163. Algorithm S (step 1) threshold comparison

STDR2[IDATA] (integer part)	STDR2[FDATA] (fractional part)	STSR1[ERR_S1]
> STAW1AR[THRH]	Any value	Set
< STAW1AR[THRL]	Any value	Set
== STAW1AR[THRH]	> STAW1BR.THRH	Set
== STAW1AR[THRL]	< STAW1BR.THRL	Set

In step 2 of supply algorithm, ADC measures the High Reference voltage (Vrh); (VREF/ VREF) is measured in order to check the integrity of sampling signal. For this particular conversion, no higher threshold value is required as the ideal value is 0xFFF. Only the lower threshold value is programmed in STAW2R.

In the supply self-test algorithm, the band gap is measured with regard to the reference voltage (Vrh) in step 0. The result is capable of catching mismatches between them. Step 1 measures the supply with regard to reference (Vrh), and the result is taken to do a ratio

(Vdda/Vbg). If the band gap voltage remains within +/-1.25% of its ideal value of 1.2 V, then by setting appropriate values of THRH and THRL in watchdog registers, the ADC self-test can catch more than 4% variation in supply and reference faithfully.

Algorithm C has total 12 steps. Output data of step 0 is compared with the threshold values stored in STAW4R, if enabled in the same register. The result of step 0 is used as an offset for rest of the steps (step 1 to 11). The absolute difference between the converted data of each of the steps (1 to 11) with the data of step 0 are compared with the threshold values of STAW5R, if enabled in STAW4R. Error in any step is indicated by STSR1[ERR_C] and the data in error case is captured in STSR4.

Calculation of Threshold values:

S1 Algorithm: Calculated voltage comparison method:

The user software can implement a method of calculating S1 algorithm IDATA, FDATA, the THRH values, and the THRL values to voltages and then perform a comparison of the calculated voltage to the calculated THRH voltage and calculated THRL voltage.

The steps involved in this method are as follows:

1. Calculate the high threshold voltage:

   (STAW1AR [THRH] + (STAW1BR [THRH] / 4096)) * Vbandgap

2. Calculate the low threshold voltage:

   (STAW1AR [THRL] + (STAW1BR [THRL] / 4096)) * Vbandgap

3. Calculate the converted results to voltage:

   (STDR2 [IDATA] + (STDR2 [FDATA] / 4096)) * Vbandgap

4. Compare result from step 3 to THRH and THRL.S1 algorithm passes

   If (low threshold voltage <= converted voltage <= high threshold voltage).

35.2.5.11.2.5 Watchdog timer

The watchdog timer is an additional check that monitors the sequence of the self-testing algorithm implemented and checks that the algorithm completes within a safe time period. Each algorithm has a different watchdog timer that runs independently of the other. The watchdog timer for a particular algorithm can be enabled by setting STAWxR[WDTE]. The safe time value can be programmed in STBRR[WDT] (the default value is 10 ms, assuming an 80 MHz clock).

The safe time is measured starting from step 0 of the algorithm (including all normal chain conversions in between) to the point at which step 0 of the same algorithm starts again.

The sequence is as follows:

1. Program NCMR0 to select channels to be converted for normal conversion in Scan mode (MODE = 1).
2. Modify the Sample period of normal conversion to 20 d cycles, write ADC_CTRx[INPSAMP] = 20 d (14 h) (for 1 MSPS operation)
3. Select the self-testing algorithm in STCR3[ALG]. By default, all algorithms are selected; that is, all algorithms will be executed step-by-step, one after the other.
4. Enable the self-testing channel by setting STCR2[EN].
5. Program the safe period value in STBRR[WDT].
6. Enable the watchdog timer by setting STAWxR[WDTE]. Assume the setting of WDTE to be t0. (It is important to do all the programming first and to then enable WDTE as the safe time check is also performed between the setting of WDTE and the start of step 0 to check that algorithm has started within the safe time).
7. Start the normal conversion by setting MCR[NSTART].
8. After first chain conversion ends, step 0 of algorithm S is executed. Lets assume the start of STEP0 to be t1.
9. Step 1 and step 2 of algorithm S are executed.
10. The next chain conversions are performed.
11. When chain conversion completes, step 0 of C algorithm is performed.
12. After each chain conversion, a consecutive step of the C algorithm is performed. A similar sequence follows for algorithm C.
13. After the last step of the C algorithm is performed, another chain conversion is executed. At the end of this chain conversion, step 0 of algorithm S starts, repeating the whole sequence. Let's assume this time (starting of step 0) to be t2.
14. For algorithm S, if (t1 – t0) > Safe Period, or if (t2 – t1) > Safe Period, then the watchdog timer flags an error and STSR1[WDTERR] is set. A critical fault is asserted and an interrupt is also generated if enabled by STCR2[MSKWDTERR]. Otherwise, the watchdog timer counter is reset and starts again to monitor the same for the next sequence.
15. A similar sequence is followed for watchdog timers for algorithm C.

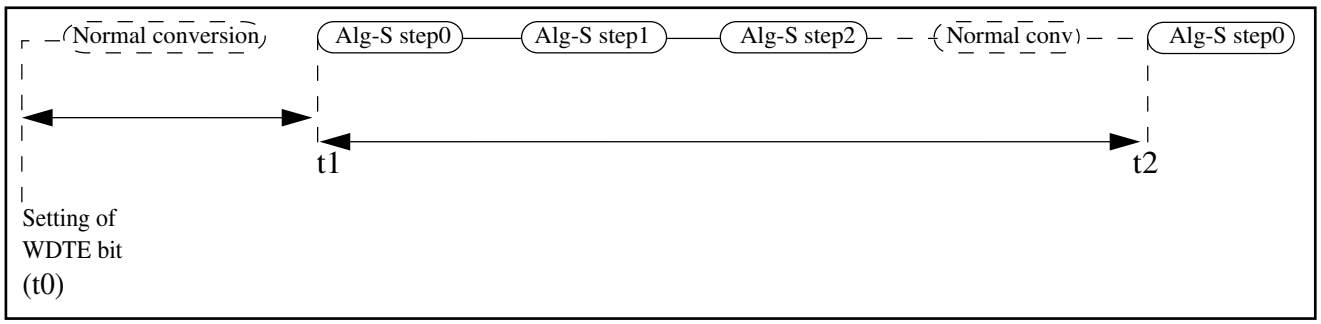


Figure 35-162. Watchdog timer monitor for algorithm S

Notes:

- Take care not to enable the watchdog timer for an algorithm that is not to be executed.

35.2.5.11.2.5.1 Watchdog sequence checking

The watchdog timer also incorporates sequence checking features that check the order of a particular algorithm's steps. If the steps are not in order, then an error is flagged by setting the STSR1[WDSERR]. A critical fault is asserted and an interrupt is also generated if enabled by STCR2[MSKWDSERR].

A watchdog sequence error is flagged in the following cases:

- If the steps of any algorithm are not executed in the proper order.
- If an abort chain occurs during a test channel conversion, that step has to be repeated at the end of the next chain. This will give a sequence error as soon as test channel conversion starts again.
  - Exception: If an abort chain occurs during the last step of algorithm S, then sequence errors are not flagged as the whole algorithm has to be repeated again.

If injected conversion occurs during the test channel, watchdog sequence errors are *not* flagged a lthough the ongoing step number is aborted and repeated again.

Note

The watchdog timer feature is applicable only for Scan mode, not for One-Shot mode.

35.2.5.11.2.6 Baud rate control for test channel

It defines the scheduling of the test channel between normal conversions. The scheduling rate is specified by STBRR[BR].

By default, if the test channel is enabled, one step of the selected algorithm is executed after every chain of normal conversion. The bandwidth consumed by the test channel depends on the number of channels in a normal chain. For example, if there are 100 normal conversions in a chain, then the test channel consumes only 1% of the total bandwidth, which is very small. However, if the number decreases to just four channels, then the bandwidth consumed by the test channel is 25%, which is significant (and may not be desirable as it slows down the normal conversion rate).

STBRR[BR] provides flexibility by scheduling the test channel conversion to be performed, not at the end of every chain, but at the end of BR + 1 number of chains. For example, if BR = 5, a single step of the selected algorithm for the test channel is performed after six chain conversions, then the next step is performed at end of the next six chain conversions, and so on. By default, the value of BR is 0.

Notes:

- This feature is applicable only for Scan mode, not for One-Shot mode. STBRR[BR] should be set to 0 for one shot mode.
- To use the baud rate feature in Scan mode, the NCMR register should have nonzero value.

35.2.5.11.2.6.1 Abort chain when baud rate is non-zero

As previously described, a nonzero value of STBRR[BR] means that the test channel conversion is performed at the end of BR + 1 number of chains. If an abort chain occurs during the chain in which the test channel is scheduled to be converted, then the test channel is converted after the next BR + 1 number of chains.

For example, if STBRR[BR] is programmed to 2, the sequence will be two normal chains (without any test channel conversion) followed by a chain with the test channel converted at the end. Now, if an abort chain occurs during first two chains it is treated as a normal chain abort and the test channel is converted at the end of the third chain only (as the case without any abort chain). But if the abort chain occurs during the third chain in which the test channel is scheduled to be converted, then the test channel is converted after next three chains (that is, at the end of sixth chain, counting from the beginning).

35.2.5.12 Conversion time

Total conversion time depends on ADC-SAR controller's operating clock. This clock can be prescaled to bus clock ÷ 2. The ADC-SAR operating clock (SAR_CLK) is always synchronous with bus clock. The frequency of SAR_CLK is dependent on settings of ADC_MCR[ADCLKSEL].

If ADC_MCR[ADCLKSEL] = 0, then SAR_CLK = (bus clock ÷ 2)

If ADC_MCR[ADCLKSEL] = 1, then SAR_CLK = bus clock

The various components of conversion time and affecting configurations are shown and described in Figure 35-163.

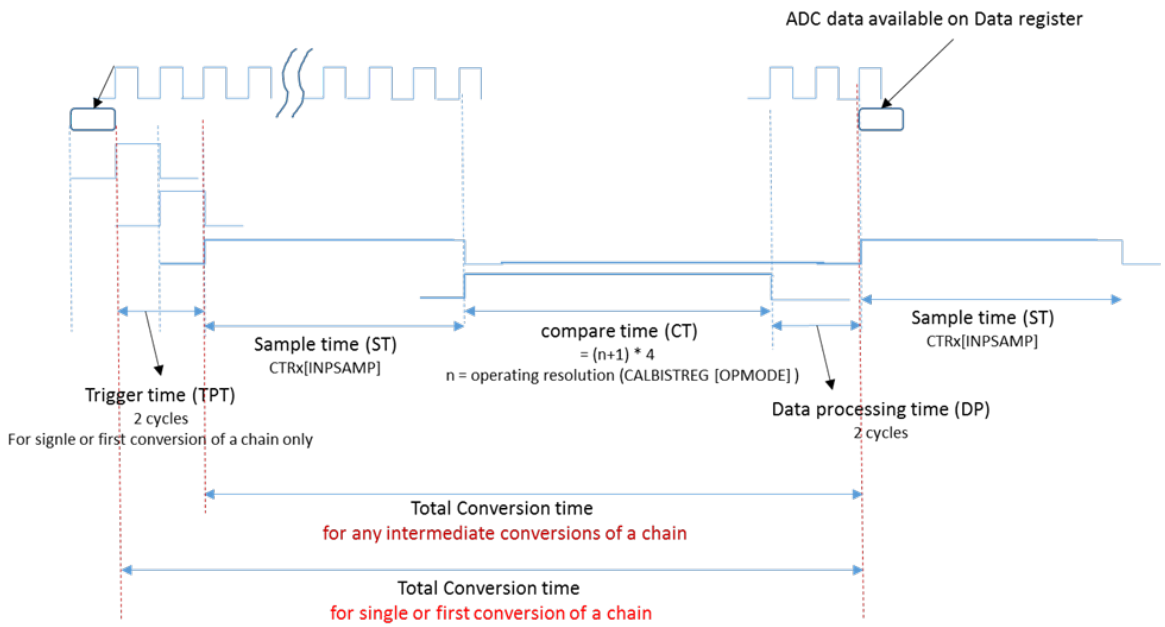


Figure 35-163. Conversion time

- Trigger processing time (TPT):
  To prepare the channel and start the operation, two clock cycles of bus clock is required by the ADC for the first conversion and for subsequent conversions in a chain, or for a continuous conversion, this time is not required as it is hidden in the pipeline operation. Triggers from Synchronous CTU interface requires two cycle of bus clock for first conversion and for back-to-back CTU conversions it takes only one cycle for trigger processing as the other cycle is hidden in pipeline operation. Triggers from Asynchronous CTU interface requires three cycles of bus clock for synchronization and processing.
- Sample phase time (ST):
  Sample time duration is controlled by the INPSAMP[7:0] field of Conversion timing Registers ADC_CTR*x* (*x* = 0...2) for different types of channels. Value in each register affects the sample time in similar way. The value in the register represents units of cycle of the SAR_CLK. The minium value of sample time is 8. If the value programmed is less than 8, then it has no effect on sample time duration and in this case sample time will be 8 SAR_CLK cycles.
- Compare phase time (CT):

The compare phase of the ADC dependends on the resolution setting (which is 13 for this chip), and takes ((13 + 1) × 4) cycles of SAR_CLK.

- Data processing time (DP):

  The ADC takes 2 cycles of SAR_CLK to post process and load the data int registers.
- Total Conversion Time:

  The total time required for single or first conversion is [TPT+ST+CT+DP] cycles of AD_clk and for subsequent conversion is a chain or continuous conversion is [ST+CT+DP] cycles of AD_clk.

  Examples:

  - CTR*x*[INPSAMP] = 20d; software/hardware triggered (NSTART):
    - Single/First Conversion: 2 + 20 + ((13 + 1) × 4) + 2 = 2 + 20 + 56 + 2 = 80 cycles of SAR_CLK
    - Subsequent/continuous conversion: 20 + ((13 + 1) × 4) + 2 = 20 + 56 + 2 = 78 cycles of SAR_CLK
  - CTR*x*[INPSAMP] = 20d; CTU triggered:
    - Single/First Conversion: 2 + 20 + ((13 + 1) × 4) + 2 = 2 + 20 + 56 + 2 = 80 cycles of SAR_CLK
    - Subsequent/continuous conversion: 1 + 20 + ((13 + 1) × 4) + 2 = 1 + 20 + 56 + 2 = 79 cycles of SAR_CLK
- Pre-Sampling:

  When pre-sampling is enabled, the ADC does pre-sampling prior to starting the actual channel sampling as described in pre-sampling section. This phase takes CTR*x*[INPSAMP] cycles of SAR_CLK. Additionally, to switch from pre-sample phase to actual channel sampling phase, it takes another two cycles of SAR_CLK. So, total CTR*x*[INPSAMP] + 2 cycles additionally required for a conversion to be completed. Thus, in case of pre-sample, all the equations and examples above are valid with the addition of (CTR*x*[INPSAMP] + 2) cycles.

35.2.5.12.1 Conversion time for BCTU interface

The BCTU can work in a different clock domain, and for this there is an asynchronous domain crossing that occurs before the CTU trigger is processed. Data is also processed after domain crossing. These times need to be considered in the conversion time calculation. The synchronization may take a maximum of two additional ADC clock cycles and three BCTU clock cycles over normal conversion time.

The total conversion time taking BCTU timing into consideration is as follows:

Total conversion time = Synchronization time + TPT + ST + CT + DP + Reverse synchronization time in BCTU

The figure below shows the different components of total conversion time.

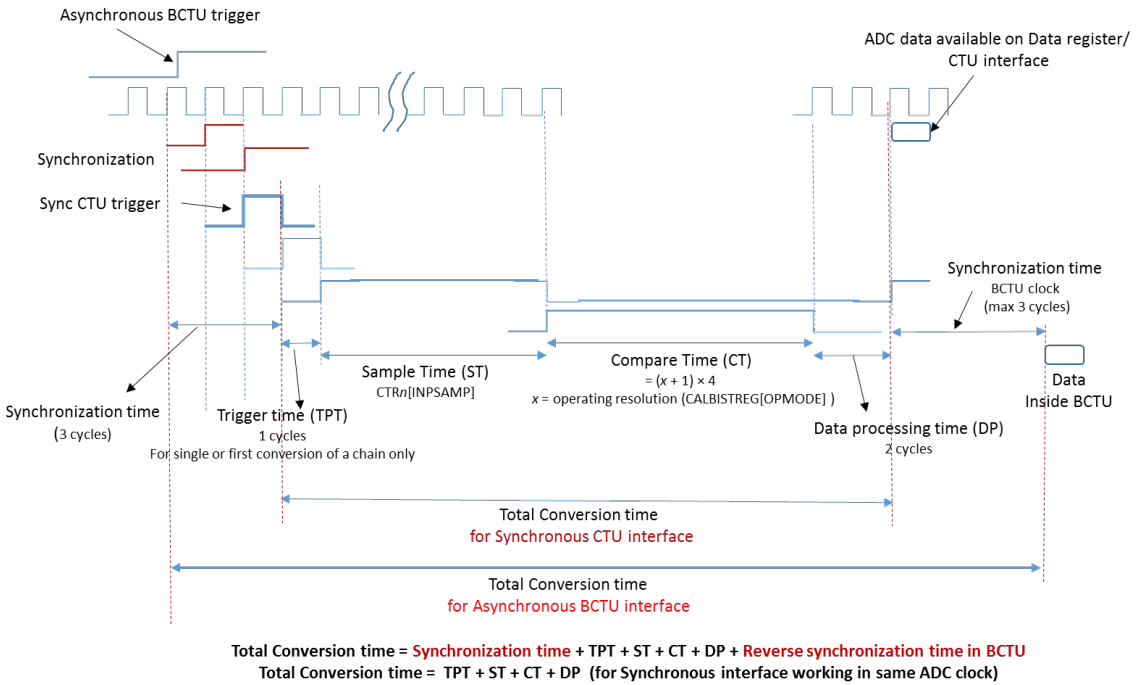


Figure 35-164. Normal Conversion time from BCTU (asynchronous, not working in ADC clock)

35.2.5.13 Conversion data processing

The raw converted ADC data contains many types of errors such as offset, gain, DC bias, and so on. Thus, to generate error-free results, raw converted data is processed before it is written to a result register. The process of error correction goes bit-by-bit during conversion with the values generated during the offset calculation and calibration process. This helps in reducing final data processing time as only one addition/addition (2's complement) is performed parallely with conversion.

35.2.6 Clock frequency

The ADC internal blocks are designed to work with an 80 MHz clock. Thus, the specified speed can be met with this clock frequency. If the clock is faster than this, ADCA cannot work properly and the ADC may malfunction. If the clock is slower, then the ADC speed

will be lower, but it will work properly. It should be noted that there is an absolute maximum value of the sampling time that the analog portion of the ADC can support. For slower clocks, the sampling clock count will need to be programmed accordingly.

Chapter 36
Decimation Filter (DECFILTER)

36.1 Decimation Filter chip-specific module information

The device has two instances of the Decimation Filter.

- The Decimations are used for digital signal processing on the SAR and SD ADC conversion results.
- Two Decimation Filters can be used in cascaded mode.
- There is no direct hardware interface from the SAR and Sigma-Delta ADCs to the Decimation Filters on the device.
- Transfer of ADC conversion results to the Decimation Filters for application processing is handled entirely by the DMA and CPU.
- All interrupts are connected to the Interrupt Controller according to Interrupt sources.
- Similarly all DMA requests are connected to appropriate DMA channel muxes according to DMA request mapping.

36.2 Introduction

This chapter describes the Decimation Filter block.

36.2.1 Overview

The decimation filter is a dedicated hardware block, designed to decimate fixed point sample conversion results. An interface is provided for use by the CPU, allowing setup of the filter parameters and read/write of the configuration registers and filter samples.

The Decimation Filter receives data samples from the master blocks (CPU) in the RX sub-block. Each sample arrives at the decimation filter with an identifier tag and associated commands. The input information is decoded by the RX and control logic sub-blocks. When receiving a filtering command, the data is transferred to the filter tap register's sub-block and is processed by the filter using the MAC (Multiply and

Accumulate) unit, the coefficient register, and the control logic sub-blocks. The result is then returned to the master block by the TX sub-block, accompanied by the corresponding tag information that provides an address for the data.

The decimation filter functions in a standalone mode. In this mode, the input data is supplied and the output results are read by the chip core processor (CPU) using status and interrupt signals or DMA requests.

An integrator unit independently accumulates the values of filter outputs. The integration can be restricted to time windows defined by hardware signals or software.

Two or more decimation filter blocks can also be configured to work in the cascade mode of operation to obtain a more complex filtering function. The output result of a filter block is connected to the input of the next filter by means of a dedicated interface.

All signals in the interface are generated in the system clock domain.

Figure 36-1 shows the block diagram for the decimation filter.

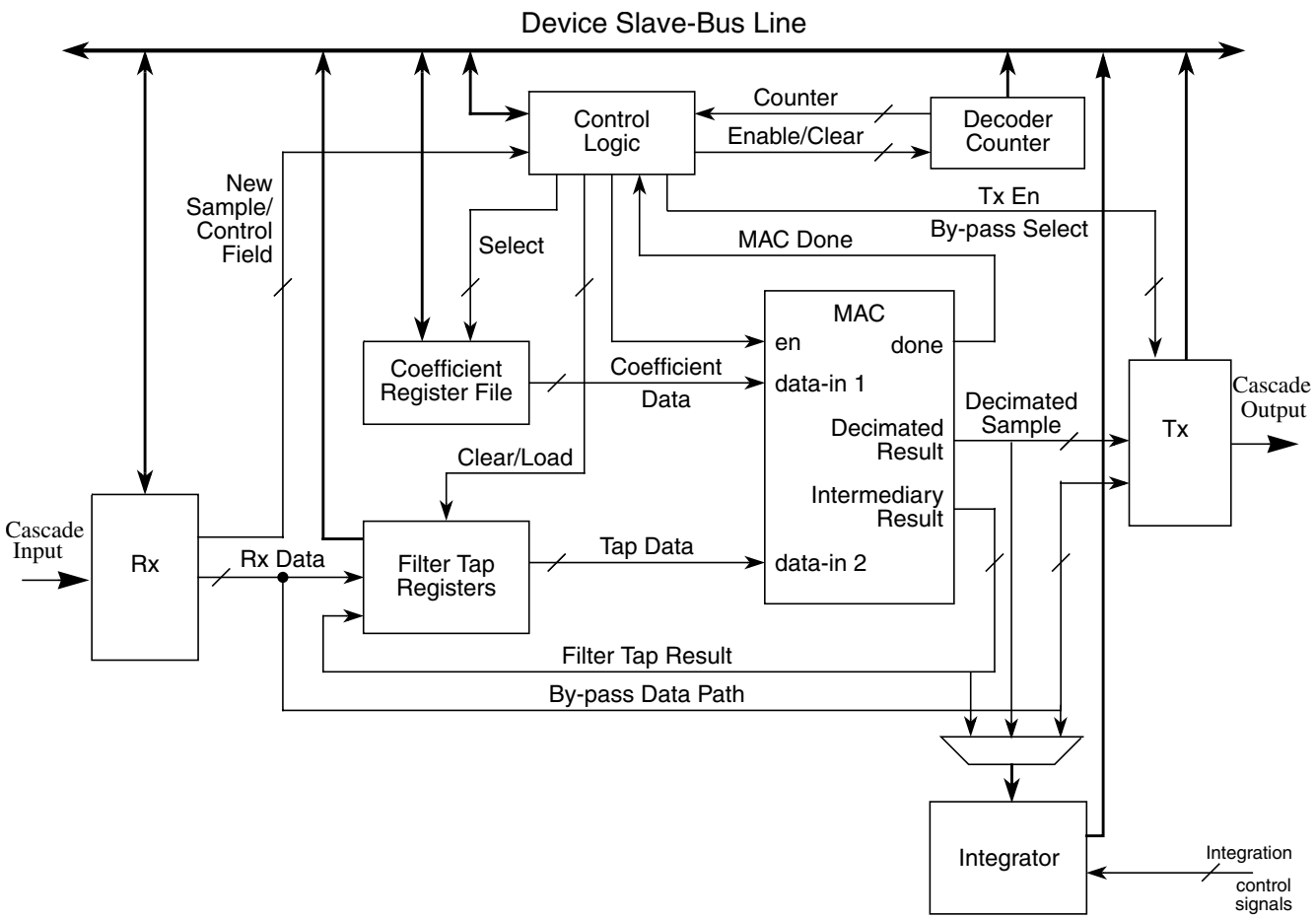


Figure 36-1. Decimation Filter Block Diagram

36.2.2 Features

The Decimation Filter block includes these distinctive features:

- Selectable 4th order IIR filter, or an 8th order FIR filter
  - Input/output with 16-bit (fixed point) two's complement signed values
  - Internal taps with 16-bit (feed-forward portion of first IIR) and 24-bit (feedback portion) resolutions (fixed point) for two's complement signed value
  - 24-bit programmable filter coefficients (fixed point) for two's complement signed value
  - MAC unit with 51-bit fixed point accumulator
  - Convergent rounding methodology
  - Two's complement overflow or saturation selection
  - 58 clock cycles to process the input
- Input and output buffers with DMA capability
- Slave-bus interface to device
- Filter taps access for debug
- Filter initialization (flush) and stabilization (prefill) commands
- Decimation controlled by an internal counter
- Integrator unit accumulates filter output values, signaled or absolute, with 32-bit resolution. The integrator can be controlled by software or hardware signals.
- Cascade of two or more individual blocks to compose a more complex filter.

36.2.3 Modes of operation

This section describes the operation modes of the Decimation Filter. The modes are selected using the following DECFILTER_MCR register fields: MDIS, FREN, FRZ, ISEL,, and CASCD (see Module Configuration Register (DECFILTER_MCR)).

The mode selection is summarized in the following table.

Table 36-1. Operation Mode Selection

Mode	MDIS	FREN, FRZ	ISEL	CASCD
Standalone	0	(0, 0)	1	00
Cascade		or (0, 1) or (1, 0)	0 or 1	01 or 10 or 11
Freeze[1]		1, 1	X	X
Low Power	1	X	X	X
Reserved	All other combinations			

1. Freeze mode can also be activated from outside the Decimation Filter if FREN=1.

36.2.3.1 Standalone mode

Data is provided by the central processor using the device slave-bus interface or DMA interface signals. Once the data is filtered the decimated result is available in the Output Buffer register. The filter output is also consumed by a CPU or DMA mastering the same device slave-bus interface. This operation mode can be used to debug the filter stability or to decimate data in System RAM.

36.2.3.2 Cascade mode

Cascade mode is a filter structure mode with two or more individual filter blocks connected in a chain to form a more complex filter function. The output result of the first block (head block) is connected to the input of the next block (middle or tail block) to be filtered again. More details in Cascade mode description.

36.2.3.3 Low Power mode

Low power mode corresponds to the module disable mode or stop mode. In the module disable mode the filter input is disabled and it is not possible to enter Freeze mode. The system clock is stopped. In stop mode, the system clock is also stopped.

36.2.3.4 Freeze mode

Freeze mode is also known as debug mode. All filter action is frozen, either through software or by the hardware SoC debug request signal. If a freeze request comes when the filter is processing an input, it enters freeze mode only after the processing finishes.

36.3 External signal description

Note

The Decimation Filter does not provide metastability protection or filtering for these signals.

36.3.1 Integrator enable signal

The integrator enable signal is used to enable the hardware integrator. For more details, see Integrator Enabling and Halting.

36.3.2 Integrator halt signal

The integrator halt signal is used to halt the hardware integrator. For more details, see Integrator Enabling and Halting.

36.3.3 Integrator reset signal

The integrator reset signal is used to reset the hardware integrator. For more details, see Integrator Reset.

36.3.4 Integrator output request signal

The integrator output request signal is used to update the integrator output result. For more details, see Integrator Outputs.

36.4 Memory map and register definition

This section provides the memory maps and detailed descriptions of all registers. Accesses to reserved areas of the memory map can return a bus error.

This module communicates with the slave-bus line that is related to the device integration.

The addresses of the Decimation Filter registers are specified as offsets from the module's base address, described in the following table. The registers allocated in this memory map are sufficient for a 4th order IIR filter implementation.

All registers are 32-bit wide.

NOTE

Reserved fields in the registers should not be written.

DECFILTER memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (DECFILTER_MCR)	32	R/W	See section	36.4.1/1141
4	Module Status Register (DECFILTER_MSR)	32	R/W	0000_0000h	36.4.2/1145
8	Module Extended Config Register (DECFILTER_MXCR)	32	R/W	0000_0000h	36.4.3/1149
C	Module Extended Status Register (DECFILTER_MXSR)	32	R/W	0000_0000h	36.4.4/1152
10	Interface Input Buffer (DECFILTER_IB)	32	R/W	0000_0000h	36.4.5/1155
14	Interface Output Buffer (DECFILTER_OB)	32	R	0000_0000h	36.4.6/1156
20	Filter Coefficient (DECFILTER_COEF0)	32	R/W	0000_0000h	36.4.7/1157
24	Filter Coefficient (DECFILTER_COEF1)	32	R/W	0000_0000h	36.4.7/1157
28	Filter Coefficient (DECFILTER_COEF2)	32	R/W	0000_0000h	36.4.7/1157
2C	Filter Coefficient (DECFILTER_COEF3)	32	R/W	0000_0000h	36.4.7/1157
30	Filter Coefficient (DECFILTER_COEF4)	32	R/W	0000_0000h	36.4.7/1157
34	Filter Coefficient (DECFILTER_COEF5)	32	R/W	0000_0000h	36.4.7/1157
38	Filter Coefficient (DECFILTER_COEF6)	32	R/W	0000_0000h	36.4.7/1157
3C	Filter Coefficient (DECFILTER_COEF7)	32	R/W	0000_0000h	36.4.7/1157
40	Filter Coefficient (DECFILTER_COEF8)	32	R/W	0000_0000h	36.4.7/1157
78	Filter TAP (DECFILTER_TAP0)	32	R	0000_0000h	36.4.8/1157
7C	Filter TAP (DECFILTER_TAP1)	32	R	0000_0000h	36.4.8/1157
80	Filter TAP (DECFILTER_TAP2)	32	R	0000_0000h	36.4.8/1157
84	Filter TAP (DECFILTER_TAP3)	32	R	0000_0000h	36.4.8/1157
88	Filter TAP (DECFILTER_TAP4)	32	R	0000_0000h	36.4.8/1157

Table continues on the next page...

DECFILTER memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8C	Filter TAP (DECFILTER_TAP5)	32	R	0000_0000h	36.4.8/1157
90	Filter TAP (DECFILTER_TAP6)	32	R	0000_0000h	36.4.8/1157
94	Filter TAP (DECFILTER_TAP7)	32	R	0000_0000h	36.4.8/1157
E0	Final Integer Value Register (DECFILTER_FINTVAL)	32	R	0000_0000h	36.4.9/1158
E4	Final Integer Count Register (DECFILTER_FINTCNT)	32	R	0000_0000h	36.4.10/ 1159
E8	Current Integer Value Register (DECFILTER_CINTVAL)	32	R	0000_0000h	36.4.11/ 1159
EC	Current Integer Count Register (DECFILTER_CINTCNT)	32	R	0000_0000h	36.4.12/ 1160

36.4.1 Module Configuration Register (DECFILTER_MCR)

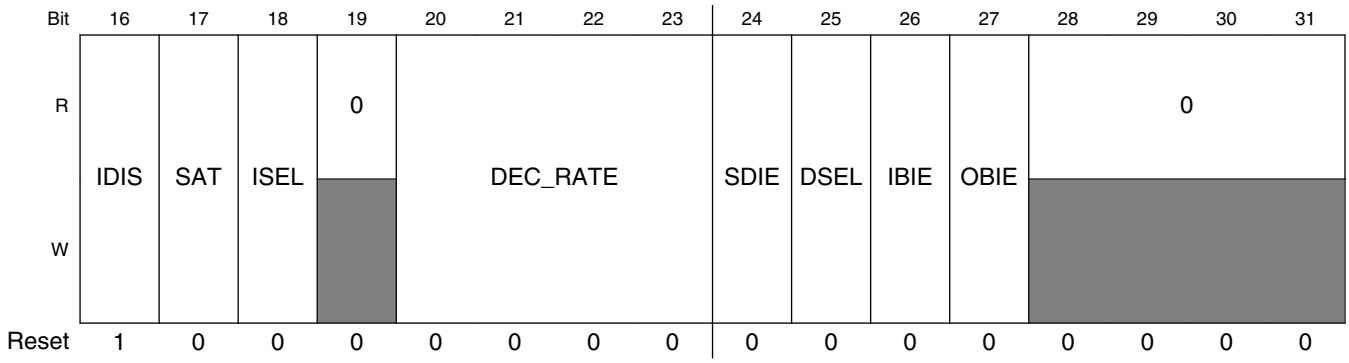
The Decimation Filter module configuration register provides configuration control bits for the Decimation Filter internal logic.

NOTE

One must not modify this register contents when the status bit BSY is set, except for fields FREN, FRZ, and IDIS. To guarantee that BSY does not set during the read-modify-write operation, set IDIS=1 and wait for BSY=0 beforehand.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MDIS	FREN	0	FRZ	0	CASCD		IDEN	ODEN	ERREN	0	FTYPE		0	SCAL	
W					SRES											
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



DECFILTER_MCR field descriptions

Field	Description
0 MDIS	Module Disable The MDIS bit puts the Decimation Filter in low power mode. Writes to the configuration register are allowed with the exception of writes to the FREN and SRES bits, which are ignored. Writes to the Coefficient registers are also allowed. The Decimation Filter cannot enter Freeze mode once in disable mode. Once the module is disabled it no longer receives the system clock. 0 Normal Mode 1 Low Power Mode
1 FREN	Freeze Enable The FREN bit enables the Decimation Filter to enter freeze mode if the SoC debug request signal or the FRZ bit is asserted. See Freeze Mode Description for more details. 0 Decimation Filter Freeze mode disabled 1 Decimation Filter Freeze mode enabled
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 FRZ	Freeze Mode The FRZ bit controls the freeze mode of the Decimation Filter. For this bit to take effect the FREN freeze enable bit also needs to be asserted. While in freeze mode the MAC operations are halted. See Freeze Mode Description for more details. 0 Decimation Filter in Normal Mode 1 Decimation Filter in Freeze Mode
4 SRES	Software-reset bit The SRES is a self-negated bit which provides the CPU with the capability to initialize the Decimation Filter through the slave-bus interface. This bit always reads as zero. See Soft-Reset Command Description for more details. 0 No action 1 Software-Reset
5–6 CASCD	Cascade Mode Configuration The CASCD[1:0] bit field configures the block to work in cascade mode of operation. For more details about the cascade mode, see Cascade mode description.

Table continues on the next page...

DECFILTER_MCR field descriptions (continued)

Field	Description
	NOTE: Any change to this field must follow the procedure described in Cascade Freeze, Stop, and Configuration Change Procedures. 00 No cascade mode (single block) 01 Cascade Mode, Head block config. 10 Cascade Mode, Tail block config. 11 Cascade Mode, Middle block config.
7 IDEN	Input Data Interrupt Enable The IDEN bit enables the Decimation Filter to generate interrupt requests on all new input data written to the Interface Input Buffer register or Input/Output Buffers register. 0 Input Data Interrupt Disabled 1 Input Data Interrupt Enabled
8 ODEN	Output Data Interrupt Enable The ODEN bit enables the Decimation Filter to generate interrupt requests on all new data written to the filter Output buffer. It is independent of how ISEL is set. 0 Output Data Interrupt Disabled 1 Output Data Interrupt Enabled
9 ERREN	Error Interrupt Enable The ERREN bit enables the Decimation Filter to generate interrupt requests based on the assertion of the DECFILTER_MSR register error flags OVF, DIVR, SVR, OVR, or IVR. 0 Error Interrupts Disabled 1 Error Interrupts Enabled
10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–12 FTYPE	Filter Type Selection bits The FTYPE[1:0] bits select the filter type. **NOTE:** Bypass must not be configured in cascade mode (see field CASCD). 00 Filter Bypass In Bypass configuration the filter is disabled. 01 IIR Filter - 1 x 4th order 10 FIR Filter - 1 x 8th order 11 reserved
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 SCAL	Filter Scaling Factor The SCAL[1:0] bit field selects the scaling factor used by the filter algorithm. 00 Scaling Factor = 1 01 Scaling Factor = 4 10 Scaling Factor = 8 11 Scaling Factor = 16

Table continues on the next page...

DECFILTER_MCR field descriptions (continued)

Field	Description
16 IDIS	Input Disable The IDIS bit disables the block input, so that writes to the input buffer have no effect (from the device slave-bus), and input DMA or interrupt requests are not issued. Input disabling is needed to change the block configuration to or from cascade mode. **NOTE:** IDIS resets in 1 (hardware reset only), and the module configuration procedures for previous versions (without IDIS bit) are upward compatible provided that the DECFILTER_MCR register is written with 0 into the IDIS position (previously reserved). 0 Input enabled 1 Input disabled
17 SAT	Saturation Enable The SAT bit enables the saturation of the filter output. See Saturation for more details. 0 Disables Saturation 1 Enable Saturation
18 ISEL	Input Selection The ISEL bit selects the source of input data to the Filter. The data source is the CPU/DMA on the device slave-bus interface. Each device slave-bus write to the Interface Input Buffer register or DMA transfer to the input buffer is interpreted as a new sample to be processed by the filter. 0 Reserved 1 Filter input from the device slave-bus interface
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 DEC_RATE	Decimation Rate Selection The DEC_RATE[3:0] field selects the decimation rate used by the Decimation Filter. The decimation rate defines the number of data samples from the master block that is required to generate one decimated result in the Decimation Filter output. 0000 No Decimation: one filter output for each sample input 0001 - 1111 One filter output for each (DEC_RATE+1) sample inputs
24 SDIE	Integrator Data Interrupt Enable The SDIE field enables output buffer interrupts due to integrator data result being ready (at registers DECFILTER_FINTVAL and DECFILTER_FINTCNT): 0 Integration ready does not cause an output interrupt. 1 Integration ready causes an output interrupt
25 DSEL	DMA Selection The DSEL bit determines whether the data transfers — to the input buffer (write to) and from the output buffer (read from) — are performed by DMA requests or by interrupt requests. 0 Interrupt requests are generated 1 DMA requests are generated
26 IBIE	Input Buffer Interrupt Request Enable The IBIE bit enables the Decimation Filter to generate interrupt requests when:

Table continues on the next page...

DECFILTER_MCR field descriptions (continued)

Field	Description
	• deviceslave-bus input is selected (ISEL=1) and DSEL=0 when the input buffer is available to receive new data; 0 Input Buffer Interrupt Request Disabled 1 Input Buffer Interrupt Request Enabled
27 OBIE	Output Buffer Interrupt Request Enable The OBIE bit enables the Decimation Filter interrupt requests when outputs are directed to the device slave-bus and DMA is not selected (DSEL=0). 0 Output Buffer Interrupt Request Disabled 1 Output Buffer Interrupt Request Enabled
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

36.4.2 Module Status Register (DECFILTER_MSR)

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BSY	0	DEC_COUNTER				0	0	0	0	0	0	0	0	0	0
W							IDFC	ODFC		IBIC	OBIC			OVFC	OVRC	IVRC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						IDF	ODF	0	IBIF	OBIF	0	0	OVF	OVR	IVR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DECFILTER_MSR field descriptions

Field	Description
0 BSY	Decimation Filter Busy indication

Table continues on the next page...

DECFILTER_MSR field descriptions (continued)

Field	Description
	The BSY bit indicates that the Decimation Filter is processing new input data from the master block in normal mode or from the device core in standalone mode. BSY is not asserted when the filter is disabled (FTYPE = 00). However, the BSY bit is asserted when the soft reset is executed. 0 Decimation Filter Idle 1 Decimation Filter Busy
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–5 DEC_COUNTER	Decimation Counter The DEC_COUNTER[3:0] field indicates the current value of the DEC_COUNTER Decimation Counter (see Figure 36-1), which counts the number of input data samples received by the Decimation Filter. When the value of this counter matches the DEC_RATE[3:0] Configuration Register field, one decimated result is generated and the DEC_COUNTER counter is re-initialized at zero. This register is cleared by a soft reset or a flush command.
6 IDFC	Input Data Flag Clear bit The IDFC bit clears the IDF Flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears IDF
7 ODFC	Output Data Flag Clear bit The ODFC bit clears the ODF Flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears ODF
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 IBIC	Input Buffer Interrupt Request Clear bit The IBIC bit clears the IBIF Flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears IBIF
10 OBIC	Output Buffer Interrupt Request Clear bit The OBIC bit clears the OBIF Flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears OBIF
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 OVFC	OVF Clear bit

Table continues on the next page...

DECFILTER_MSR field descriptions (continued)

Field	Description
	The OVFC bit clears the OVF Output Overflow bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears OVF
14 OVRC	OVR Clear bit The OVRC bit clears the OVR Output Overrun bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears OVR
15 IVRC	IVR Clear bit The IVRC bit clears the IVR Filter Input Overrun indication bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears IVR
16–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 IDF	Input Data Flag The IDF bit flag indicates when new data is available at the DECFILTER_IB register or at the PSI bus data, input and output. This flag generates an Interrupt Request if enabled by the IDEN bit in the Configuration Register. This Flag is cleared by the IDFC Status bit or by a soft reset of the decimation filter. **NOTE:** OBS: This flag is not used for read / write requests. It is used only to announce the input data event. For read / write request flag, refer to IBIF. 0 Sample not received 1 New Sample received
23 ODF	Output Data Flag The ODF bit flag indicates when a new decimated sample is available at the DECFILTER_OB register or at the PSI bus data, input and output. This flag generates an Interrupt Request if enabled by the ODEN bit in the Configuration Register. This Flag is cleared by the ODFC Status bit or by a soft reset of the decimation filter. **NOTE:** OBS: This flag is not used for read requests. It is used only to announce the output data event. For read request flag, refer to OBIF. 0 No new Decimated Output Sample available 1 New Decimated Output Sample available
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 IBIF	Input Buffer Interrupt Request Flag The IBIF bit flag indicates that the input buffer DECFILTER_IB is available to be filled with new data. IBIF assertion also asserts the interrupt signal when enabled by the IBIE bit in the Configuration Register when DMA is not selected (DSEL=0). This Flag is cleared by the IBIC Status bit or by a soft reset of the decimation filter.

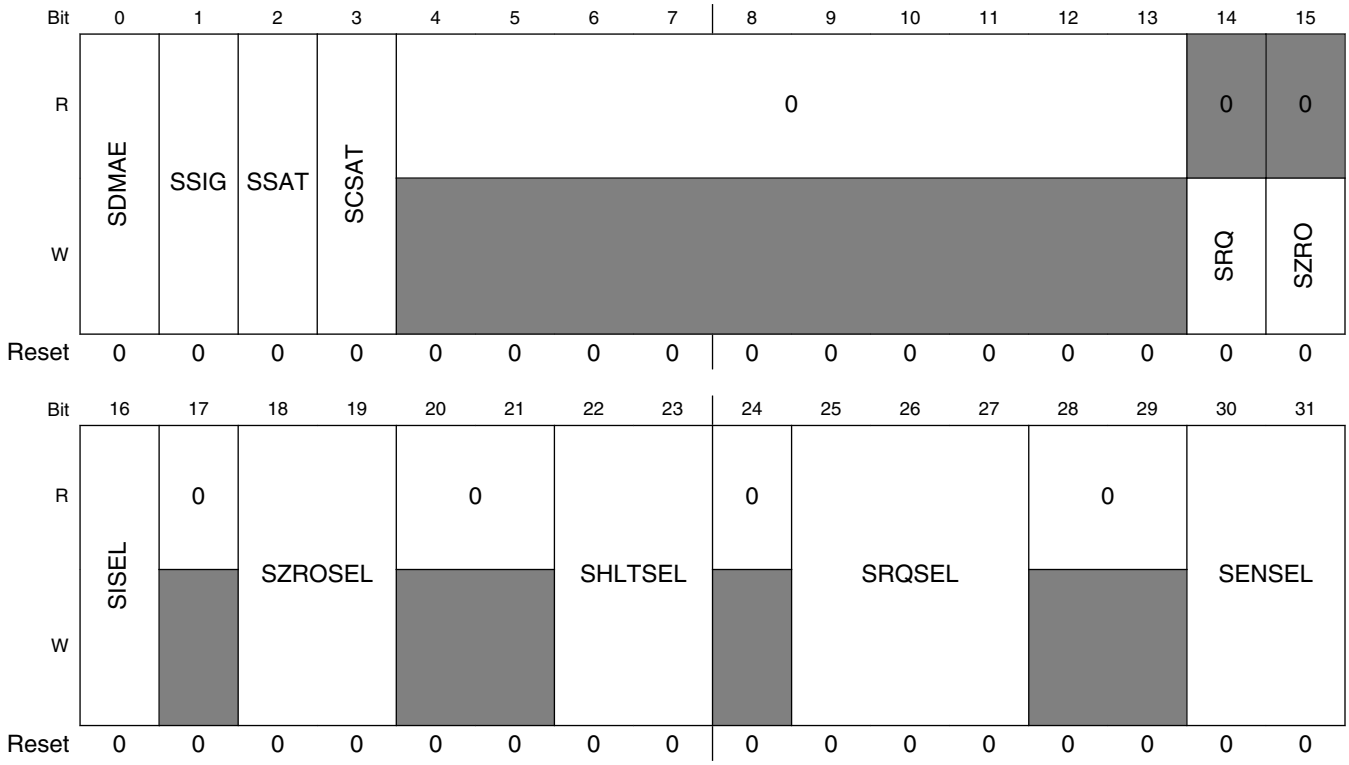
Table continues on the next page...

DECFILTER_MSR field descriptions (continued)

Field	Description
	0 No action 1 New Sample is requested.
26 OBIF	Output Buffer Interrupt Request Flag The OBIF bit flag indicates that a new decimated sample is available at the DECFILTER_OB register. This flag generates an Interrupt Request if enabled by the OBIE bit in the Configuration Register and with DSEL=0. This Flag is cleared by the OBIC Status bit or by a soft reset of the decimation filter. 0 No new Decimated Output available 1 New Decimated Output available
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 OVF	Filter Overflow Flag The OVF bit indicates that an overflow occurred in the filtered sample result. This flag generates an Interrupt Request if enabled by the ERREN bit in the Configuration Register. This Flag is cleared by the OVFC Status bit or by a soft reset of the decimation filter. 0 No overflow 1 Overflow occurred
30 OVR	Output Interface Buffer Overrun The OVR bit indicates that a decimated sample was overwritten by a new sample in the Interface Output Buffer Register. This flag generates an Interrupt Request if enabled by the ERREN bit in the Configuration Register. This Flag is cleared by the OVRC Status bit or by a soft reset of the decimation filter. 0 No Output Overrun 1 Filter Output Overrun occurred
31 IVR	Input Interface Buffer Overrun The IVR bit indicates that a received sample in the Filter Interface Input Register was overwritten by a new sample. This was probably caused by a violation of the Decimation Filter maximum throughput. This flag generates an Interrupt Request if enabled by the ERREN bit in the Configuration Register. This Flag is cleared by the IVRC Status bit or by a soft reset of the decimation filter. **NOTE:** IVR does not set due to input register writes when input is disabled (DECFILTER_MCR bit IDIS=1). 0 Input Buffer Overrun did not occur 1 Input Buffer Overrun occurred

36.4.3 Module Extended Config Register (DECFILTER_MXCR)

Address: 0h base + 8h offset = 8h



DECFILTER_MXCR field descriptions

Field	Description
0 SDMAE	Integrator DMA Enable The SDMAE bit enables the DMA request when an integrator output is requested (see Integrator Outputs). **NOTE:** The DMA channel used is the same one used for filter outputs, and any configuration that generates DMA requests from both of those sources is not allowed. 0 integrator DMA request disabled 1 integrator DMA request enabled
1 SSIG	Integrator Signal operation selection The SSIG bit defines how the filtered data signal is treated for integration: 0 integrator input takes the absolute value of filter output 1 integrator input takes signaled filter output
2 SSAT	Integrator Saturated operation selection The SSAT bit defines how the integrator accumulator behaves in case of an overflow.

Table continues on the next page...

DECFILTER_MXCR field descriptions (continued)

Field	Description
	NOTE: In saturated operation the overflown integration sum holds the value 0xFFFFFFFF for absolute integration (SSIG=0), or values 0x7FFFFFFF (positive saturation) and 0x80000000 (negative saturation) for signaled integration (SSIG=1). **NOTE:** Non-saturated mode is not supported with signaled integration, therefore one must not configure SSIG=1 and SSAT=0. 0 integrator accumulator holds a modulo 2^{17} value (considering the 15-bit fractional part) on an overflow. 1 integrator accumulator saturates on an overflow
3 SCSAT	Integrator Counter Saturated operation selection The SCSAT bit defines how the integrator sample counter behaves in case of an overflow. 0 integrator sample counter holds a modulo 2^{32} value on an overflow. 1 integrator sample counter saturates on an overflow, holding a value of 0xFFFFFFFF.
4–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 SRQ	Integrator Output Request The SRQ bit is used to command the update of the integrator output, reflected in the registers DECFILTER_FINTVAL and DECFILTER_FINTCNT. It may also cause a DMAor interrupt request, depending on the DECFILTER_MCR bit SDIE and DECFILTER_MXCR bit SDMAE. This is a write-only bit, so reads always return 0. For more details see Integrator Reset. 0 no integrator output update request 1 requests integrator output update
15 SZRO	Integrator Zero The SZRO bit is used to zero the integrator sum. This is a write-only bit, reads always return 0. For more details see Integrator Reset. **NOTE:** If bits SRQ and SZRO are both written 1 at the same time, the integrator is reset only after the registers DECFILTER_FINTVAL and DECFILTER_FINTCNT are updated. 0 does not zero integrator sum 1 zeroes integrator sum
16 SISEL	Integrator Input Selection The SISEL bit selects the input of the integrator. For more details see Integrator Inputs. 0 decimated filter outputs feed the integrator 1 filter outputs before the decimation feed the integrator
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–19 SZROSEL	Integrator Zero Control Mode Selection The SZROSEL field defines the use of the integrator zero hardware input signal. For more details see Integrator Reset. 00 hardware integrator zero request disabled 01 integrator zero on toggle of hardware signal 10 integrator zero on rising edge of hardware signal 11 integrator zero on falling edge of hardware signal

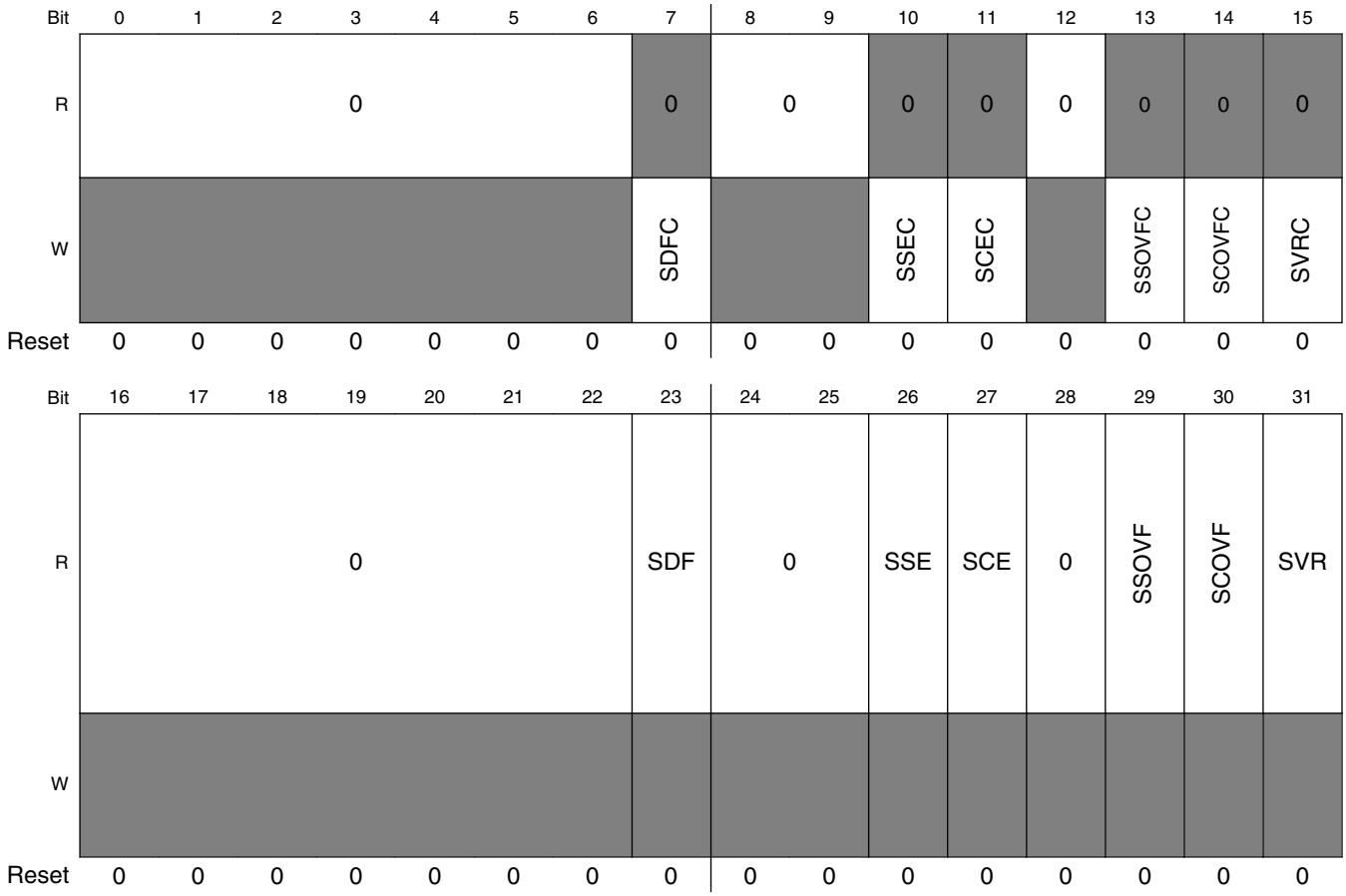
Table continues on the next page...

DECFILTER_MXCR field descriptions (continued)

Field	Description
20–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 SHLTSEL	Integrator Halt Control Selection The SHLTSEL field defines the integrator halting mechanism. When the integrator is halted, the integration accumulator remains unaltered on filter outputs independently of the enabling selected by SENSEL. For more details see Integrator Enabling and Halting. 00 hardware halt control signal disabled 01 integrator halted, independently of the hardware signal 10 integrator halted when signal is at logical 0 11 integrator halted when signal is at logical 1
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–27 SRQSEL	Integrator Output Read Request Mode Selection The SRQSEL field defines the use of the integrator output request hardware input signal. An integrator output request updates the registers DECFILTER_FINTVAL and DECFILTER_FINTCNT, also causing a DMA or interrupt request. Note that DMA or interrupt requests due to integrator output updates depend on the DECFILTER_MXCR bit SDMAE and DECFILTER_MCR bit SDIE. When continuous output is on, an integrator output request is issued whenever a new filter output is accumulated. For more details see Integrator Outputs. 000 hardware output request disabled 001 integrator output request on toggle of hardware signal 010 integrator output request on rising edge of hardware signal 011 integrator output request on falling edge of hardware signal 100 reserved 101 continuous output request on, independently of hardware signal 110 continuous output request on when signal is at logical 0 111 continuous output request on when signal is at logical 1
28–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 SENSEL	Integrator Enable Control Selection The SENSEL field defines the integrator enabling mechanism. When the integrator is enabled, filter outputs selected by the SISEL bit are added to the integration accumulator. When the integrator is disabled, the integration accumulator remains unaltered on filter outputs. For more details see Integrator Enabling and Halting. 00 integrator disabled, independently of the hardware enable control signal 01 integrator enabled, independently of the hardware signal 10 integrator enabled when signal is at logical 0 11 integrator enabled when signal is at logical 1

36.4.4 Module Extended Status Register (DECFILTER_MXSR)

Address: 0h base + Ch offset = Ch



DECFILTER_MXSR field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SDFC	Integrator Output Data Flag Clear bit The SDFC bit clears the SDF Flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears SDF
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 SSEC	Integrator Sum Exception Clear bit The SSEC bit clears the SSE flag bit in the Status Register. This bit is self negated, therefore it is always read as zero.

Table continues on the next page...

DECFILTER_MXSR field descriptions (continued)

Field	Description
	0 No action 1 Clears SSE
11 SCEC	Integrator Count Exception Clear bit The SCEC bit clears the SCE flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears SCE
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SSOVFC	Integrator Sum Overflow Clear bit The SSOVFC bit clears the SSOVF Flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears SSOVF
14 SCOVFC	Integrator Count Overflow Clear bit The SCOVFC bit clears the SCOVF Flag bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears SCOVF
15 SVRC	SVR Clear bit The SVRC bit clears the SVR Integrator Data Overrun indication bit in the Status Register. This bit is self negated, therefore it is always read as zero. 0 No action 1 Clears SVR
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SDF	Integrator Data Flag The SDF bit flag indicates when a new integrator result is available at the DECFILTER_FINTVAL register. This flag generates an Interrupt Request if enabled by the SDIE bit in the Configuration Register. This flag is cleared by the SDFC Status bit or by a soft reset of the decimation filter. 0 no new integrator result available 1 new integrator result available
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 SSE	Integrator Sum Exception flag The SSE bit indicates an exceptional condition of the integrator accumulator. This flag generates an Interrupt Request if enabled by the DECFILTER_MCR bit ERREN, and it is cleared by the SSEC bit or by a soft reset. Integrator exceptions are defined in section“Integrator Exceptions”.

Table continues on the next page...

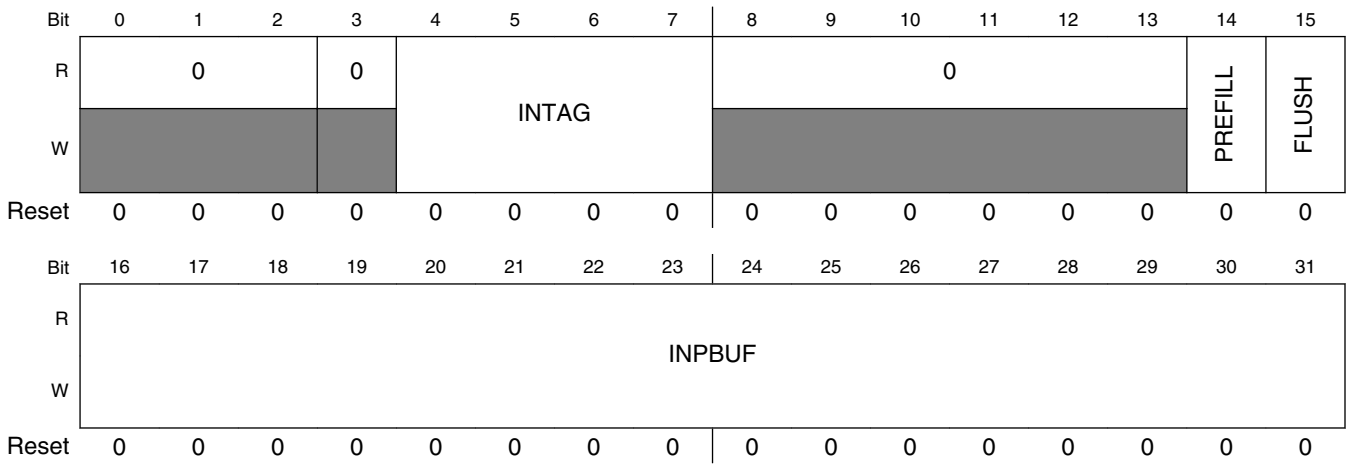
DECFILTER_MXSR field descriptions (continued)

Field	Description
	0 No exception in the integrator accumulator. 1 Integrator accumulator exception.
27 SCE	Integrator Count Exception flag The SCE bit indicates an exceptional condition of the integrator counter. This flag generates an Interrupt Request if enabled by the DECFILTER_MCR bit ERREN, and it is cleared by the SCEC bit or by a soft reset. Integrator exceptions are defined in section“Integrator Exceptions”. 0 No exception in the integrator counter. 1 Integrator counter exception.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 SSOVF	Integrator Sum Overflow Flag The SSOVF bit indicates an overflow of the integrator accumulator. This flag is cleared by the SSOVFC bit or by a soft reset. **NOTE:** The SSOVF bit samples the integrator accumulator overflow condition when and only when either registers DECFILTER_FINTVAL or DECFILTER_CINTCNT are updated. Therefore, only one of the register pairs (DECFILTER_FINTVAL/DECFILTER_FINTCNT and DECFILTER_CINTVAL/ DECFILTER_CINTCNT) must be used by the application, in order to avoid races. 0 No overflow in the integrator accumulator. 1 Integrator accumulator overflown.
30 SCOVF	Integrator Count Overflow Flag The SCOVF bit flag indicates an overflow of the internal integrated sample counter. This flag is cleared by the SCOVFC bit or by a soft reset. **NOTE:** The SCOVF bit samples the integrator accumulator overflow condition when and only when either registers DECFILTER_FINTVAL or DECFILTER_CINTCNT are updated. Therefore, only one of the register pairs (DECFILTER_FINTVAL/DECFILTER_FINTCNT and DECFILTER_CINTVAL/ DECFILTER_CINTCNT) must be used by the application, in order to avoid races. 0 No overflow in the integrator sample counter. 1 Integrator sample counter overflown.
31 SVR	Integrator Data Overrun The SVR bit indicates that an integration value and count in the registers DECFILTER_FINTVAL and DECFILTER_FINTCNT was overwritten by a new integrator output request and was not read by the CPU or DMA. This flag generates an Interrupt Request if enabled by the ERREN bit in the Configuration Register. This Flag is cleared by the SVRC bit or by a soft reset. 0 Integrator Data Overrun did not occur. 1 Integrator Data Overrun occurred.

36.4.5 Interface Input Buffer (DECFILTER_IB)

The Input Buffer Register provides access to the Input buffer of the decimation filter when the filter is in the standalone mode of operation. Writes to this register are interpreted as requests to the Decimation Filter to process new sample data.

Address: 0h base + 10h offset = 10h



DECFILTER_IB field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 INTAG	Decimation filter input tag bits The INTAG[3:0] bit field is defined as a selector signal and it is used to identify different destinations for the INPBUF[15:0] data.
8–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 PREFILL	Decimation Filter Prefill/Filter control bit The PREFILL bit selects the Decimation Filter operation mode. For more details, see Filter Prefill Control Description 0 Decimation Filter normal sample 1 Decimation Filter prefill sample
15 FLUSH	Decimation Filter Flush control bit Assertion of the FLUSH bit initializes the Decimation Filter to a initial state, as defined in Flush Command Description. This bit is self negated and it is cleared only when the data is read and the flush is executed.

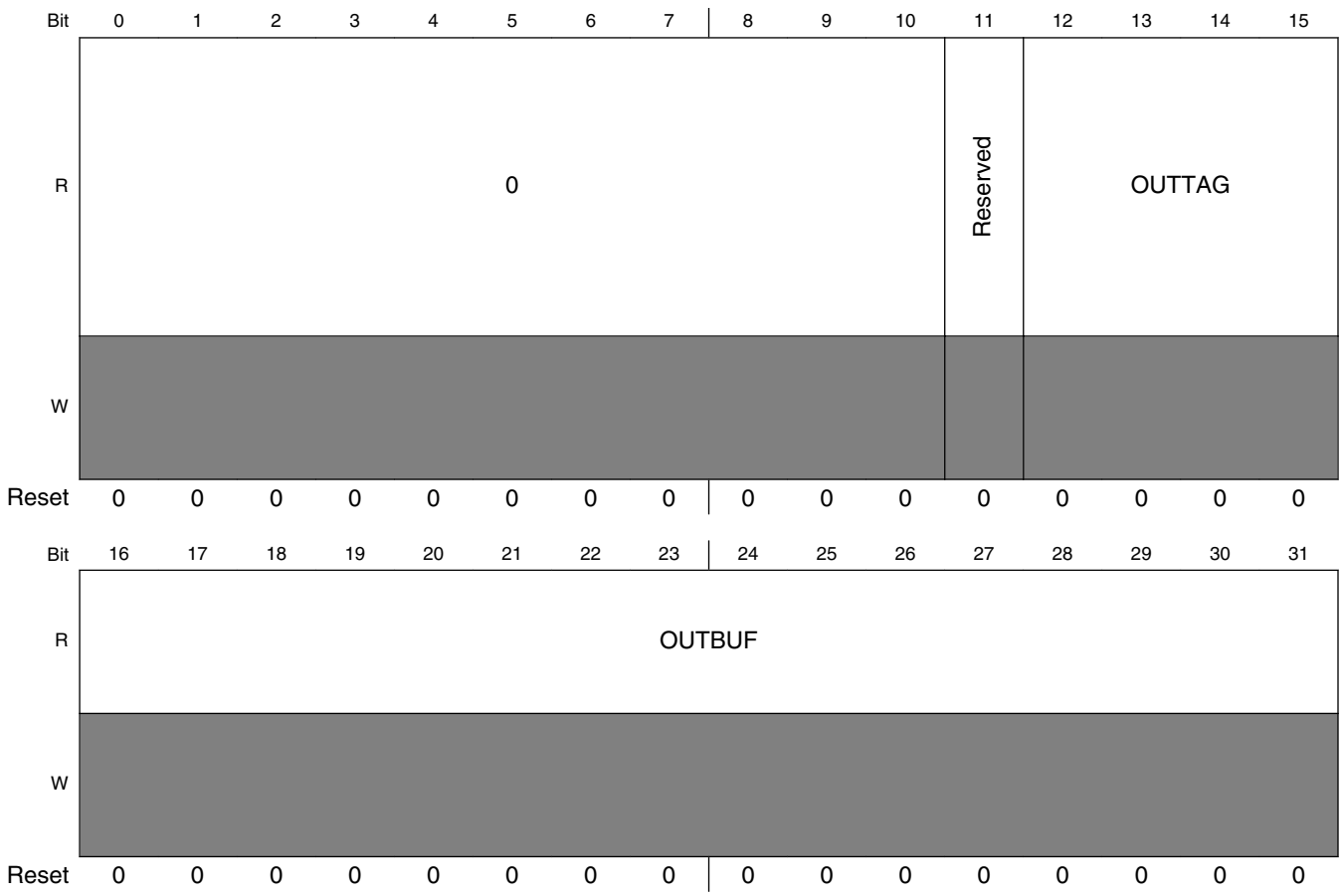
Table continues on the next page...

DECFILTER_IB field descriptions (continued)

Field	Description
	0 no flush request 1 flush request
16–31 INPBUF	Input Buffer Data The INPBUF[15:0] bit field carries the sample data to be filtered. See Input Buffer Description for more details.

36.4.6 Interface Output Buffer (DECFILTER_OB)

Address: 0h base + 14h offset = 14h



DECFILTER_OB field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 Reserved	This field is reserved.

Table continues on the next page...

DECFILTER_OB field descriptions (continued)

Field	Description
12–15 OUTTAG	Decimation filter output tag bits The OUTTAG[3:0] bit field is defined as a selector signal and it is used to identify different destinations for the OUTBUF[15:0] data.
16–31 OUTBUF	Output Buffer Data The OUTPBUF[15:0] bit field is the result data in the decimation filter Output Buffer. It represents a fixed point signed number in two's complement format and is updated only when a decimated result is ready to be transmitted, meaning it contains the last decimated result from the filter.

36.4.7 Filter Coefficient (DECFILTER_COEF*n*)

Address: 0h base + 20h offset + (4d × i), where i=0d to 8d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COEFnSIGNR								COEFn																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DECFILTER_COEF*n* field descriptions

Field	Description
0–7 COEFnSIGNR	COEFn sign repeated Bit COEFn[23] is repeated 8 times.
8–31 COEFn	Coefficient n field The COEFn[23:0] bit fields are the digital filter coefficients registers. The coefficients are fractional signed values in two's complement format, in the range (Coef is greater than or equal to -1 and less than 1). **NOTE:** Reads to this register are sign-extended, meaning the coefficient's sign bit is copied to all 8 most significant register bits. **NOTE:** Writing to these fields when BSY=1 is not allowed.

36.4.8 Filter TAP (DECFILTER_TAP*n*)

The TAP register stores, on each filter node, the input sample data and, for the IIR type, the filter intermediary results.

Address: 0h base + 78h offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TAPnSIGNR								TAPn																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DECFILTER_TAP*n* field descriptions

Field	Description
0–7 TAPnSIGNR	TAPn sign repeated Bit TAPn[23] is repeated 8 times.
8–31 TAPn	TAPn Register The read-only TAPn[23:0] bit fields shows the contents of the digital filter tap registers, as fractional signed values in two's complement format, in the range (Coef is greater than or equal to -1 and less than 1). **NOTE:** Reads to this register are sign-extended, meaning the coefficient's sign bit is copied to all 8 most significant register bits. The content of these registers is meaningless when BSY=1.

36.4.9 Final Integer Value Register (DECFILTER_FINTVAL)

Address: 0h base + E0h offset = E0h

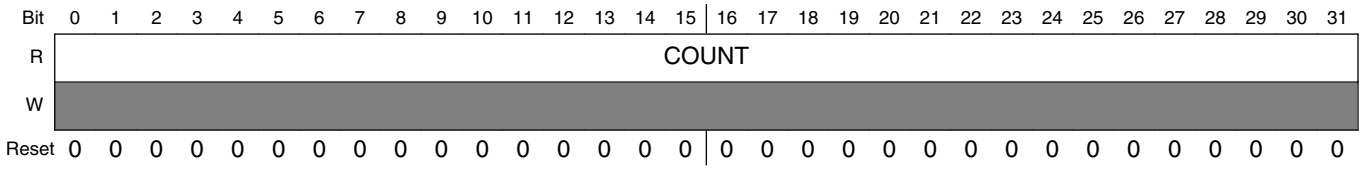
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SUM_VALUE																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DECFILTER_FINTVAL field descriptions

Field	Description
0–31 SUM_VALUE	Integration Sum Value The SUM_VALUE[31:0] field holds the sum of filtered output values. The 17 most significant bits hold the integer part, and the 15 least significant ones the fractional part of the integration value. The control of the integration sum and update of this register is determined by the register DECFILTER_MXCR. The register is updated only upon an integration output request. SUM_VALUE should be taken as an unsigned number when the integrator is configured for absolute operation (DECFILTER_MXCR bit SSIG=0), and a two's complement signed number otherwise. **NOTE:** If the DEFILTER_MXCR bit SSAT=1, the integration sum is saturated, so that if the accumulation overflows DECFILTER_FINTVAL holds the value 0xFFFFFFFF for absolute integration (SSIG=0), or values 0x7FFFFFFF (positive saturation) and 0x80000000 (negative saturation) for signaled integration (SSIG=1). If SSAT=0, DECFILTER_FINTVAL holds the integration sum modulo 2^{17} (considering the 15-bit fractional part).

36.4.10 Final Integer Count Register (DECFILTER_FINTCNT)

Address: 0h base + E4h offset = E4h

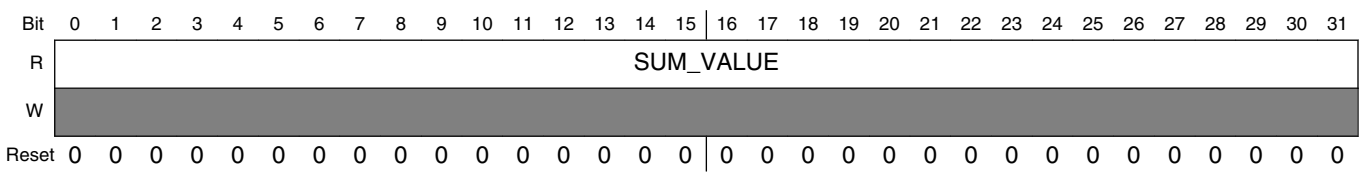


DECFILTER_FINTCNT field descriptions

Field	Description
0–31 COUNT	Integration Count Value The COUNT field holds the count of filtered outputs integrated. The control of the integration sum and update of this register is determined by the register DECFILTER_MXCR. The register is updated together with DECFILTER_FINTVAL, only upon an integration output request.

36.4.11 Current Integer Value Register (DECFILTER_CINTVAL)

Address: 0h base + E8h offset = E8h



DECFILTER_CINTVAL field descriptions

Field	Description
0–31 SUM_VALUE	Integration Sum Value The SUM_VALUE[31:0] field holds an unsigned number representing the sum of filtered output values, continuously updated as the integration proceeds. The control of the integration sum is determined by the register DECFILTER_MXCR. **NOTE:** If the DEFILTER_MXCR bit SSAT=1, the integration sum is saturated, so that if the accumulation overflows DECFILTER_CINTVAL holds the value 0xFFFFFFFF for absolute integration (SSIG=0), or values 0x7FFFFFFF (positive saturation) and 0x80000000 (negative saturation) for signaled integration (SSIG=1). If SSAT=0, DECFILTER_FINTVAL holds the integration sum modulo 2^{17} (considering the 15-bit fractional part). **NOTE:** A read on this register automatically commands an update of the register DECFILTER_CINTCNT.

36.4.12 Current Integer Count Register (DECFILTER_CINTCNT)

Address: 0h base + ECh offset = ECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DECFILTER_CINTCNT field descriptions

Field	Description
0–31 COUNT	Integration Count Value The COUNT field holds the count of filtered outputs integrated. The value is updated only when register DECFILTER_CINTVAL is read, to keep the coherency between the integration and count values.

36.5 Functional Description

36.5.1 Overview

Figure 36-1 shows the block diagram of the Decimation Filter. The Control Logic provides the control signals for all other sub-modules. The data interface is sub-divided into two sub-modules, transmitter and receiver, that are accessed by slave-bus interfaces. The bypass path is used when the filter is disabled and the incoming data can be transmitted back to the master block without being processed by the Filter algorithm. The Filter hardware is implemented in such a way that an IIR (1 x 4 poles) or FIR filter type can be implemented. The selection between the two types of filter algorithms is implemented by the Control Logic sub-block.

The Coefficient register file provides the digital filter coefficients. This block is a register bank with read/write access by the deviceslave-bus interface. The Filter TAP registers are also accessed through the device slave-bus line interface, providing additional debug capabilities to the Decimation Filter block. The MAC (Multiply Accumulate) sub-block executes the filter arithmetic operations controlled by the Control Logic. The MAC results are routed to the Filter TAP registers and to the output buffer when the result is a decimated filter sample.

36.5.2 Data Interface sub-block description

This section describes the operation of the data interface sub-block which is responsible for communication and data exchange between the slave-bus and the Decimation Filter block. It is important to note that there are only one internal input buffer and only one internal output buffer.

The decimation filter receives sample data from the memory . The input data bus format is presented in Interface Input Buffer (DECFILTER_IB). The sample data arrives along with the control bits. These control bits are decoded and the proper action is decided in the Control Logic sub-block. When the decimation filter finishes its processing and a result is available, the read request signal is issued to the selected master block. This data transfer request remains set until the result is read by the master block.

36.5.3 Input Buffer Description

The decimation filter receives data samples for filtering from the CPU using the device slave bus interface. The data source is selected by the ISEL bit (ISEL=1) of the module configuration register DECFILTER_MCR.

When the device slave bus interface is selected and DMA operation is chosen (DECFILTER_MCR bit DSEL=1), the input data request signal is asserted when the input buffer is empty. When DMA operation is not chosen (DSEL=0) in standalone, the logic asserts an input interrupt request and the input buffer waits for data from the device slave bus interface.

Input buffer filling is flagged by the DECFILTER_MSR bit IDF. The IDF flag remains set, even after the input data has been consumed and the buffer is free, until it is cleared by software.

Input buffer overrun is detected and flagged by the DECFILTER_MSR bit IVR. The idle (BSY=0) Decimation Filter is able to receive two consecutive input writes without input overrun. For more details, see Input Buffer Overrun.

Soft reset clears the input buffer indication flags and any data write/read request that was generated in any mode.

36.5.3.1 Input Buffer Overrun

An input overrun occurs when the input buffer is holding input data and new data is received by the filter. See Input Buffer Description for details of the input buffer.

The input buffer overrun can occur only when the input is enabled (DECFILTER_MCR bit IDIS=0).

36.5.4 Output Buffer Description

The decimation filter has an output buffer to send filtering results to the CPU using the device slave-bus, as selected through the DECFILTER_MCR bits ISEL.

Filtering of prefill inputs do not update the output buffer, so the flag ODF is not set.

When in filter operation mode with input from the device slave-bus (ISEL=1), only sample data is processed by the filter. When the filter result from the MAC is ready, this is immediately written to the output buffer when the decimation count is reached. The flag ODF is set when the buffer is updated. It also generates a DMA read request if DSEL=1 in standalone mode.

When the filter is bypassed (FTYPE=00), and the device slave-bus is selected as output (standalone mode), the data written into the input buffer is immediately written into the output buffer. The flag ODF is set when the buffer is updated and a DMA read request is asserted if DSEL=1.

Soft reset clears the output buffer, as well as any data read request generated in any mode.

36.5.4.1 Output Buffer Overrun

An output overrun occurs when the output buffer is holding output data that has not been read is overwritten with another word of data.

Output overruns are flagged by the DECFILTER_MSR bit OVR, if the output buffer is updated when not empty. The output buffer empty condition depends on the mode and output selection as follows:

- If cascade mode is selected, the output buffer is considered empty if the last output request was acknowledged.
- If the device slave-bus is selected (standalone mode) for output and DMA is selected (DSEL=1), the output buffer is considered empty if the last output has been read.
- If the device slave-bus is selected (standalone mode) for output and DMA is not selected (DSEL=0), the output buffer is considered empty when the ODF flag is negated.

Note

When the device slave-bus is selected (standalone mode) for output and DMA is not selected (DSEL=0), the ODF flag must be cleared to avoid overrun, even if its corresponding interrupt is not used (ODEN=0).

Prefill inputs do not cause IIR or FIR output overrun in standalone, and cascade tail mode, but can cause overruns in cascade head or middle configurations (see Cascade mode).

When bypass is selected, the output overrun does not occur because the data written into the input buffer is written into the output buffer only when this buffer is empty, but an input overrun may still occur (see Input Buffer Overrun).

36.5.5 Bypass Configuration Description

Bypass operation is configured by setting the field FTYPE[1:0] of module configuration register DECFILTER_MCR to 00. In this case, the input sample and tag are sent to the output with no change. This behavior is independent of the ISEL setting. The following applies to the bypass configuration:

- flush is ignored
- prefill is ignored
- counted decimation is ignored
- BSY bit is not set
- the input and output flags are set

Note

Bypass must not be configured in cascade mode (see Cascade mode description).

36.5.6 IIR and FIR Filter

This section describes the IIR filters implemented in the Decimation Filter block. Figure 36-31 shows the filters functional diagrams. The figure shows a IIR filter of fourth order (4poles). The filter topology, not the hardware, is represented. The hardware implementation is based on a MAC unit controlled by an FSM which implements the filter algorithm.

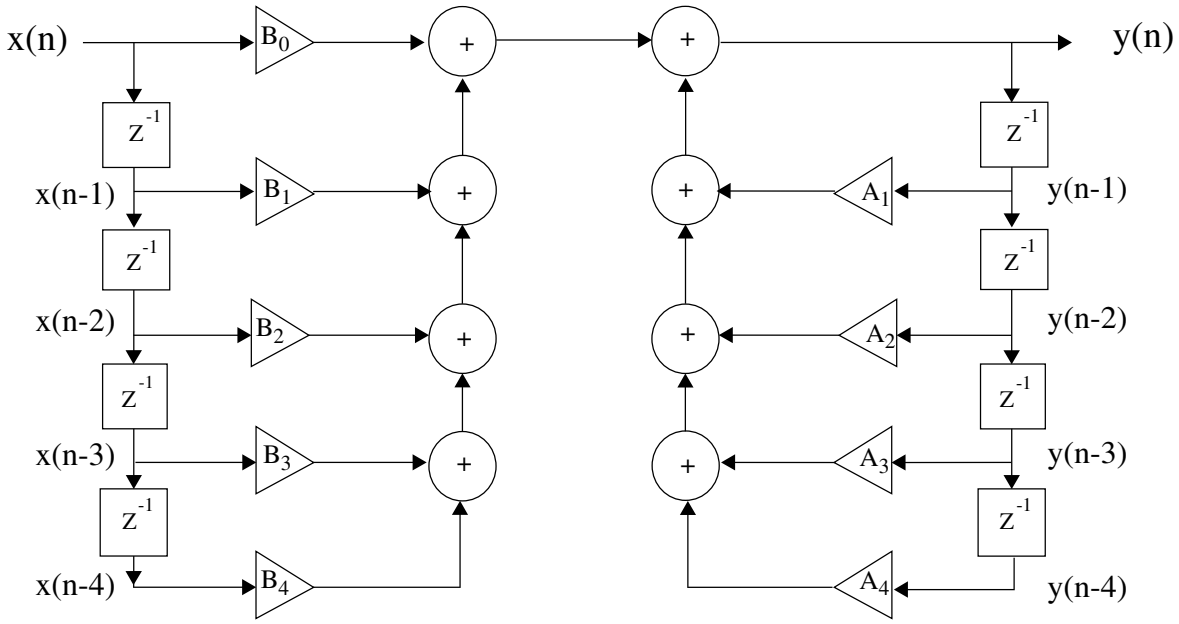


Figure 36-31. 1 x 4 Poles IIR Filter Functional Diagram

The difference equation for the IIR filter of Figure 36-31 can be written as:

$$y(n) = \sum_{i=0}^{N} B_i x(n-i) + \sum_{j=1}^{M} A_j y(n-j)$$

Figure 36-32. Eqn-1

where $x(n)$ is the filter input at time n, $y(n)$ is the filter output at time n, N is the number of feed-forward filter coefficients minus one, B_i are the feed-forward filter coefficients, M is the number of feed-back filter coefficients, and A_j are the feedback filter coefficients.

Figure 36-32 can be written as:

$$y(n) = S\left\{\sum_{i=0}^{N} \frac{B_i}{S} x(n-i) + \sum_{j=1}^{M} \frac{A_j}{S} y(n-j)\right\}$$

Figure 36-33. Eqn-2

Where all the coefficients are scaled down by S. The block diagram for Figure 36-33 is shown in Figure 36-34 in a fourth-order IIR filter implementation where the coefficients A_j and B_i are called *coefficient n*, where n=0-8.

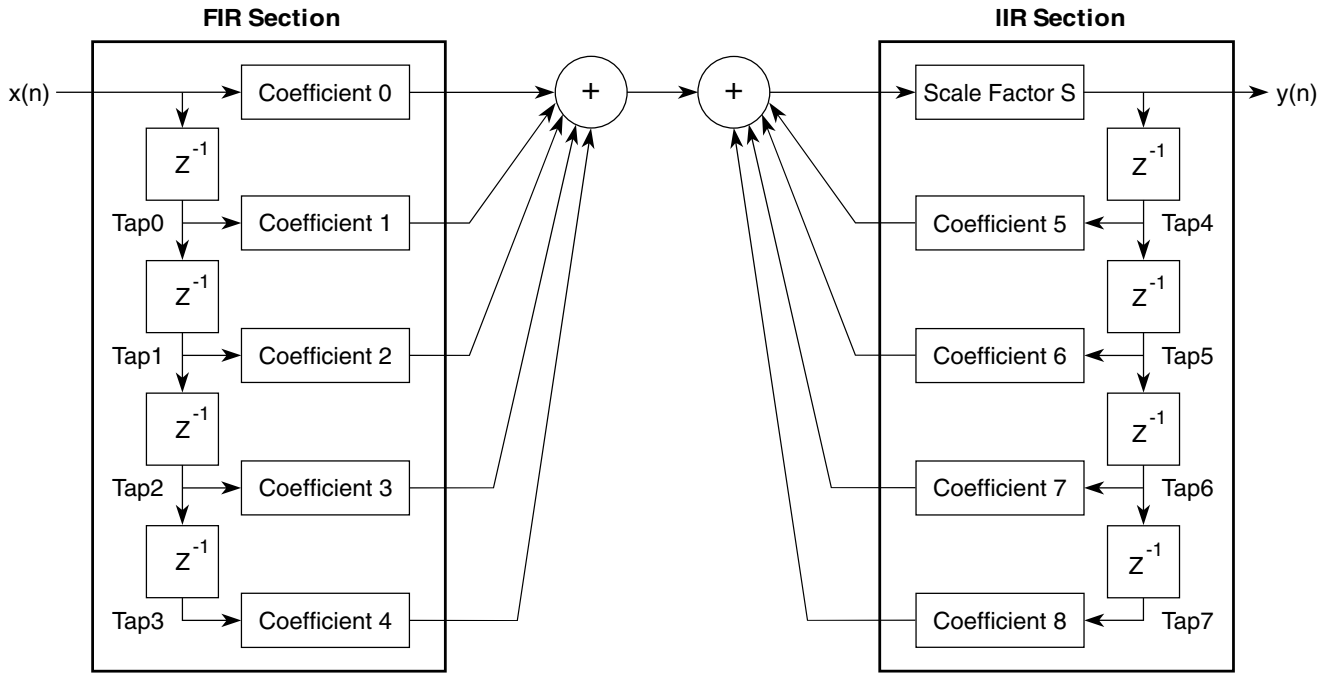


Figure 36-34. Fourth Order IIR Filter Implementation Block Diagram

The fourth order IIR filter is implemented with a FIR section followed by an IIR section. If the FIR type filter mode is selected, the IIR section is converted into an FIR section. In this case the order of the FIR filter is twice the IIR filter order, since all the TAP and coefficient registers are allocated for the FIR section. The Filter configuration paths are shown in Figure 36-35. Multiplexer A controls the *bypass* filter path and multiplexer B controls/selects the filter mode of operation, to either IIR mode or FIR mode. The selection is controlled by the FTYPE[1:0] bits in the Filter Module Configuration register. The order of the filter can be controlled by setting the appropriate *filter coefficients* to *zero*.

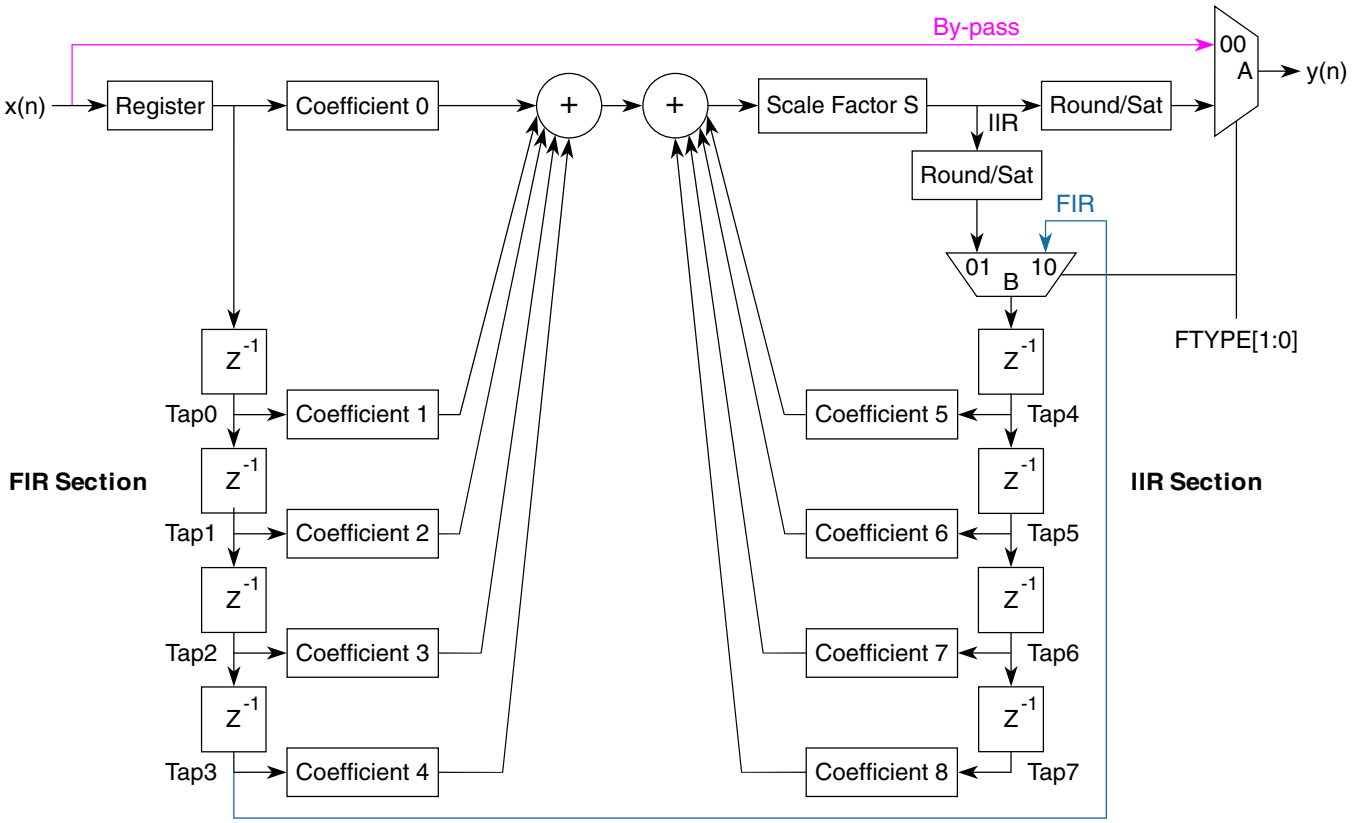


Figure 36-35. Filter Configuration Paths (FIR or 1x4Poles IIR)

36.5.6.1 Rounding

The Decimation Filter performs rounding operations in two different locations, as shown in Figure 36-35 :

- to obtain the filter output result with 16 bits
- to obtain the IIR feedback result to be stored in tap4 registers with 24 bits

The rounding mechanism implements the Convergent Rounding methodology (also known as round-to-nearest even number), which makes the decision on rounding up or down based on the value of the lower portion of data to be rounded (LS_WORD). The rounding up/down condition is equal to the traditional rounding except when the LS_WORD has the format {1000…00}. In this particular case, the rounding procedure is like the example of Figure 36-36. If the MS_WORD is odd, the value is rounded up. Otherwise the value is rounded down.

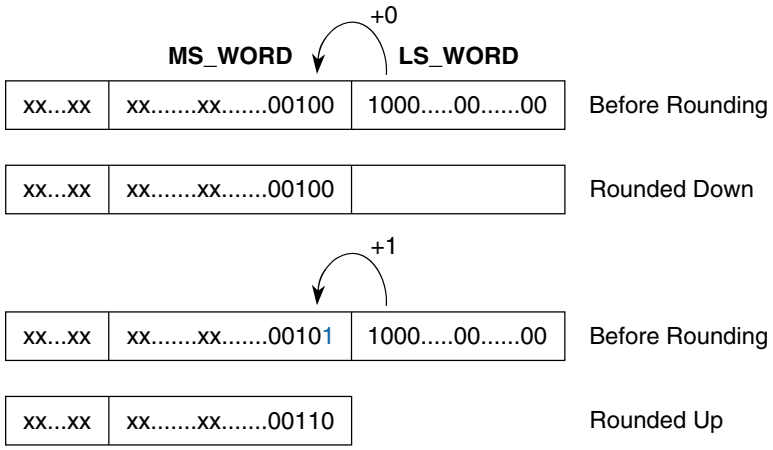


Figure 36-36. Convergent Rounding Methodology

36.5.6.2 Saturation

Filter output saturation occurs when an overflow or underflow condition of the filter is detected by dedicated logic, and if it is enabled by the SAT control bit of the configuration register DECFILTER_MCR. In this condition, the filter output is set to a saturated value equal to the maximum or minimum value that can be represented by the 16-bit output port. Also, for the IIR filter an equivalent logic is used to assert the saturation for the 24-bit feedback result.

36.5.7 Filter Prefill Control Description

A prefill indicates that the input data should be accepted by the Decimation Filter, but no decimated output should be generated while the control field indicates prefill. Therefore the prefill function is used in the beginning of the filter operation to initialize and stabilize the Decimation Filter without generating decimated samples. In addition, the prefill does not operate when the filter is in bypass (FTYPE=0b00).

When ISEL=1, it is controlled by the PREFILL field in the DECFILTER_IB register. The prefill control is usually activated only in a certain number of words of sample data in the beginning of the input data sequence.

When the prefill control is set, the decimation filter block operates as follows:

- Input data is processed normally by the digital filter and tap values are updated.
- The decimation counter is maintained in reset value.

- The output buffer is not updated and no output interrupt or read request is generated.
- The overflow detector/flag operates normally and the error interrupt request is set if enabled.

In Cascade Mode, prefill control is ignored in all cascaded filters except the Tail one: the prefill samples are filtered, decimated and forwarded to the next block with the prefill indication. The Tail block takes the samples with prefill indication, filter them and discard the result.

Note

The combined decimation count of the cascade combo may not be reset after the last prefill sample. Unlike in a single (non-cascaded) filter, the number of non-prefill inputs until an output comes out depends on the number of past prefill inputs.

36.5.8 Flush Command Description

The flush signal is used by the Decimation Filter to execute a partial reset of the filter. This is useful when the same filter is used on a new set of data samples after finishing the filtering of another set of data.

When the flush control is detected, all filter TAPs are cleared and the DEC_COUNTER[3:0] field in the status register DECFILTER_MSR is reset.

The flush function does not clear the Coefficient registers file in the Decimation Filter, thus it is not required to re-write these registers after a flush. The integrator accumulator and sample count are not cleared either. The output buffer also keeps the last result and may be retrieved until the next output is posted.

The flush control precedes the input data to be filtered. Therefore, the corresponding sample data is processed by the block after the flush. When ISEL=1, the field FLUSH in the DECFILTER_IB register is processed. Note that a word of valid sample data can be available at the same time the flush signal is asserted. In this case the flush is executed and the sample is processed after the flush.

When the filter is disabled by the FTYPE[1:0] control bit field, the flush command is not executed.

Note

In Cascade Mode, the flush command is forwarded to the next cascaded block together with the output, after the decimation count. Therefore it is possible that, in a given moment, the taps of a cascaded block are zeroed after a flush input, while the following ones still retain the old values.

36.5.9 Soft-Reset Command Description

The Soft-Reset command is requested through the self-negated bit SRES of the DECFILTER_MCR register and provides the CPU with the capability to initialize the Decimation Filter through the slave-bus interface. The procedure below must be performed for a software reset when the filter is active:

1. Disable filter inputs, writing DECFIL_MCR bit IDIS = 1.
2. Poll the register DECFIL_MSR until the bit BSY is 0.
3. Repeat the step 2 polling; this is necessary to cover the case when a sample is left in the input buffer.
4. Write DECFILTER_MCR bit SRES = 1.

After the software reset is issued, all internal Filter TAP registers, the decimation counter, the integrator outputs (except DECFILTER_CINTCNT), and the state machine are put in the initial state. The status register DECFILTER_MSR is also cleared. The Coefficient registers are not affected by the SRES. In case there is some filter processing, the filter process is aborted and the last sample is discarded. In addition, data in the input buffer waiting to be processed, and data in the output buffer waiting to be read, are discarded (the requests of service are cleared). The software reset command has high priority and the BSY bit is set during its operation.

The configuration registers DECFILTER_MXCR and DECFILTER_MCR are also not affected by a soft reset, except the bit SRES that is self-negated and is always read as zero.

When in debug or freeze mode, the soft reset is executed but the filter remains in debug or freeze mode.

Note

It is recommended to clear the IBIE bit before a software reset, especially if ISEL changes, in order to avoid unwanted interrupt requests.

Note

DMA transfers must not be active during soft reset. Data loss can occur.

36.5.10 Interrupts Requests Description

36.5.10.1 Block Interrupt Request

There are several interrupt request events that can be enabled using the module configuration register DECFILTER_MCR. Basically, the interrupt request can be issued under any of the following conditions:

- when a word of input data is received
- when a word of output data is available
- when an error has occurred.

The input data flag IDF is set when a word of data is received from the CPU when in standalone mode. It is not used to generate the read or write requests (as defined in Input Buffer Interrupt Request) when DSEL=0.

Output data is available and its flag (ODF) is set when the input data sample is processed by the filter and the decimation counter matches the decimation rate value. It is not used to generate the read requests (as defined in Output Buffer Interrupt Request) when DSEL=0.

An error event in the decimation filter block is defined as one of these events:

- Overflow in the filter, flagged by OVF
- Overrun in the decimation filter input, flagged by IVR
- Overrun in the decimation filter output, flagged by OVR
- Integrator overrun, flagged by SVR
- Integrator value exception, flagged by SSE
- Integrator count exception, flagged by SCE

An overflow occurs when the two's-complement result value from the MAC accumulator is out of the range of values that can be stored in tap register 4 (IIR) or in the output register.

An input overrun occurs when the input buffer is holding a word of input data and one more word of data is received by the filter. See Input Buffer Overrun, for more details.

An output overrun occurs when a new word of data is sent to the output buffer but the previous word of data has not been handled yet. See Output Buffer Overrun, for more details.

These flags are only cleared by:

- the CPU, or
- by the soft reset command in the DEC_FILTER_MCR, or
- by the clear flag registers in the DEC_FILTER_MSR register.

36.5.10.2 Input Buffer Interrupt Request

This interrupt is enabled by the register IBIE in the DECFILTER_MCR register and is asserted only when DSEL=0. This request is flagged in the DECFILTER_MSR by field IBIF.

In standalone mode, the input buffer interrupt request is asserted when the input buffer is available to receive a conversion sample and DMA operation is not selected (DSEL=0), meaning the block is requesting data to be written into the input buffer. The interrupt request is cleared when the CPU writes a "1" in field IBIC of the DECFILTER_MSR register, or by the soft reset command.

36.5.10.3 Output Buffer Interrupt Request

This interrupt is enabled by the register OBIE in the DECFILTER_MCR register and is asserted only when DSEL=0. This request is also indicated in the field OBIF of the DECFILTER_MSR register.

When in standalone mode, the output buffer can be read by the CPU with the DMA disabled. The output buffer interrupt request is asserted when the output buffer receives a new result from the filter sub-block. This means the block is requesting data to be read by the CPU.

The output buffer interrupt request can also be asserted due to an integrator result being ready, as flagged by the DECFILTER_MXSR bit SDF, when the DECFILTER_MCR bit SDIE=1. Note that both the filter output and integrator share the same interrupt source.

This interrupt request is cleared when a "1" is written in the bit OBIC and/or bit SDFC of the DECFILTER_MXSR register, or by the soft reset command.

36.5.11 DMA Requests Description

The DMA function for input and output buffers is enabled using DSEL=1 in the DECFILTER_MCR register. The DMA function for integrator result is described in another section.

36.5.11.1 Input Buffer DMA Request

This DMA request is enabled by the ISEL and DSEL bits in the DECFILTER_MCR register.

When in standalone mode, the input buffer can be written by the DMA. The input buffer DMA request is asserted when the input buffer is available to receive a conversion sample (it is not holding a word of data). This DMA request is cleared when an input data word is written in the input buffer. Therefore, the DMA request is always cleared before it is asserted again.

This DMA request is cleared when the DMA transfer is complete or by soft reset.

36.5.11.2 Output Buffer DMA Request

This DMA request is enabled by the ISEL and DSEL bits in the DECFILTER_MCR register.

When in standalone mode, the output buffer can be read using the DMA. The output buffer DMA request is asserted when the output buffer receives a new result from the internal filter sub-block. This DMA request is cleared when the output buffer is read by the processor.

The output buffer DMA request is also asserted when an integrator output is ready, if DECFILTER_MXCR bit SDMAE=1 with DSEL=0 .

The DMA request is also cleared by soft reset.

36.5.12 Freeze Mode Description

The freeze mode operation is asserted using the FREN enable bit and FRZ bit in the DEC_FILTER_MCR register or by the modules debug input .

It is not possible to enter freeze mode when the module is disabled by the configuration bit MDIS.

In case of a freeze mode request during the processing of an input sample, the current processing is finished and then the module enters freeze mode.

Access to input and output buffers remain operational in freeze, as well as their related flags.

Note

For cascaded blocks (see Cascade mode description), it is recommended to freeze only the Head block. If it is really necessary to freeze all the blocks, it is recommended that they are frozen sequentially starting from the Head block towards the Tail one. Unfreezing is recommended to be done sequentially from the Tail block towards the Head one.

36.5.13 Integrator

The hardware sample integrator accumulates the filter output values for determined periods.

36.5.13.1 Integrator Inputs

The integrator can be fed either by raw or decimated filter outputs, selected by the DECFILTER_MXCR bit SISEL (see Module Extended Config Register (DECFILTER_MXCR)). The accumulated input value taken can be the filtered sample "as is" (signaled), or its absolute value, depending on the DECFILTER_MXCR register bit SSIG.

Note

The integrator accumulates input samples when bypass is selected.

36.5.13.2 Integrator Outputs

The integrator output is either:

- the 32-bit, fixed point unsigned accumulation of the absolute values from the filter output, when the integrator is configured for absolute operation (DECFILTER_MXCR bit SSIG=0). This resolution allows a minimum of 131071 samples to be integrated before an overflow occurs in absolute operation, or 65536 samples in signed operation.
- the 32-bit, fixed point signed two's complement accumulation of the signed values from the filter output, when the integrator is configured for signed operation (SSIG=1).

The fractional part of the accumulation is 15 bits wide in both cases.

An accumulation overflow is flagged by the DECFILTER_MXSR bit SSOVF. The accumulator can overflow in either of the ways below, selected through the DECFILTER_MXCR bit SSAT:

- saturated accumulation (SSAT=1), so that an overflow results in the value of 0xFFFFFFFF for absolute value accumulation (SSIG=0), or 0x7FFFFFFF (positive) and 0x80000000 (negative) for signaled accumulation (SSIG=1).
- non-saturated accumulation (SSAT=0), so that an overflow results in the modulo 2^{17} accumulation value. This operation is only allowed in absolute accumulation (SSIG=0).

The integrator output value becomes available in register DECFILTER_FINTVAL (see Final Integer Value Register (DECFILTER_FINTVAL)) when an integrator output request is issued. The integrator output request can be issued in the following ways:

- by hardware, controlled by an external signal; the enabling and the selection of the signal request modes is done through the DECFILTER_MXCR field SRQSEL (see Module Extended Config Register (DECFILTER_MXCR));
- by software, writing 1 to the DECFILTER_MXCR bit SRQ;

The SSOVF flag is asserted upon an integrator output request, based on the overflow state of the internal accumulator. This internal overflow state is cleared upon an output request, just after SSOVF is asserted, or upon an integrator reset (see Integrator Reset).

The internal overflow state is also cleared by writing SSOVFC to 1, but only in saturated accumulation. Therefore, a non-saturated overflow that occurs before an SSOVF clear is still flagged in the next output request.

The integrator output request also updates the register DECFILTER_FINTCNT, which holds the number of samples accumulated into the register DECFILTER_FINTVAL. This internal accumulated sample counter can operate either in a saturated or "wrapped" count mode, as selected by the DECFILTER_MXCR bit SCSAT. In both cases, the counter overflow is flagged by the DECFILTER_MXSR bit SCOVF.

The SCOVF flag is asserted upon an integrator output request, based on the overflow state of the internal counter. This internal overflow state is negated upon an output request, just after SCOVF is asserted, or upon an integrator reset (see Integrator Reset). The internal overflow state is also negated by writing SCOVFC to 1, but only in saturated count. Therefore, a non-saturated overflow that occurs before an SCOVF clear is still flagged in the next output request.

An integrator output update can also issue a DMA or interrupt request. The interrupt and DMA requests are the same ones used for the filter output buffer (see Output Buffer Interrupt Request and Output Buffer DMA Request). The DECFILTER_MCR bit SDIE is used to enable integrator interrupts, and the DECFILTER_MXCR bit SDMAE enables the DMA integrator requests. The integrator DMA request uses the same signal as the filter output DMA request, so one must never use any configuration that allows both the integrator and filter output to make DMA requests.

Integrator output updates are flagged by the DECFILTER_MXSR bit SDF. The integrator overrun is detected in the same way as a filter output buffer overrun, and is flagged by DECFILTER_MXSR bit SVR. An integrator overrun also generates an error interrupt if the DECFILTER_MCR bit ERREN=1 (see Output Buffer Overrun).

Registers DECFILTER_CINTVAL and DECFILTER_CINTCNT provide a way to poll intermediate integration values and sample counts, respectively (see Current Integer Value Register (DECFILTER_CINTVAL) and Current Integer Count Register (DECFILTER_CINTCNT)). DECFILTER_CINTVAL is updated whenever the integrator is reset or a new sample is accumulated. DECFILTER_CINTCNT is updated only when DECFILTER_CINTVAL is read, so that coherency between the value and count values is guaranteed. Therefore, the read access order of that pair of registers must be DECFILTER_CINTVAL first, followed by DECFILTER_CINTCNT.

Note

The flags SSOVF and SCOVF can also be asserted when DECFILTER_CINTVAL is read. The SSOVF and SCOVF set and clearing rules apply for the DECFILTER_CINTVAL read the same way as for an integrator output request.

36.5.13.3 Integrator Reset

The integration value is reset to the value of zero, in the following ways:

- by hardware: on hardware reset, or controlled by an external signal; the enabling and the selection of the zero signal modes is done through the DECFILTER_MXCR field SZROSEL (see Module Extended Config Register (DECFILTER_MXCR));
- by software: on software reset, or writing 1 to the DECFILTER_MXCR bit SZRO;

The integrator reset also zeroes the internal counter of accumulated samples and the internal overflow state (but not SSOVF and SCOVF). Software and hardware resets reset all integrator registers immediately. Integrator zero command from external signal or by software (SZRO) affects the integrator registers and flags as follows:

- DECFILTER_CINTVAL resets immediately;
- DECFILTER_CINTCNT does not reset immediately; it is updated only upon a DECFILTER_CINTVAL read, loaded with the number of integrated samples occurred after the reset;
- DECFILTER_FINTVAL and DECFILTER_FINTCNT do not reset immediately; being updated only upon a new output request (see Integrator Outputs); if a integrator software zero command (through SZRO bit) and an integrator output request (through SRQ bit) are made at the same time, the registers DECFILTER_FINTVAL and DECFILTER_FINTCNT are updated with the last internal values before reset; the same applies to simultaneous integrator zero command and output request by hardware signal;
- the SSOVF and SCOVF flags do not negate; however, the internal overflow states which assert SSOVF and SCOVF do reset immediately, so that the next output update (either by hardware request, software request or DECFILTER_CINTVAL read) before an overflow does not assert SSOVF/SCOVF.

Note

The integrator reset does not depend on the integrator enabling (see Integrator Enabling and Halting).

36.5.13.4 Integrator Enabling and Halting

Two mechanisms, enabling and halting, drive the integrator accumulation, allowing it to be controlled by a combination of two distinct sources, both software, both hardware, or one hardware and other software. Values are accumulated when the integrator is enabled and not halted. The integrator halt and enable states can be controlled in the following ways:

- by hardware, through external signals ; the enabling and the selection of the signal request modes is done through the DECFILTER_MXCR fields SENSEL and SHLTSEL, respectively (see Module Extended Config Register (DECFILTER_MXCR));
- by software, through the same DECFILTER_MXCR fields SENSEL and SHLTSEL. Note that these fields are in different bytes, so that two distinct, concurrent software tasks can avoid coherency problems by changing the fields using byte read-modify-write accesses.

Note

Enabling and halting does not affect output requests or integrator reset.

36.5.13.5 Integrator Exceptions

Integrator may run into exception states due to overflow, either of the accumulated value or the sample counter. Exceptions are flagged by the DECFILTER_MXSR bits SSE, for sum value exception, and SCE, for counter exception. These flags generate an error interrupt, if it is enabled (see Block Interrupt Request).

The accumulator exception condition depends on whether it operates in saturated mode or not, as follows:

- In Saturated operation (DECFILTER_MXCR bit SSAT=1): a sum exception occurs (SSE=1) whenever an overflow is flagged; SSE asserts together with SSOVF.

- In Non-saturated operation (DECFILTER_MXCR bit SSAT=0): a sum exception occurs (SSE=1) when an overflow is flagged and the DECFILTER_MXSR bit SSOVF is already set to 1.
- In Non-saturated operation, an accumulator exception also occurs if the accumulator overflows twice without any update of the final integrator value DECFILTER_FINTVAL or the current integrator counter DECFILTER_CINTCNT (by a read to the DECFILTER_CINTVAL register), neither an integrator reset occurs. The SSOVF flag does not assert in this situation.

Note

The SSOVF flag can only be asserted upon a hardware request, a software request, or when DECFILTER_CINTVAL is read, based on the internal accumulator overflow state.

Similarly, the sample counter exception condition depends on whether it operates in saturated mode or not, as follows:

- In Saturated operation (DECFILTER_MXCR bit SCSAT=1): a counter exception occurs (SCE=1) whenever an overflow is flagged; SCE asserts together with SCOVF.
- In Non-saturated operation (DECFILTER_MXCR bit SCSAT=0): a counter exception occurs (SCE=1) when an overflow is flagged and the DECFILTER_MXSR bit SCOVF is already set to 1.
- In Non-saturated operation, a counter exception also occurs if the counter overflows twice without any update of the final count DECFILTER_FINTCNT or the current integrator counter DECFILTER_CINTCNT (by a read to the DECFILTER_CINTVAL register), neither an integration reset occurs. The SCOVF flag does not assert in this situation.

Note

The SCOVF flag can only be asserted upon a hardware request, a software request, or when DECFILTER_CINTVAL is read (also updating DECFILTER_CINTCNT), based on the internal counter overflow state.

36.5.14 Cascade mode description

The cascade mode is defined as a configuration mode of the decimation filter to work together with other ones in a chain arrangement. All blocks in the arrangement, hereafter called a **cascade combo**, are configured to operate in cascade mode by the CASCD[1:0] field in the DECFILTER_MCR register. Figure 36-37 shows an example of cascade combo:

NOTE

As this device only has two DECFILTER instances, all description about the "Middle" block and references to more than two filters do not apply.

- The figure shows data input and data output in standalone mode.
- The left most block, named Head receives the raw data to be filtered from the device slave-bus interface in standalone mode. The head type is configured using CASCD[1:0] = 01 in the configuration register DECFILTER_MCR.
- The right most block, named Tail, is the last filter block in the chain. It sends the output result back to the device slave-bus interface in standalone mode. This type of cascaded block is configured using CASCD[1:0] = 10.
- The blocks in between, named Middle, do not exchange data (receive / transmit) with the device slave-bus, only with other decimation filter blocks. This type of block is configured by setting CASCD[1:0] = 11. Middle blocks are optional in a cascaded combo: two blocks, one Head block feeding a Tail one, can be used in cascade.
- Each decimation filter block has one cascade-in and one cascade-out ports. The arrows show how they are physically connected. The bold arrows show the connections used in cascade mode.
- The block configurations as Head, Tail or Middle must respect their physical connections such that all the following apply:
  - a Head block feeds a Middle or Tail one
  - a Middle block feeds another Middle block or a Tail one
  - Tail feeds no other block, and Head is fed by no other block.
- As a consequence of the conditions above, there must be one and only one Head block and one and only one Tail block in a cascade combo.

- A group of physically chained blocks can form more than one cascade combo. For instance, Figure 36-38 shows the same physical chain, but now with blocks 1 and 2 configured as Head and Tail, respectively, forming one combo. The remaining blocks form another combo starting with block 3 (Head), and ending with block N (Tail).
- Blocks not used in a cascaded chain can be used normally, isolated (DEFILTER_MCR field CASCD[1:0] = 00), as exemplified in Figure 36-39.
- The optional connection show from block N to block 1 in Figure 36-37 allows block N to be configured as Head or Middle, feeding block 1 configured as Middle or Tail, yielding more flexibility, as in the last example of Figure 36-39.

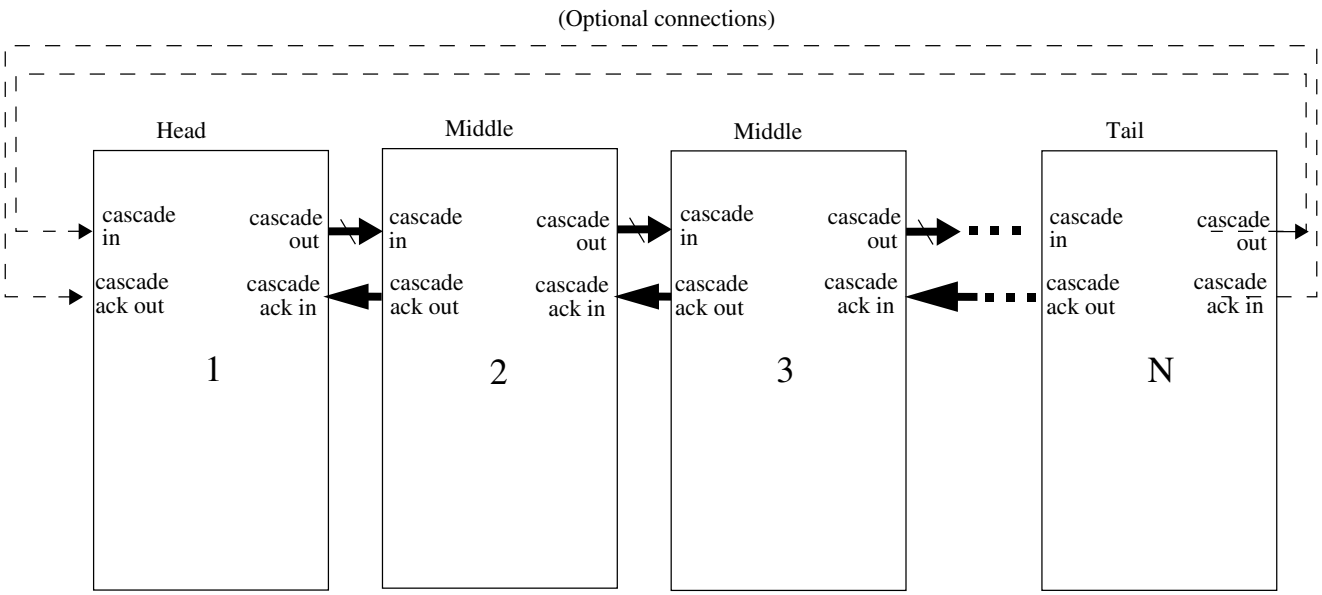


Figure 36-37. Cascade Mode Chain Structure

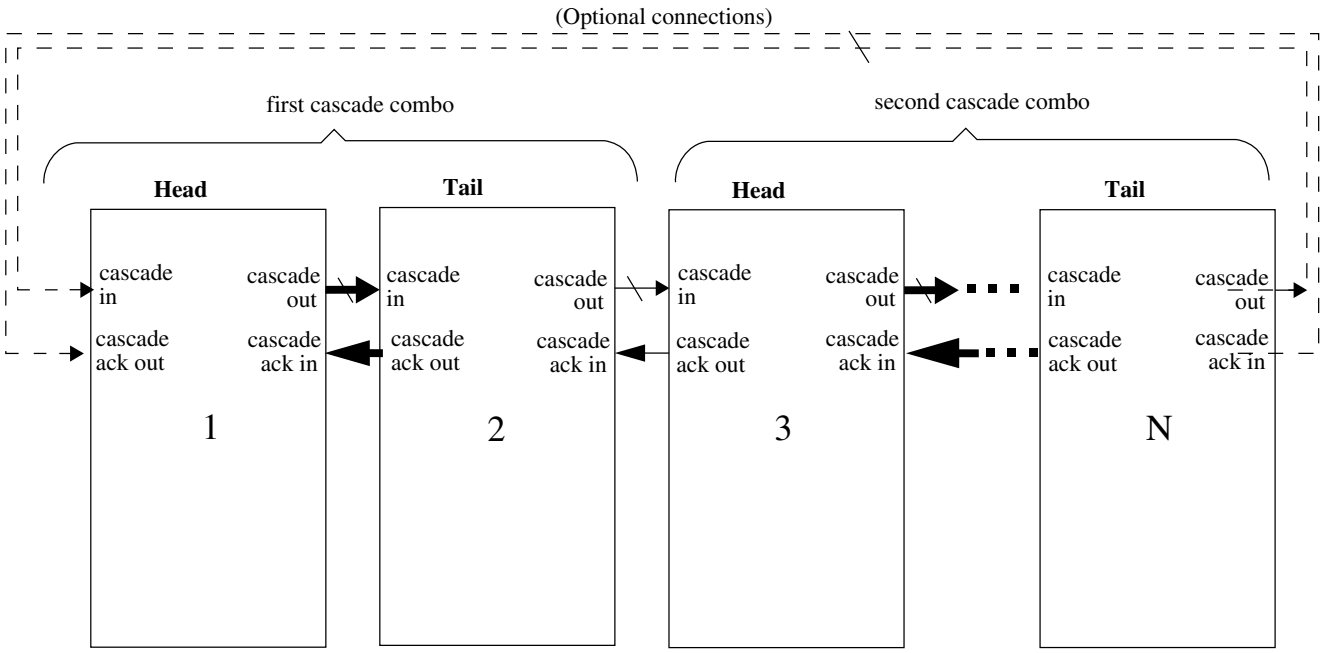
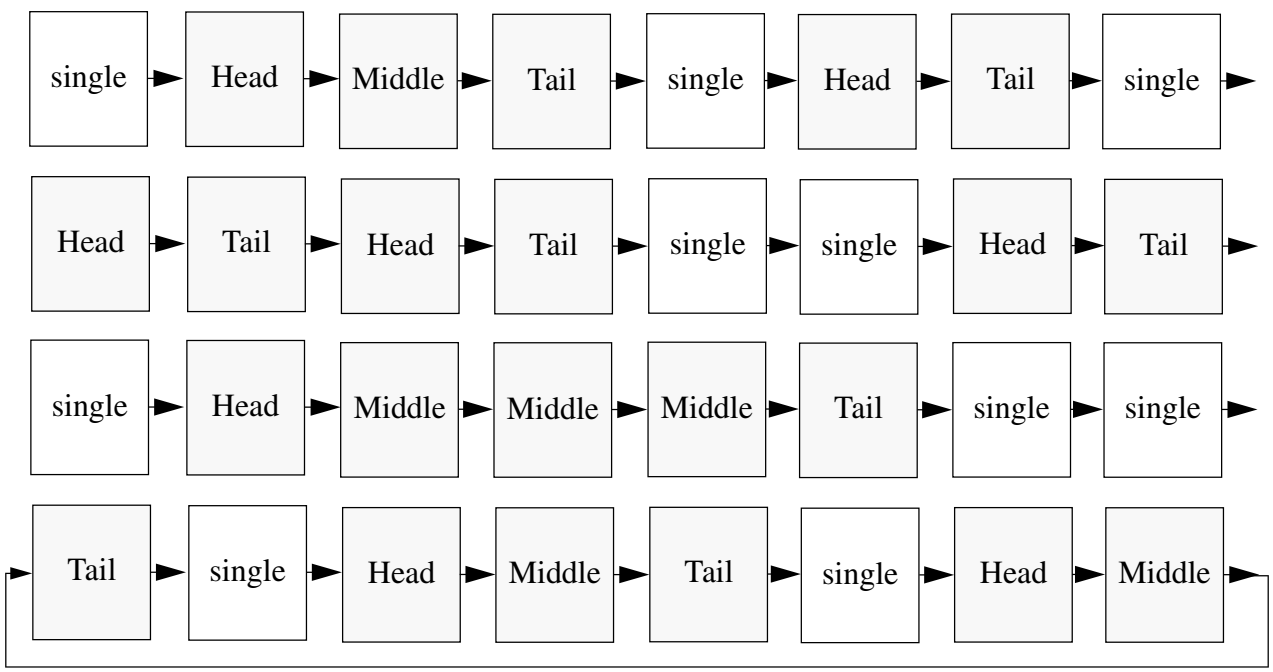


Figure 36-38. Multiple Cascade Mode Chain Structure



(arrows show just the physical cascading order)

Figure 36-39. Examples of mixed Cascaded and single blocks

36.5.14.1 Cascade Freeze, Stop, and Configuration Change Procedures

To change a block configuration mode to or from cascade mode, the following safe procedures must be observed:

- To modify a cascade combo, either to single or any other cascade combo combination, all the cascade combo blocks must have their inputs disabled (using DECFILTER_MCR bit IDIS), in order, from the Head to the Tail block. After a block IDIS bit has been set to 1 (one), one must wait for its DECFILTER_MSR bit BSY to be 0 (zero) before disabling the input of the next block in the sequence.
- Each block in a new cascade combo must be configured with its input disabled. When the mode configuration is done, the combo blocks must have their inputs enabled in order, from the Tail towards the Head block.
- A single block must also be reconfigured the same way, to or from a cascade combo configuration: first disabling its input, and then waiting for a non-busy state before writing DECFILTER_MCR field CASCD.

To take cascade combo blocks to or from freeze or low power modes, a similar procedure must be used:

- Take the Head to freeze or low-power first, wait for DECFILTER_MSR bit BSY=0, and repeat the procedure for the other blocks in the chain in sequence, towards the Tail block.
- Take the blocks out of freeze or low-power modes in the inverse sequence, from Tail to Head.

36.6 Initialization Information

Following are some simple initialization steps to be done before using the decimation filter block. These steps assume that the user has already calculated the filter coefficients using a filter design tool.

36.6.1 Initialization Procedure

The sequence of steps for the block initialization is as follows:

1. Program the configuration registers DECFILTER_MCR and DECFILTER_MXCR as desired for your application.
2. Write all filter coefficient registers DECFILTER_COEFn with the previously calculated values.
3. Enable the filter input, writing DECFILTER_MCR bit IDIS=0.
4. Run a soft-reset cycle if necessary.
5. The module is ready to receive data from the device slave-bus interface.

36.7 Filter Example Simulation

The decimation filter block operation was checked in a Verilog simulation using calculated filter coefficient values and noisy input data. The expected output values and the RMS error were then calculated.

36.7.1 Coefficients Calculation

The coefficients were calculated using a digital filter design tool. We have supplied some hypothetical filter parameters to the tool and obtained the filter coefficients. The input parameters are:

- Filter characteristics: elliptic/low pass
- Filter type: 4th order IIR
- Input sample rate: 800k sample/s
- Passband edge: 100 kHz
- Stopband edge: 150 kHz
- Passband attenuation: ≤ 1 dB

The software tool gives the IIR filter coefficients in the Z-transform format expressed by Figure 36-40 :

$$\frac{Y(s)}{X(s)} = \frac{B_0+B_1s+B_2s^2+B_3s^3+B_4s^4}{A_0+A_1s+A_2s^2+A_3s^3+A_4s^4}$$

Figure 36-40. Eqn-3

The coefficient results in fixed point decimal representation are shown in Table 36-32.

Table 36-32. Coefficient Values Given by SPW Digital Filter Design Tool

Coefficient	Decimal Value		Coefficient	Decimal Value
B0	0.0221455		A0	1.0
B1	0.00445582948893748		A1	-2.69772868375858
B2	0.0318517846509088		A2	3.234056294853
B3	0.00445582948893748		A3	-1.92028561712454
B4	0.0221455		A4	0.47939080709495

Comparing Figure 36-33 with Figure 36-40, we obtain the relationship between the calculated values of coefficients and the values to be loaded in the DECFILTER_COEFn registers. Note that A0=1 in both cases and that the coefficients A1 to A4 in Figure 36-40 has the opposite signal from Figure 36-33.

See Table 36-33 below to obtain the coefficients. A scale factor of eight is used, being the smallest divider factor to have all coefficient values in the range (-1 ≤ Coef < +1).

Table 36-33. Coefficient Values for Decimation Filter

SCALE	S=1	S=8	
COEFn	Decimal Value	Decimal Value	Hexadecimal Values (24 bits)
Coef0 = B0/S	0.0221455	0.00276815891266	0x005AB5
Coef1 = B1/S	0.00445582948893748	0.00055694580078	0x001240
Coef2 = B2/S	0.0318517846509088	0.00398147106171	0x008277
Coef3 = B3/S	0.00445582948893748	0.00055694580078	0x001240
Coef4 = B4/S	0.0221455	0.00276815891266	0x005AB5
Coef5 = -A1/S	2.69772868375858	0.33721613883972	0x2B29E6
Coef6 = -A2/S	-3.234056294853	-0.40425717830658	0xCC414E
Coef7 = -A3/S	1.92028561712454	0.24003565311432	0x1EB97D
Coef8 = -A4/S	-0.47939080709495	-0.05992400646210	0xF8546A

36.7.2 Input Data Calculation

The 24-bit words of input data are samples of a sum of two tones: one tone of 30kHz and another tone of 250kHz. The samples were calculated at the rate of 800k samples per second. The tones have the same amplitude and it is assured that the resultant amplitude is smaller than 1 so as to obtain samples in the range (-1 ≤ sample < +1). It is supposed the input data are signed values in the two's complement format in the range (-1 ≤ sample < +1).

36.7.3 Filter Results

The decimation filter was simulated using the calculated coefficients and the input data samples. A scaling factor of eight in the configuration register DECFILTER_MCR, and no decimation factor, were used to obtain the maximum of output results from the filter.

The theoretical expected values from this filter were also calculated, and these results were compared with those from the decimation filter. The resultant RMS error when considering about 500 samples was about -97 dB.

Chapter 37
Temperature Sensor (TSENS)

37.1 Introduction

The device includes two on-chip temperature sensors. The temperature sensor module monitors device temperature and delivers one analog output signal and two digital output signals each (under- and over-temperature flags). The analog output consists of a voltage signal directly proportional to the internal junction temperature. The analog output is connected to an input channel of an ADC on the device (see the ADC details for the channel number). The internal junction temperature must be calculated by software based on the sampled temperature sensor voltage; sampled bandgap reference voltage, which comes from another module; and calibration parameter values stored in internal flash memory. The two digital outputs, connected to the PMC module, are used to signal under- and over-temperature operating conditions.

The digital outputs signal when the junction temperature drifts below the low temperature threshold or above the high temperature threshold. These signals notify the device to take action to appropriately adjust the device temperature in response to an out-of-specification low or high temperature operating condition. Calibration parameter values associated with the temperature threshold detection feature are determined and stored in internal flash memory during production testing at the factory.

37.2 Functional Description

The temperature sensor generates one analog output voltage, $V_{TSENS}(T)$, which is proportional to the absolute current junction temperature of the device, and two digital outputs that signal whether the junction temperature has reached either a pre-set low temperature threshold or a pre-set high temperature threshold.

An on-chip ADC module is used to convert the analog output voltage, as well as the bandgap reference voltage, into a digital representation. These values, along with parameter values stored in on-chip flash memory, are used by software to calculate the device junction temperature.

37.2.1 Temperature threshold detection (digital output generation)

Temperature threshold trimming values are adjusted through calibration during factory test and stored in on-chip flash memory. While enabled, if a new trimming value is loaded into the Temperature Sensor module, it may take up to 1 microsecond for the digital outputs to be updated to reflect the new trimming condition.

When the temperature threshold detection feature is enabled, the temperature sensor monitors the internal junction temperature of the chip and asserts a signal if any of the following temperature thresholds are crossed.

- The low temperature digital output signals if the junction temperature falls below the low temperature threshold (-40°C).
- The high temperature digital output signals if the junction temperature rises above the high temperature threshold (150°C).

The threshold detection circuit is calibrated with respect to the factory-test temperatures: the low temperature threshold is trimmed to match the cold insertion test temperature (nominally equal to –40°C), and the high temperature threshold is trimmed to match the hot insertion test temperature (nominally equal to 150°C). In other words, the thresholds are not trimmed to absolute values of temperature but to the specific temperatures for which the device is validated during factory test.

Hysteresis is applied whenever outputs transition to a logic level low in order to eliminate spurious toggling. Digital output behavior is illustrated on Figure 37-1.

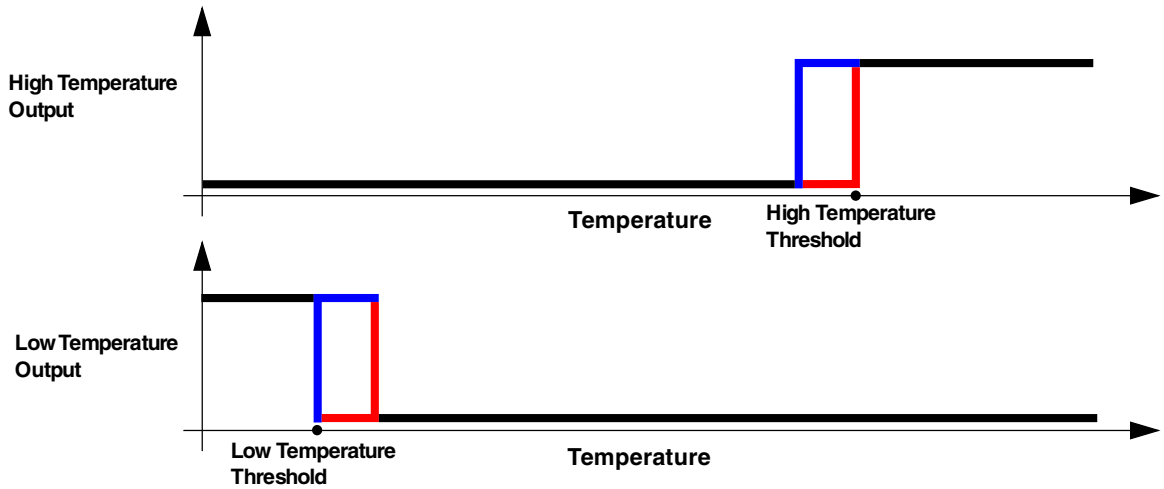


Figure 37-1. Digital output behavior with temperature

37.2.2 Linear temperature sensor (analog output generation)

When analog output generation is enabled, the temperature sensor outputs a voltage proportional to the internal junction temperature of the chip. This analog voltage signal is converted into a digital code by an on-chip ADC. The temperature value is obtained from a linear voltage-temperature relation with coefficients adjusted by calibration parameters extracted during factory test and programmed into flash memory.

37.3 Temperature formula

The system chain that translates device junction temperature into a digital variable is composed of the temperature sensor, a bandgap reference voltage source and the on-chip ADC. Both analog output voltages of the temperature sensor and the bandgap reference voltage source must be converted by the ADC into digital codes to obtain the device junction temperature. The temperature formula shown in Figure 37-2 is used to calculate internal device junction temperature.

$$T \times 2^2 = \frac{(K_1 \times V_{BG_CODE}(T)) \times 2^{-1} + (K_2 \times V_{TSENS_CODE}(T)) \times 2^3}{[(K_3 \times V_{BG_CODE}(T)) \times 2^2 + (K_4 \times V_{TSENS_CODE}(T))] \times 2^{-10}} \quad [K]$$

$$V_{TSENS_CODE}(T) = \frac{V_{TSENS}(T)}{V_{ref}} \times 2^{12} \qquad V_{BG_CODE}(T) = \frac{V_{BG}(T)}{V_{ref}} \times 2^{12}$$

- T is the internal junction temperature in Kelvin
- $V_{TSENS}(T)$ is the temperature sensor output sampled by the ADC
- $V_{BG}(T)$ is the bandgap reference voltage sampled by the ADC
- V_{ref} is the ADC reference voltage in normal usage of the device
- K1, K2, K3, and K4 are 16-bit signed integer constants in two's-complement representation stored in flash memory (obtained from factory calibration)

Figure 37-2. Temperature formula

Parameters K1, K2, K3, and K4 are 16-bit signed integer constants in two's-complement representation that are stored in flash memory during factory test and calibration. These constants must be used in the temperature formula to calculate device junction temperature.

37.4 Calculating device temperature

Software must perform the following steps in order to calculate device temperature.

1. Measure the bandgap reference voltage using the on-chip ADC module and calculate $V_{BG_CODE}(T)$ according to the formula shown in Figure 37-2.
2. Measure the temperature sensor output voltage using the on-chip ADC module and calculate $V_{TSENS_CODE}(T)$ according to the formula shown in Figure 37-2
3. Retrieve parameters K1, K2, K3, and K4 from flash memory.
4. Calculate T according to the formula shown in Figure 37-2.

Note

Temperature Formula may be calculated using signed 32-bit (integer) operations showing negligible precision loss.

Chapter 38
Timers Overview

38.1 Timers Overview

Three types of timers are implemented on the MPC5746R:

Table 38-1. Timers

Timer	Instances	Core	Feature Note
Sytem Timer Module (STM)	STM0	CPU0	
	STM1	CPU1	
Software Watchdog Timer (SWT)	SWT0	CPU0	Supports Fixed address mode and Incremental address modes.
	SWT1	CPU1	
	SWT3	n/a	Serviced by CPU0 and CPU1 while executing security routing. Does not support Fixed address mode and Incremental address modes. No IAC servicing
Periodic Interrupt Timer (PIT)	PIT0	Shared among all cores	8 channels with RTI
	PIT1		2 chained channels without RTI

There are two instances of the System Timer Module. Each of the two main cores has a dedicated STM. The checker core does not have an STM.

There are three instances of the Software Watchdog Timer. Each computational core has a dedicated SWT. The checker core does not have a dedicated SWT. SWT3 is different from the other two in that it does not have Instruction Address Compare servicing.

There are two instances of the Periodic Interrupt Timer, both which are shared among all cores. PIT1 does not have Real Time Interrupt.

Chapter 39
System Timer Module (STM)

39.1 System Timer Module (STM) configuration

39.1.1 STM chip-specific module information

MPC5746R implements two STMs: STM0 and STM1.

Table 39-1. System Timer Module (STM) instances

Instance	Description	Number of Channels	Number of Registers (32-bit)
STM0	Intended for Main Core_0 in the high-speed computational clock domain. However, by properly programming the INTC_PSR PRC_SEL field, interrupt requests generated by STM0 can still be serviced by the Core_1 and vice-versa.	4	14
STM1	Intended for Main Core_1 in the high-speed computational clock domain.	4	14

39.2 Introduction

This section provides an overview, list of features, and modes of operation for the STM.

39.2.1 Overview

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel. The counter is driven by the system clock divided by an 8-bit prescale value (1 to 256).

39.2.2 Features

The STM has the following features:

- One 32-bit up counter with 8-bit prescaler
- Four 32-bit compare channels
- Independent interrupt source for each channel
- Counter can be stopped in debug mode

39.2.3 Modes of operation

The STM supports two device modes of operation: normal and debug. When the STM is enabled in normal mode, its counter runs continuously. In debug mode, operation of the counter is controlled by the FRZ bit in the STM_CR register. If the FRZ bit is set, the counter is stopped in debug mode, otherwise it continues to run.

39.3 External signal description

The STM does not have any external interface signals.

39.4 Memory map and registers

The STM programming model includes a group of 32-bit registers—a module control register, a counter value register, and three registers for each channel.

The STM registers can only be accessed using 32-bit (word) accesses. Attempted references using a different size or to a reserved address generates a bus error termination.

STM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	STM Control Register (STM_CR)	32	R/W	0000_0000h	39.4.1/1195

Table continues on the next page...

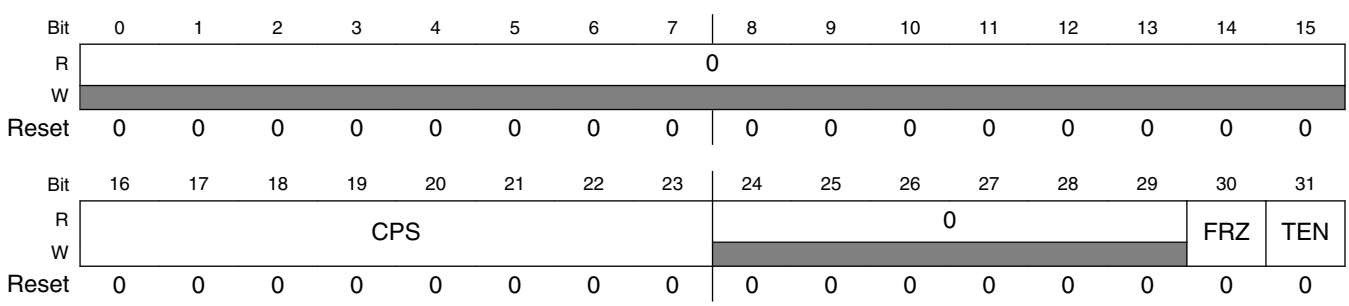
STM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4	STM Count Register (STM_CNT)	32	R/W	0000_0000h	39.4.2/1196
10	STM Channel Control Register (STM_CCR0)	32	R/W	0000_0000h	39.4.3/1197
14	STM Channel Interrupt Register (STM_CIR0)	32	w1c	0000_0000h	39.4.4/1197
18	STM Channel Compare Register (STM_CMP0)	32	R/W	0000_0000h	39.4.5/1198
20	STM Channel Control Register (STM_CCR1)	32	R/W	0000_0000h	39.4.3/1197
24	STM Channel Interrupt Register (STM_CIR1)	32	w1c	0000_0000h	39.4.4/1197
28	STM Channel Compare Register (STM_CMP1)	32	R/W	0000_0000h	39.4.5/1198
30	STM Channel Control Register (STM_CCR2)	32	R/W	0000_0000h	39.4.3/1197
34	STM Channel Interrupt Register (STM_CIR2)	32	w1c	0000_0000h	39.4.4/1197
38	STM Channel Compare Register (STM_CMP2)	32	R/W	0000_0000h	39.4.5/1198
40	STM Channel Control Register (STM_CCR3)	32	R/W	0000_0000h	39.4.3/1197
44	STM Channel Interrupt Register (STM_CIR3)	32	w1c	0000_0000h	39.4.4/1197
48	STM Channel Compare Register (STM_CMP3)	32	R/W	0000_0000h	39.4.5/1198

39.4.1 STM Control Register (STM_CR)

The STM Control Register includes the prescale value, freeze control and timer enable bits.

Address: 0h base + 0h offset = 0h



STM_CR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 CPS	Counter Prescaler. Selects the clock divide value for the prescaler (1 - 256). 0x00 Divide system clock by 1 0x01 Divide system clock by 2

Table continues on the next page...

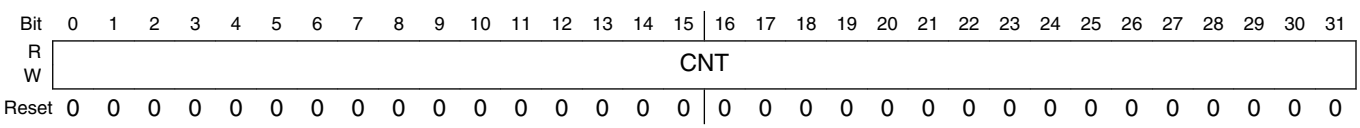
STM_CR field descriptions (continued)

Field	Description
	---------- 0xFF Divide system clock by 256
24–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 FRZ	Freeze. Allows the timer counter to be stopped when the device enters debug mode. **NOTE:** When the MCU enters debug mode, the STM is notified and uses the FRZ bit to determine counter mode. 0 STM counter continues to run in debug mode. 1 STM counter is stopped in debug mode.
31 TEN	Timer counter Enabled. 0 Counter is disabled. 1 Counter is enabled.

39.4.2 STM Count Register (STM_CNT)

The STM Count Register holds the timer count value.

Address: 0h base + 4h offset = 4h



STM_CNT field descriptions

Field	Description
0–31 CNT	Timer count value used as the time base for all channels. When enabled, the counter increments at the rate of the system clock divided by the prescale value.

39.4.3 STM Channel Control Register (STM_CCR*n*)

The STM Channel Control Register (STM_CCRn) has the enable bit for channel n of the timer.

Address: 0h base + 10h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															CEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM_CCR*n* field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CEN	Channel Enable 0 The channel is disabled. 1 The channel is enabled.

39.4.4 STM Channel Interrupt Register (STM_CIR*n*)

The STM Channel Interrupt Register has the interrupt flag for channel n of the timer.

Address: 0h base + 14h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															CIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM_CIR*n* field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CIF	Channel Interrupt Flag 0 No interrupt request. 1 Interrupt request due to a match on the channel.

39.4.5 STM Channel Compare Register (STM_CMP*n*)

The STM channel compare register (STM_CMPn) holds the compare value for channel n.

Address: 0h base + 18h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CMP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM_CMP*n* field descriptions

Field	Description
0–31 CMP	Compare value for channel n. If the STM_CCRn[CEN] bit is set and the STM_CMPn register matches the STM_CNT register, a channel interrupt request is generated and the STM_CIRn[CIF] bit is set.

39.5 Functional description

The STM has one 32-bit up counter (STM_CNT) that is used as the time base for all channels. When enabled, the counter increments at the system clock frequency divided by a prescale value. The STM_CR[CPS] field sets the divider to any value in the range from 1 to 256. The counter is enabled with the STM_CR[TEN] bit. When enabled in normal mode, the counter continuously increments. When enabled in debug mode, the counter operation is controlled by the STM_CR[FRZ] bit. When the STM_CR[FRZ] bit is set, the counter is stopped in debug mode; otherwise, it continues to run in debug mode. The counter rolls over at 0xFFFF_FFFF to 0x0000_0000 with no restrictions at this boundary.

The STM has four identical compare channels. Each channel includes a channel control register (STM_CCR*n*), a channel interrupt register (STM_CIR*n*), and a channel compare register (STM_CMP*n*). The channel is enabled by setting the STM_CCR*n*[CEN] bit.

When enabled, the channel will set the STM_CIR*n*[CIF] bit and generate an interrupt request when the channel compare register matches the timer counter. The interrupt request is cleared by writing 1 to the STM_CIR*n*[CIF] bit. A write of 0 to the STM_CIR*n*[CIF] bit has no effect.

NOTE

The STM counter does not advance when the system clock is stopped.

Chapter 40
Software Watchdog Timer (SWT)

40.1 Software Watchdog Timer (SWT) chip-specific module information

40.1.1 Implementation

- Internal 16MHz RC oscillator clock is used for timer operation
- Programmable selection of “instruction address” or “legacy” servicing
  - For “instruction address” servicing: programmable selection of fixed or changing address
  - For “legacy” servicing: Programmable selection of fixed or keyed servicing
- Master access protection

40.1.2 Configuration

MPC5746R implements three SWTs:

- SWT0 (CPU0)
- SWT1 (CPU1), enabled by default

NOTE

CPU1 can access SWT0 and CPU0 can access SWT1, but by using SMPU we can restrict the access of CPU0 to only SWT0 and same can be done for CPU1.

- SWT3 (Security: accessible by CPU0 and CPU1)

MPC5746R-specific register reset values are shown in the following table:

Table 40-1. Register reset values

Register	Reset value
SWT0_CR	0xFF00_010A
SWT0_TO	0x0005_FCD0
SWT1_CR	0xFF00_011B
SWT1_TO	0x0003_FDE0
SWT3_CR	0xFF00_010A
SWT3_TO	0x0003_FDE0

SWT_TO register: The IRCOSC is used to drive the down counter.

40.1.3 SWT_TO register minimum value

The SWT Time-out Register (SWT_TO) holds the watchdog time-out period in clock cycles. The minimum value is 100h. If a value smaller than the minimum value is written to the register, the time-out value is set to 100h.

40.1.4 SWT time-out events: configuring reset and interrupt requests

Configure the outcomes of SWT time-out events by programming ITR in the SWT Control Register (SWT_CR) as well as FCCU reactions to associated non-critical faults (NCFs). The following table shows the NCFs to which the SWT generates reset and interrupt requests.

Table 40-2. SWT reset and interrupt request routing

SWT instance	SWT_CR[ITR] is 0	SWT_CR[ITR] is 1	
	Any time-out generates reset request to	Initial time-out generates interrupt request to	Second consecutive time-out generates reset request to
SWT_0	NCF[21]	IRQ[32] and NCF[20]	NCF[21]
SWT_1	NCF[19]	IRQ[33]	NCF[19]
SWT_3	NCF[17]	IRQ[35]	NCF[17]

CAUTION

For any NCF that receives an SWT reset request: If the default FCCU reaction is no reset but the intended application requires a reset, configure the FCCU's reaction to be a long functional reset or short functional reset.

40.2 Introduction

This section provides an overview, list of features, and modes of operation for the SWT.

40.2.1 Overview

The Software Watchdog Timer (SWT) is a peripheral module that can prevent system lockup in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. When enabled, the SWT requires periodic execution of a watchdog servicing operation. The servicing operation resets the timer to a specified time-out period. If this servicing action does not occur before the timer expires, the SWT generates an interrupt or hardware reset request. The SWT can be configured to generate a reset request or interrupt on an initial time-out. The SWT always generates a reset request on a second consecutive time-out.

40.2.2 Features

The SWT has the following features:

- 32-bit time-out register to set the time-out period
- Programmable selection of window mode or regular servicing
- Programmable selection of reset or interrupt on an initial time-out
- Programmable selection of the servicing mode
- Master access protection
- Hard and soft configuration lock bits
- Reset configuration inputs allow timer to be enabled out of reset

40.2.3 Modes of operation

The SWT supports three device modes of operation: normal, debug and stop. When the SWT is enabled in normal mode, its counter runs continuously. In debug mode, operation of the counter is controlled by the FRZ bit in the SWT_CR. If the FRZ bit is set, the counter is stopped in debug mode, otherwise it continues to run. In stop mode, operation of the counter is controlled by the STP bit in the SWT_CR. If the STP bit is set, the counter is stopped in stop mode, otherwise it continues to run.

40.3 External signal description

The SWT module does not have any external interface signals.

40.4 Memory Map and Registers

The SWT programming model has seven 32-bit registers. The programming model can only be accessed using 32-bit (word) accesses. References using a different size are invalid. Other types of invalid accesses include: writes to read-only registers, incorrect values written to the service register when enabled, accesses to reserved addresses and accesses by masters without permission. If the RIA bit in the SWT_CR is set then the SWT generates a system reset on an invalid access otherwise a bus error is generated. If either the HLK or SLK bits in the SWT_CR are set, then the SWT_CR, SWT_TO, SWT_WN, and SWT_SK registers are read-only.

The SWT memory map is shown in the following table.

SWT memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	SWT Control Register (SWT_CR)	32	R/W	See section	40.4.1/1205
4	SWT Interrupt Register (SWT_IR)	32	R/W	0000_0000h	40.4.2/1208
8	SWT Time-out Register (SWT_TO)	32	R/W	See section	40.4.3/1208
C	SWT Window Register (SWT_WN)	32	R/W	0000_0000h	40.4.4/1209
10	SWT Service Register (SWT_SR)	32	W	0000_0000h	40.4.5/1209
14	SWT Counter Output Register (SWT_CO)	32	R	0000_0000h	40.4.6/1210
18	SWT Service Key Register (SWT_SK)	32	R/W	0000_0000h	40.4.7/1211

40.4.1 SWT Control Register (SWT_CR)

NOTE

The reset value of this register is implementation specific. See the chip-specific SWT information.

The SWT_CR contains fields for configuring and controlling the SWT.

This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MAP0	MAP1	MAP2	MAP3	MAP4	MAP5	MAP6	MAP7	0							
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SMD		RIA	WND	ITR	HLK	SLK	Reserved	STP	FRZ	WEN
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value of this register is implementation specific. See the chip-specific SWT information.

SWT_CR field descriptions

Field	Description
0 MAP0	Master Access Protection for Master 0. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled
1 MAP1	Master Access Protection for Master 1. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled
2 MAP2	Master Access Protection for Master 2. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch.

Table continues on the next page...

SWT_CR field descriptions (continued)

Field	Description
	0 Access for the master is not enabled 1 Access for the master is enabled
3 MAP3	Master Access Protection for Master 3. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled
4 MAP4	Master Access Protection for Master 4. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled
5 MAP5	Master Access Protection for Master 5. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled
6 MAP6	Master Access Protection for Master 6. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled
7 MAP7	Master Access Protection for Master 7. The platform bus master assignments are device specific. Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch. 0 Access for the master is not enabled 1 Access for the master is enabled
8–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–22 SMD	Service Mode. 00 Fixed Service Sequence, the watchdog is serviced by writing the fixed sequence 0xA602, 0xB480 to the SWT_SR. 01 Keyed Service Sequence, the watchdog is serviced by writing two pseudorandom key values to the SWT_SR. 10 Fixed Address Execution, the watchdog is serviced by executing code at the address loaded into the designated IAC register, which cannot be updated while the watchdog is enabled. 11 Incremental Address Execution, the watchdog is serviced by executing code at the address loaded into the designated IAC register, which can be updated.
23 RIA	Reset on Invalid Access. 0 Invalid access to the SWT generates a bus error 1 Invalid access to the SWT causes a system reset if WEN=1

Table continues on the next page...

SWT_CR field descriptions (continued)

Field	Description
24 WND	Window Mode. 0 Regular mode, service sequence can be done at any time 1 Windowed mode, the service sequence is only valid when the down counter is less than the value in the SWT_WN register.
25 ITR	Interrupt Then Reset **NOTE:** For a description of how this chip implements SWT reset requests resulting from SWT time-outs, see the chip-specific SWT information. 0 Generate a reset request on a time-out 1 Generate an interrupt on an initial time-out; generate a reset request on a second consecutive time-out
26 HLK	Hard Lock. This bit is cleared only at reset. 0 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read/write registers if SLK=0 1 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read-only registers
27 SLK	Soft Lock. This bit is cleared by writing the unlock sequence to the service register. 0 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read/write registers if HLK=0 1 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read-only registers
28 Reserved	Reserved This field is reserved.
29 STP	Stop Mode Control. Allows the watchdog timer to be stopped when the device enters stop mode. 0 SWT counter continues to run in stop mode 1 SWT counter is stopped in stop mode
30 FRZ	Debug Mode Control. Allows the watchdog timer to be stopped when the device enters debug mode. 0 SWT counter continues to run in debug mode 1 SWT counter is stopped in debug mode
31 WEN	Watchdog Enabled. 0 SWT is disabled 1 SWT is enabled

40.4.2 SWT Interrupt Register (SWT_IR)

The SWT_IR contains the time-out interrupt flag.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SWT_IR field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TIF	Time-out Interrupt Flag The flag and interrupt are cleared by writing a 1 to this bit. Writing a 0 has no effect. 0 No interrupt request 1 Interrupt request due to an initial time-out

40.4.3 SWT Time-out Register (SWT_TO)

NOTE

The reset value of this register is implementation specific. See the chip-specific SWT information.

The SWT Time-out (SWT_TO) register contains the 32-bit time-out period. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WTO																															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- The reset value of this register is implementation specific. See the chip-specific SWT information.

SWT_TO field descriptions

Field	Description
0–31 WTO	Watchdog time-out period in clock cycles. An internal 32-bit down counter is loaded with this value or 0x100, whichever is greater, when the service sequence is written or when the SWT is enabled.

40.4.4 SWT Window Register (SWT_WN)

The SWT Window (SWT_WN) register contains the 32-bit window start value. This register is cleared on reset. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WST																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SWT_WN field descriptions

Field	Description
0–31 WST	Window Start Value When window mode is enabled, the service sequence can only be written when the internal down counter is less than this value.

40.4.5 SWT Service Register (SWT_SR)

The SWT Service (SWT_SR) register is the target for service operation writes used to reset the watchdog timer.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																															
W																	WSC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SWT_SR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 WSC	Watchdog Service Code This field is used to service the watchdog and to clear the soft lock bit (SWT_CR[SLK]). If the SWT_CR[SMD] field is 01b, two pseudorandom key values are written to service the watchdog, see Functional description for details. Otherwise, the sequence 0xA602 followed by 0xB480 is written to the WSC field. To clear the soft lock bit (SWT_CR[SLK]), the value 0xC520 followed by 0xD928 is written to the WSC field. When read, the WSC field always returns zero.

40.4.6 SWT Counter Output Register (SWT_CO)

The SWT Counter Output (SWT_CO) register is a read-only register that shows the value of the internal down counter when the SWT is disabled.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SWT_CO field descriptions

Field	Description
0–31 CNT	Watchdog Count. When the watchdog is disabled (SWT_CR[WEN]=0), this field shows the value of the internal down counter. When the watchdog is enabled (SWT_CR[WEN]=1), this field is cleared (the value is 0x0000_0000). Values in this field can lag behind the internal counter value for up to six system plus eight counter clock cycles. Therefore, the value read from this field immediately after disabling the watchdog may be higher than the actual value of the internal counter.

40.4.7 SWT Service Key Register (SWT_SK)

The SWT Service Key (SWT_SK) register holds the previous (or initial) service key value. This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																SK															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SWT_SK field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 SK	Service Key This field is the previous (or initial) service key value used in keyed service mode. If SWT_CR[SMD] is 01b, the next key value to be written to the SWT_SR is (17*SK+3) mod 2^{16} .

40.5 Functional description

40.5.1 Introduction

The SWT is a 32-bit window watchdog timer designed to enable the system to recover in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. It includes:

- a control register (SWT_CR),
- an interrupt register (SWT_IR),
- a time-out register (SWT_TO),
- a window register (SWT_WN),
- a service register (SWT_SR),
- a counter output register (SWT_CO), and
- a service key register (SWT_SK).

Accesses to SWT registers occur with no peripheral bus wait states. (The peripheral bus bridge may add one or more system wait states.) However, due to synchronization logic in the SWT design, recognition of the service sequence or configuration changes may require up to three system plus seven counter clock cycles.

40.5.1.1 SWT_CR register

The SWT_CR register includes bits to enable the timer, set configuration options, and lock configuration of the module. The watchdog is enabled by setting the SWT_CR[WEN] bit. The reset value of the SWT_CR[WEN] bit is device specific.[3] If the reset value of this bit is 1, the watchdog starts operation automatically after reset is released. Some devices can be configured to clear this bit automatically during the boot process.

40.5.1.2 SWT_TO register

The SWT_TO register holds the watchdog time-out period in clock cycles unless the value is less than 0x100, in which case the time-out period is set to 0x100. When the SWT is enabled, the time-out period is loaded into an internal 32-bit down counter each time a valid service operation is performed. See the Configuration section to determine the source that is used to clock the down counter.

40.5.1.3 SWT_CO register

The SWT_CO register shows the value of the down counter when the watchdog is disabled. When the watchdog is enabled this register is cleared. The value shown in this register can lag behind the value in the internal counter for up to six system plus eight counter clock cycles.

The SWT_CO register can be used during a software self test of the SWT. For example, the SWT can be enabled and not serviced for a fixed period of time less than the time-out value. Then the SWT can be disabled (SWT_CR[WEN] cleared) and the value of the SWT_CO register read to determine if the internal down counter is working properly.

3. See the chip-specific SWT information.

40.5.2 Configuration locking

The configuration of the SWT can be locked through use of either a soft lock or a hard lock. In either case, when locked, the SWT_CR, SWT_TO, SWT_WN and SWT_SK registers are read-only.

40.5.2.1 Hard lock

The hard lock is enabled by setting the SWT_CR[HLK] bit, which can only be cleared by a reset.

40.5.2.2 Soft lock

The soft lock is enabled by setting the SWT_CR[SLK] bit and is cleared by writing the unlock sequence to the service register.

40.5.3 Unlock sequence

The unlock sequence is a write of 0xC520 followed by a write of 0xD928 to the SWT_SR[WSC] field. There is no timing requirement between the two writes. The unlock sequence logic ignores service sequence writes and recognizes the 0xC520, 0xD928 sequence regardless of previous writes. The unlock sequence can be written at any time and does not require the SWT_CR[WEN] bit to be set.

40.5.4 Servicing operations

When enabled, the SWT requires periodic execution of a servicing operation that is determined by the SWT_CR[SMD] field. Properly servicing the watchdog loads the internal down counter with the time-out period. The servicing modes are:

- fixed service sequence
- keyed service sequence
- fixed execution address
- incremental execution address

40.5.4.1 Fixed service sequence mode

If the SWT_CR[SMD] field is 00b, the fixed service sequence mode is selected, which requires writing 0xA602, then 0xB480 to the SWT_SR[WSC] field to service the watchdog. There is no timing requirement between the two writes and the service sequence logic ignores unlock sequence writes.

40.5.4.2 Keyed service sequence mode

If the SWT_CR[SMD] field is 01b, then the keyed service sequence mode is selected, which requires writing two pseudorandom keys to the SWT_SR[WSC] field to service the watchdog. The key values are determined by the pseudorandom key generator defined by the equation in the following figure. This algorithm will generate a sequence of 2^{16} different key values before repeating. The state of the key generator is held in the SWT_SK register. For example, if SWT_SK[SK] is 0x0100, then the service sequence keys are 0x1103, 0x2136. In this mode, each time a valid key is written to the SWT_SR register, the SWT_SK register is updated. So, after servicing the watchdog by writing 0x1103 and then 0x2136 to the SWT_SR[WSC] field, SWT_SK[SK] is 0x2136 and the next key sequence is 0x3499, 0x7E2C.

$$SK_{n+1} = (17 \times SK_n + 3) \bmod 2^{16}$$

Figure 40-8. Pseudorandom Key Generator

40.5.4.3 Fixed execution address mode

If the SWT_CR[SMD] field is set to 10b, the fixed execution address mode is selected, which requires executing code at the address loaded into the designated IAC register to service the watchdog. In this mode, the IAC register cannot be updated while the watchdog is enabled.

40.5.4.4 Incremental execution address mode

If the SWT_CR[SMD] field is set to 11b, the incremental execution address mode is selected, which requires executing code at the address loaded into the designated IAC register to service the watchdog. In this mode, the IAC register can be updated while the watchdog is enabled.

40.5.4.5 Window mode

If window mode is enabled (SWT_CR[WND] bit is set), the service sequence must be performed in the last part of the time-out period defined by the window register. The window is open when the down counter is less than the value in the SWT_WN register. Outside of this window, service sequence writes are invalid accesses and generate a bus error or reset depending on the value of the SWT_CR[RIA] bit. For example, if the SWT_TO register is set to 5000 and SWT_WN register is set to 1000, then the service sequence must be performed in the last 20% of the time-out period. There is a short lag in the time it takes for the window to open due to synchronization logic in the watchdog design. This delay could be up to three system plus four counter clock cycles.

40.5.5 Time-out

The Interrupt Then Reset bit (SWT_CR[ITR]) controls the SWT's action taken when a time-out occurs.

- If the SWT_CR[ITR] bit is 0, the SWT generates a reset request immediately on any time-out.
- If the SWT_CR[ITR] bit is 1, an initial time-out causes the SWT to generate an interrupt and load the down counter with the time-out period.
  - The interrupt is indicated by the Time-out Interrupt Flag (SWT_IR[TIF]).
  - Clear the interrupt by writing 1 to SWT_IR[TIF].

  If the service sequence is not written before the second consecutive time-out, the SWT generates a reset request.

40.5.6 Initialization

All registers should be initialized before setting the SWT_CR[WEN] bit to enable the watchdog. Registers can be initialized in any sequence.

Chapter 41
Periodic Interrupt Timer (PIT-RTI)

41.1 Periodic Interrupt Timer (PIT-RTI) chip-specific module information

41.1.1 Overview

The PIT is an array of timers that can be used to raise interrupts and trigger DMA channels. On some devices it provides a dedicated Real Time interrupt Timer (RTI), which runs on a separate IRC clock and can be used for system wake-up.

41.1.2 Configuration

MCP5746R implements two instances of the Periodic Interrupt Timer Module (PIT), PIT0 and PIT1. Each has a different feature set as shown in the following table.

Table 41-1. PIT0 and PIT1 feature set differences

Feature	PIT0	PIT1
Real Time Interrupt (RTI)	Yes	No
Lifetime Timer	No	Yes
Timer Chaining to create a 64-bit timer	No	Yes
Number of 32-bit timer channels	8	2
Timer value unchanged by a non-destructive reset	No	Yes

41.1.3 General-use PIT

The following sections describe the superset of features available for the PIT module and might describe features not implemented on this MCU. The table above shows features that are implemented on this MCU.

PIT0 has eight 32-bit timer channels. Accordingly, the register map shows eight each of the LDVAL, CVAL, TCTRL, and TFLG registers. PIT1 has only two channels and therefore the latter six of the eight registers that are shown would not be implemented.

41.2 Introduction

The following sections show the PIT's:

- Block diagram
- Features

41.2.1 Block diagram

The following figure shows the block diagram of the PIT-RTI module.

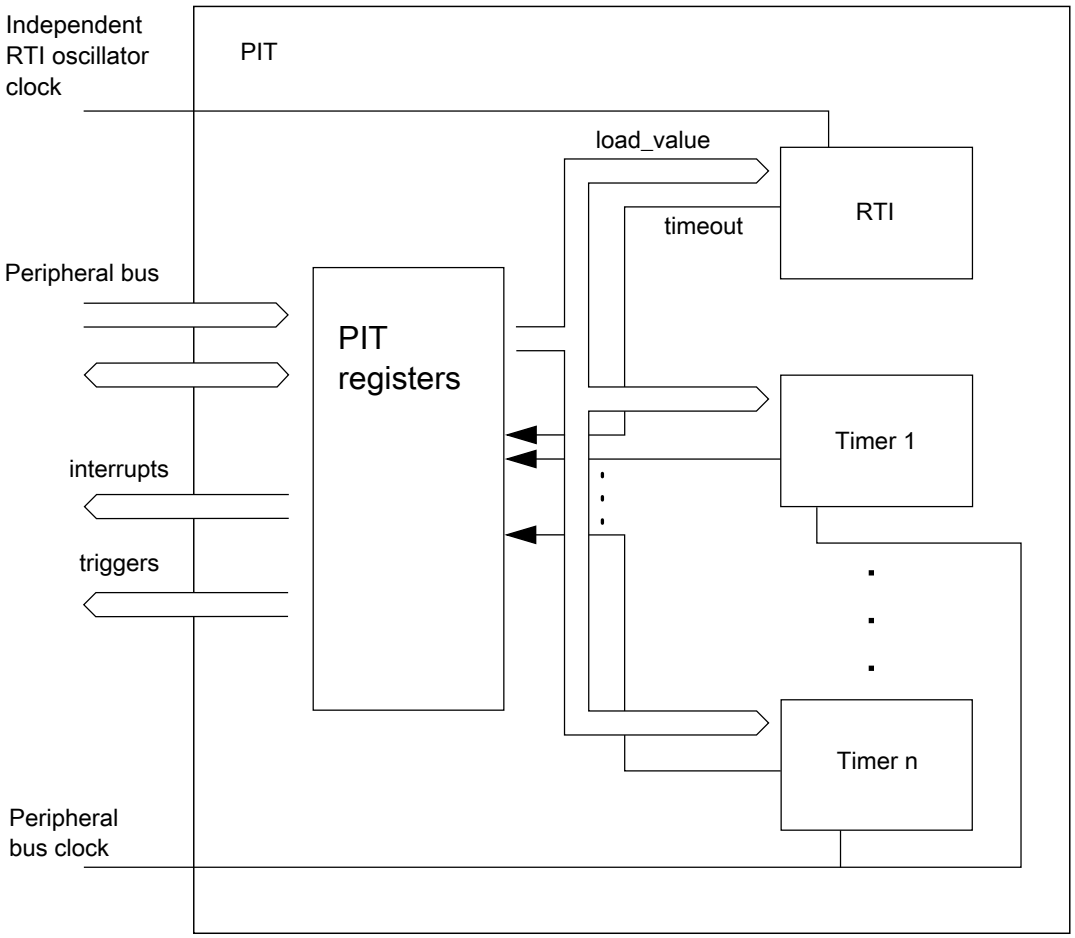


Figure 41-1. Block diagram of the PIT-RTI

NOTE

See the chip-specific PIT information for the number of PIT channels used in this MCU.

41.2.2 Features

The main features of this block are:

- One RTI (Real-Time Interrupt) timer to wakeup the CPU in stop mode
- Ability of timers to generate DMA trigger pulses
- Ability of timers to generate interrupts
- Maskable interrupts
- Option to raise RTI interrupt, even when the bus clock is switched off

- Power saving with a separate input clock for the RTI timer. All other timers share one common core clock.
- Independent timeout periods for each timer

41.3 Modes of operation

This subsection describes briefly all operating modes supported by the PIT.

- Run Mode

  All functional parts of the PIT are running during normal Run Mode.

- Stop Mode

  The RTI can continue to run in STOP mode.

41.4 Signal description

The PIT module has no external pins.

41.5 Memory map/register description

This section provides a detailed description of all registers accessible in the PIT_RTI module.

- Reserved registers will read as 0, writes will have no effect.
- See the chip-specific PIT information for the number of PIT channels used in this MCU.
- The RTI registers should be programmed only when the RTI clock is running.

NOTE

When a chip has more than one instance of the PIT, the different instances might have different feature and register sets. Please see the configuration information.

PIT memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	PIT Module Control Register (PIT_MCR)	32	R/W	0000_0006h	41.5.1/1222
E0	PIT Upper Lifetime Timer Register (PIT_LTMR64H)	32	R	0000_0000h	41.5.2/1223
E4	PIT Lower Lifetime Timer Register (PIT_LTMR64L)	32	R	0000_0000h	41.5.3/1224
F0	Timer Load Value Register (PIT_RTI_LDVAL)	32	R/W	0000_0000h	41.5.4/1224
F4	Current Timer Value Register (PIT_RTI_CVAL)	32	R	0000_0000h	41.5.5/1225
F8	Timer Control Register (PIT_RTI_TCTRL)	32	R/W	0000_0000h	41.5.6/1225
FC	Timer Flag Register (PIT_RTI_TFLG)	32	R/W	0000_0000h	41.5.7/1226
100	Timer Load Value Register (PIT_LDVAL0)	32	R/W	0000_0000h	41.5.8/1227
104	Current Timer Value Register (PIT_CVAL0)	32	R	0000_0000h	41.5.9/1227
108	Timer Control Register (PIT_TCTRL0)	32	R/W	0000_0000h	41.5.10/ 1228
10C	Timer Flag Register (PIT_TFLG0)	32	R/W	0000_0000h	41.5.11/ 1229
110	Timer Load Value Register (PIT_LDVAL1)	32	R/W	0000_0000h	41.5.8/1227
114	Current Timer Value Register (PIT_CVAL1)	32	R	0000_0000h	41.5.9/1227
118	Timer Control Register (PIT_TCTRL1)	32	R/W	0000_0000h	41.5.10/ 1228
11C	Timer Flag Register (PIT_TFLG1)	32	R/W	0000_0000h	41.5.11/ 1229
120	Timer Load Value Register (PIT_LDVAL2)	32	R/W	0000_0000h	41.5.8/1227
124	Current Timer Value Register (PIT_CVAL2)	32	R	0000_0000h	41.5.9/1227
128	Timer Control Register (PIT_TCTRL2)	32	R/W	0000_0000h	41.5.10/ 1228
12C	Timer Flag Register (PIT_TFLG2)	32	R/W	0000_0000h	41.5.11/ 1229
130	Timer Load Value Register (PIT_LDVAL3)	32	R/W	0000_0000h	41.5.8/1227
134	Current Timer Value Register (PIT_CVAL3)	32	R	0000_0000h	41.5.9/1227
138	Timer Control Register (PIT_TCTRL3)	32	R/W	0000_0000h	41.5.10/ 1228
13C	Timer Flag Register (PIT_TFLG3)	32	R/W	0000_0000h	41.5.11/ 1229
140	Timer Load Value Register (PIT_LDVAL4)	32	R/W	0000_0000h	41.5.8/1227
144	Current Timer Value Register (PIT_CVAL4)	32	R	0000_0000h	41.5.9/1227
148	Timer Control Register (PIT_TCTRL4)	32	R/W	0000_0000h	41.5.10/ 1228
14C	Timer Flag Register (PIT_TFLG4)	32	R/W	0000_0000h	41.5.11/ 1229
150	Timer Load Value Register (PIT_LDVAL5)	32	R/W	0000_0000h	41.5.8/1227
154	Current Timer Value Register (PIT_CVAL5)	32	R	0000_0000h	41.5.9/1227
158	Timer Control Register (PIT_TCTRL5)	32	R/W	0000_0000h	41.5.10/ 1228

Table continues on the next page...

PIT memory map (continued)

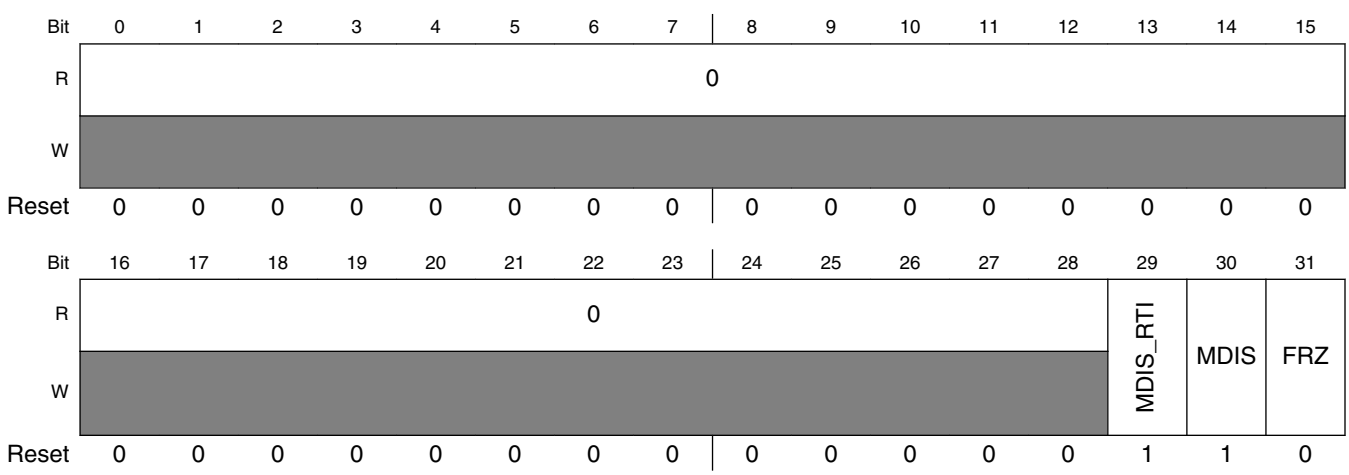
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
15C	Timer Flag Register (PIT_TFLG5)	32	R/W	0000_0000h	41.5.11/ 1229
160	Timer Load Value Register (PIT_LDVAL6)	32	R/W	0000_0000h	41.5.8/1227
164	Current Timer Value Register (PIT_CVAL6)	32	R	0000_0000h	41.5.9/1227
168	Timer Control Register (PIT_TCTRL6)	32	R/W	0000_0000h	41.5.10/ 1228
16C	Timer Flag Register (PIT_TFLG6)	32	R/W	0000_0000h	41.5.11/ 1229
170	Timer Load Value Register (PIT_LDVAL7)	32	R/W	0000_0000h	41.5.8/1227
174	Current Timer Value Register (PIT_CVAL7)	32	R	0000_0000h	41.5.9/1227
178	Timer Control Register (PIT_TCTRL7)	32	R/W	0000_0000h	41.5.10/ 1228
17C	Timer Flag Register (PIT_TFLG7)	32	R/W	0000_0000h	41.5.11/ 1229

41.5.1 PIT Module Control Register (PIT_MCR)

This register enables or disables the PIT timer clocks and controls the timers when the PIT enters the Debug mode.

Access: User read/write

Address: 0h base + 0h offset = 0h



PIT_MCR field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PIT_MCR field descriptions (continued)

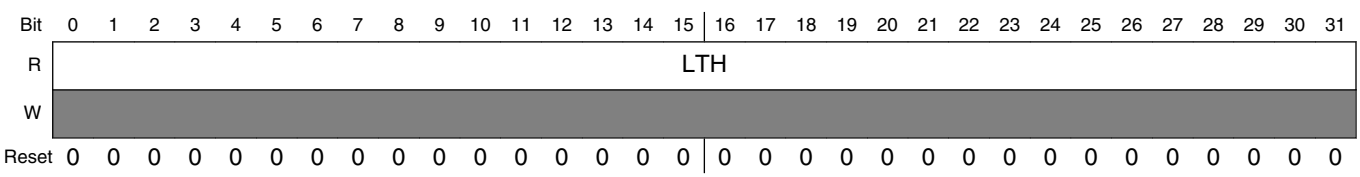
Field	Description
29 MDIS_RTI	Module Disable - RTI section This is used to disable the RTI timer. This bit must be enabled before any RTI setup is done. 0 Clock for RTI Timers is enabled 1 Clock for RTI Timers is disabled
30 MDIS	Module Disable - (PIT section) Disables the standard timers.The RTI timer is not affected by this field. This field must be enabled before any other setup is done. **NOTE:** Always write to this bit after at least 3 bus clock cycles of enabling the PIT clock gate in the device clock generation module. 0 Clock for standard PIT timers is enabled. 1 Clock for standard PIT timers is disabled.
31 FRZ	Freeze Allows the timers to be stopped when the device enters the Debug mode. 0 Timers continue to run in Debug mode. 1 Timers are stopped in Debug mode.

41.5.2 PIT Upper Lifetime Timer Register (PIT_LTMR64H)

This register is intended for applications that chain timer 0 and timer 1 to build a 64-bit lifetimer.

Access: User read only

Address: 0h base + E0h offset = E0h



PIT_LTMR64H field descriptions

Field	Description
0–31 LTH	Life Timer value Shows the timer value of timer 1. If this register is read at a time t1, LTMR64L shows the value of timer 0 at time t1.

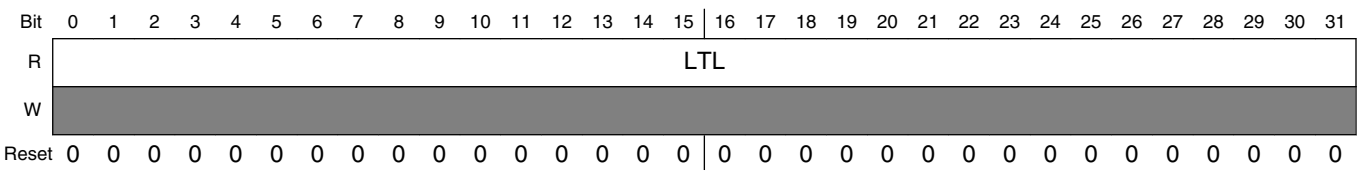
41.5.3 PIT Lower Lifetime Timer Register (PIT_LTMR64L)

This register is intended for applications that chain timer 0 and timer 1 to build a 64-bit lifetimer.

To use LTMR64H and LTMR64L, timer 0 and timer 1 need to be chained. To obtain the correct value, first read LTMR64H and then LTMR64L. LTMR64H will have the value of CVAL1 at the time of the first access, LTMR64L will have the value of CVAL0 at the time of the first access, therefore the application does not need to worry about carry-over effects of the running counter.

Access: User read only

Address: 0h base + E4h offset = E4h



PIT_LTMR64L field descriptions

Field	Description
0–31 LTL	Life Timer value Shows the value of timer 0 at the time LTMR64H was last read. It will only update if LTMR64H is read.

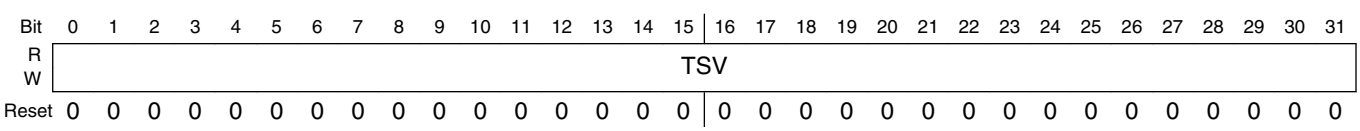
41.5.4 Timer Load Value Register (PIT_RTI_LDVAL)

These registers select the timeout period for the timer interrupts.

In the case of the RTI, it will take several cycles until this value is synchronized into the RTI clock domain. For all other timers the value change is visible immediately. The synchronisation mechanism allows 0 wait states in this case.

Access: User read/write

Address: 0h base + F0h offset = F0h



PIT_RTI_LDVAL field descriptions

Field	Description
0–31 TSV	Timer Start Value Sets the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer; instead the value will be loaded after the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again. **NOTE:** The RTI timer must not be set to a value lower than 32 cycles, otherwise interrupts may be lost, as it takes several cycles to clear the RTI interrupt. For the other timers, this limit does not apply, however there will be practical limits, since the processor will require several cycles to service an interrupt.

41.5.5 Current Timer Value Register (PIT_RTI_CVAL)

These registers indicate the current timer position.

For the RTI timers, they show a value which is several cycles old, since it originates from a potentially different clock domain.

Access: User read only

Address: 0h base + F4h offset = F4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TVL																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_RTI_CVAL field descriptions

Field	Description
0–31 TVL	Current Timer Value Represents the current timer value, if the timer is enabled. **NOTE:** • If the timer is disabled, do not use this field as its value is unreliable. • The timer uses a downcounter. The timer values are frozen in Debug mode if MCR[FRZ] is set.

41.5.6 Timer Control Register (PIT_RTI_TCTRL)

These registers contain the control bits for each timer.

Access: User read/write

Address: 0h base + F8h offset = F8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0														TIE	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_RTI_TCTRL field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 TIE	Timer Interrupt Enable When an interrupt is pending or, RTI_TFLG[TIF] is set, enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated RTI_TFLG[TIF] flag must be cleared first. 0 Interrupt requests from the RTI Timer are disabled. 1 Interrupt will be requested whenever TIF is set.
31 TEN	Timer Enable Bit This bit enables or disables the timer. 0 RTI Timer is disabled. 1 RTI Timer is active.

41.5.7 Timer Flag Register (PIT_RTI_TFLG)

These registers hold the PIT interrupt flags.

Access: User read/write

Address: 0h base + FCh offset = FCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_RTI_TFLG field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TIF	Timer Interrupt Flag. TIF is set to 1 at the end of the timer period. This flag can be cleared only by writing it with 1 . Writing 0 has no effect. If enabled (TIE = 1), TIF causes an interrupt request. 0 Time-out has not yet occurred. 1 Time-out has occurred.

41.5.8 Timer Load Value Register (PIT_LDVAL*n*)

These registers select the timeout period for the timer interrupts.

Access: User read/write

Address: 0h base + 100h offset + (16d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TSV																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_LDVAL*n* field descriptions

Field	Description
0–31 TSV	Timer Start Value Sets the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer; instead the value will be loaded after the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again.

41.5.9 Current Timer Value Register (PIT_CVAL*n*)

These registers indicate the current timer position.

Access: User read only

Address: 0h base + 104h offset + (16d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TVL																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_CVAL*n* field descriptions

Field	Description
0–31 TVL	Current Timer Value Represents the current timer value, if the timer is enabled. **NOTE:** • If the timer is disabled, do not use this field as its value is unreliable. • The timer uses a downcounter. The timer values are frozen in Debug mode if MCR[FRZ] is set.

41.5.10 Timer Control Register (PIT_TCTRL*n*)

These registers contain the control bits for each timer.

Access: User read/write

Address: 0h base + 108h offset + (16d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0													CHN	TIE	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_TCTRL*n* field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 CHN	Chain Mode When activated, Timer n-1 needs to expire before timer n can decrement by 1. Timer 0 cannot be chained. The RTI timer cannot be chained. 0 Timer is not chained. 1 Timer is chained to previous timer. For example, for Channel 2, if this field is set, Timer 2 is chained to Timer 1.
30 TIE	Timer Interrupt Enable When an interrupt is pending, or, TFLGn[TIF] is set, enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TFLGn[TIF] must be cleared first. 0 Interrupt requests from Timer n are disabled. 1 Interrupt will be requested whenever TIF is set.
31 TEN	Timer Enable Enables or disables the timer.

Table continues on the next page...

PIT_TCTRL*n* field descriptions (continued)

Field	Description
	0 Timer n is disabled. 1 Timer n is enabled.

41.5.11 Timer Flag Register (PIT_TFLG*n*)

These registers hold the PIT interrupt flags.

Access: User read/write

Address: 0h base + 10Ch offset + (16d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_TFLG*n* field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TIF	Timer Interrupt Flag Sets to 1 at the end of the timer period. Writing 1 to this flag clears it. Writing 0 has no effect. If enabled, or, when TCTRLn[TIE] = 1, TIF causes an interrupt request. 0 Timeout has not yet occurred. 1 Timeout has occurred.

41.6 Functional description

This section provides the functional description of the module.

41.6.1 General operation

This section gives detailed information on the internal operation of the module. Each timer can be used to generate trigger pulses and interrupts. Each interrupt is available on a separate interrupt line. Additionally the RTI timer can be used to wake up the processor.

41.6.1.1 Timers / RTI

The timers generate triggers at periodic intervals, when enabled. The timers load the start values as specified in their LDVAL registers, count down to 0 and then load the respective start value again. Each time a timer reaches 0, it will generate a trigger pulse and set the interrupt flag.

All interrupts can be enabled or masked by setting TCTRLn[TIE]. A new interrupt can be generated only after the previous one is cleared. Since in the case of the RTI, clearing the interrupt crosses clock domains, a minimum load value of 32 must be maintained.

If desired, the current counter value of the timer can be read via the CVAL registers. The value of the RTI counter can be delayed considerably, as it is synchronized to the bus clock from the RTI clock domain.

The counter period can be restarted, by first disabling, and then enabling the timer with TCTRLn[TEN]. See the following figure.

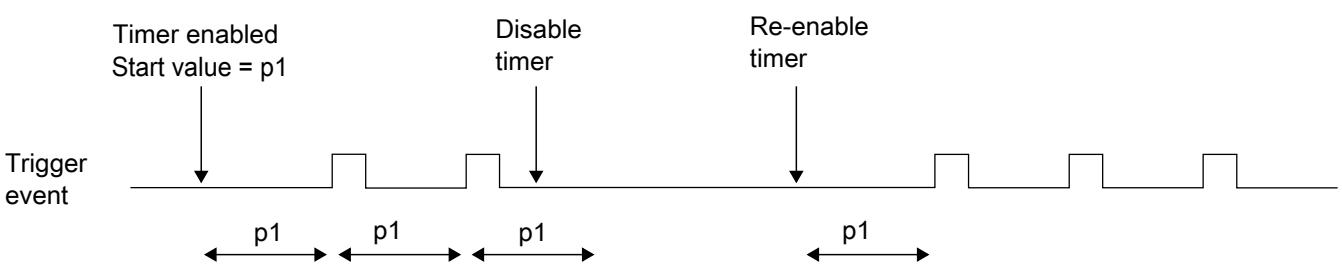


Figure 41-45. Stopping and starting a timer

The counter period of a running timer can be modified, by first disabling the timer, setting a new load value, and then enabling the timer again. See the following figure.

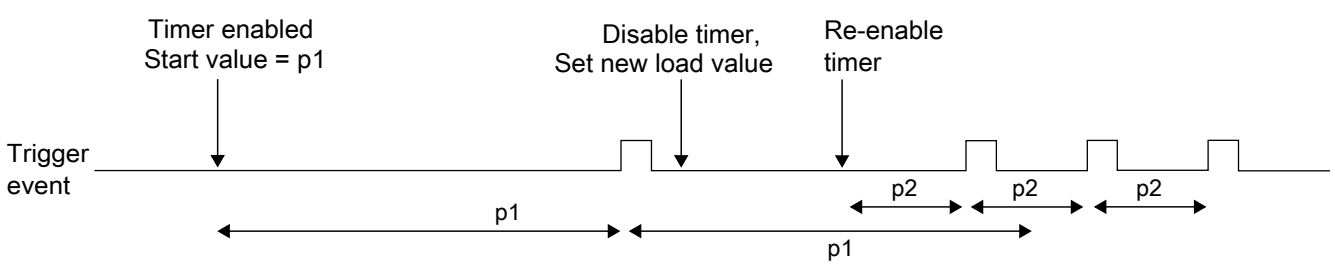


Figure 41-46. Modifying running timer period

It is also possible to change the counter period without restarting the timer by writing LDVAL with the new load value. This value will then be loaded after the next trigger event. See the following figure.

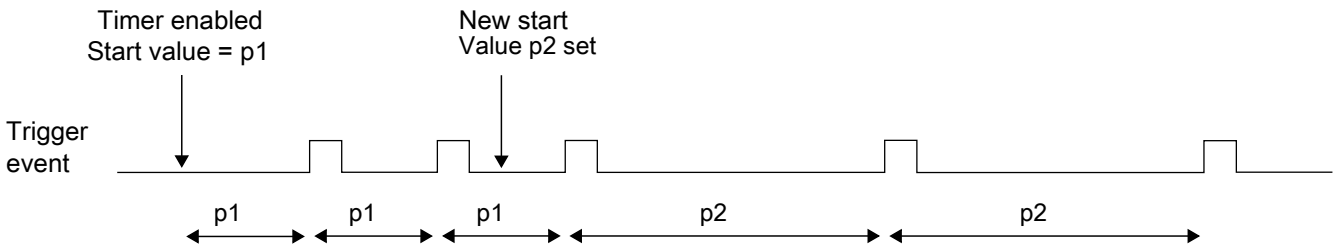


Figure 41-47. Dynamically setting a new load value

41.6.1.2 Debug mode

In Debug mode, the timers will be frozen based on MCR[FRZ]. This is intended to aid software development, allowing the developer to halt the processor, investigate the current state of the system, for example, the timer values, and then continue the operation.

41.6.2 Interrupts

All the timers support interrupt generation. The RTI is typically used for system wakeup, but can be used for interrupt generation as well. See the MCU specification for related vector addresses and priorities.

Timer interrupts can be enabled by setting TCTRLn[TIE]. TFLGn[TIF] are set to 1 when a timeout occurs on the associated timer, and are cleared to 0 by writing a 1 to the corresponding TFLGn[TIF].

The PIT_RTI generates a real time interrupt when the selected interrupt time period elapses. The RTI interrupt is disabled locally by setting TCTCRLn[TIE] to zero. The real time interrupt flag (TIF) is set to 1 when a timeout occurs, and is cleared by writing a 1 to the corresponding TFLGn[TIF]. The flag will be set regardless whether the interrupt is enabled.

The RTI can be used for periodic wakeup from STOP mode. It can also be used to generate a general-purpose interrupt.

41.6.3 Chained timers

When a timer has chain mode enabled, it will only count after the previous timer has expired. So if timer n-1 has counted down to 0, counter n will decrement the value by one. This allows to chain some of the timers together to form a longer timer. The first timer (timer 0) cannot be chained to any other timer.

41.7 Initialization and application information

In the example configuration:

- The PIT clock has a frequency of 50 MHz.
- The RTI clock has a frequency of 10 MHz.
- The RTI creates a wakeup interrupt every 500 ms.
- Timer 1 creates an interrupt every 5.12 ms.
- Timer 3 creates a trigger event every 30 ms.

The PIT module must be activated by writing a 0 to MCR[MDIS].

The 50 MHz clock frequency equates to a clock period of 20 ns and the 10 MHz frequency equates to a clock period of 100 ns. Therefore, the RTI timer needs to trigger every 500 ms/100 ns = 5,000,000 cycles. Timer 1 needs to trigger every 5.12 ms/20 ns = 256,000 cycles and Timer 3 every 30 ms/20 ns = 1,500,000 cycles. The value for the LDVAL register trigger is calculated as:

LDVAL trigger = (period / clock period) -1

This means RTI LDVAL will be written with 004C4B3F hex; LDVAL1 and LDVAL3 must be written with 0x0003E7FF and 0x0016E35F respectively.

To generate the wakeup interrupt, the interrupt line must be enabled by writing 1 to the RTI TIE bit in the TCTRL register. To start the RTI, the TEN in the RTI TCTRL register must also be set.

The interrupt for Timer 1 is enabled by setting TCTRL1[TIE]. The timer is started by writing 1 to TCTRL1[TEN].

Timer 3 shall be used only for triggering. Therefore, Timer 3 is started by writing a 1 to TCTRL3[TEN]. TCTRL3[TIE] stays at 0.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;


// RTI
PIT_RTI_LDVAL = 0x004C4B3F; // setup RTI for 5000000 cycles
PIT_RTI_TCTRL = PIT_TIE; // let RTI generate interrupts
PIT_RTI_TCTRL |= PIT_TEN; // start RTI


// Timer 1
PIT_LDVAL1 = 0x0003E7FF; // setup timer 1 for 256000 cycles
PIT_TCTRL1 = TIE; // enable Timer 1 interrupts
PIT_TCTRL1 |= TEN; // start Timer 1


// Timer 3
PIT_LDVAL3 = 0x0016E35F; // setup timer 3 for 1500000 cycles
PIT_TCTRL3 |= TEN; // start Timer 3
```

41.8 Example configuration for chained timers

In the example configuration:

- The PIT clock has a frequency of 100 MHz.
- Timers 1 and 2 are available.
- An interrupt shall be raised every 1 minute.

The PIT module needs to be activated by writing a 0 to MCR[MDIS].

The 100 MHz clock frequency equates to a clock period of 10 ns, so the PIT needs to count for 6000 million cycles, which is more than a single timer can do. So, Timer 1 is set up to trigger every 6 s (600 million cycles). Timer 2 is chained to Timer 1 and programmed to trigger 10 times.

The value for the LDVAL register trigger is calculated as number of cycles-1, so LDVAL1 receives the value 0x23C345FF and LDVAL2 receives the value 0x00000009.

The interrupt for Timer 2 is enabled by setting TCTRL2[TIE], the Chain mode is activated by setting TCTRL2[CHN], and the timer is started by writing a 1 to TCTRL2[TEN]. TCTRL1[TEN] needs to be set, and TCTRL1[CHN] and TCTRL1[TIE] are cleared.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;


// Timer 2
```

```
PIT_LDVAL2 = 0x00000009; // setup Timer 2 for 10 counts
PIT_TCTRL2 = TIE; // enable Timer 2 interrupt
PIT_TCTRL2 |= CHN; // chain Timer 2 to Timer 1
PIT_TCTRL2 |= TEN; // start Timer 2


// Timer 1
PIT_LDVAL1 = 0x23C345FF; // setup Timer 1 for 600 000 000 cycles
PIT_TCTRL1 = TEN; // start Timer 1
```

41.9 Example configuration for the lifetime timer

To configure the lifetimer timer, channels 0 and 1 need to be chained together.

First the PIT module needs to be activated by writing a 0 to the MDIS bit in the CTRL register, then the LDVAL registers need to be set to the maximum value.

The timer is a downcounter.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;


// Timer 1
PIT_LDVAL1 = 0xFFFFFFFF; // setup timer 1 for maximum counting period
PIT_TCTRL1 = 0x0; // disable timer 1 interrupts
PIT_TCTRL1 |= CHN; // chain timer 1 to timer 0
PIT_TCTRL1 |= TEN; // start timer 1


// Timer 0
PIT_LDVAL0 = 0xFFFFFFFF; // setup timer 0 for maximum counting period
PIT_TCTRL0 = TEN; // start timer 0
```

To access the lifetime, read first LTMR64H and then LTMR64L.

```
current_uptime = PIT_LTMR64H<<32;
current_uptime = current_uptime + PIT_LTMR64L;
```

Chapter 42
Enhanced Modular IO Subsystem (eMIOS200)

42.1 Enhanced Modular IO Subsystem (eMIOS253) chip-specific module information

42.1.1 Configuration

This chapter describes the full set of features offered by this eMIOS200 module, some of which may have been configured differently for this MPC5746R MCU.

In order to meet the enhanced safety requirements of MPC5746R, two identical eMIOS200 modules are implemented. They are located on separate peripheral bridges.

The following table describes the configuration of both eMIOS200 modules used on this MCU.

Table 42-1. Module configuration for this MCU

Item	Configuration
Channels implemented	8–23 (0–7 and 24–31 should not be used)
Channel modes	All modes are available for each channel.
Timer width	24-bits
STAC bus enabled?	Yes
UC prescaler width value	2
Clock source	DPLL0 clock (through a divider)

42.1.2 Global Timebase Enabling

The global time base enable output signal (GTBE_OUT) of both eMIOS and eTPU are ORed together and connected to the enable input signal (GTBE_IN). This has the effect that when the global time base enable bit is written in one module, all of the timebases in all of the other modules become enabled as well.

42.1.3 Clocking

The eMIOS253 is clocked by the PLL0 clock through a divider.

42.1.4 STAC bus

The eMIOS is a STAC bus client module and the STAC bus is connected to both eMIOS 253 modules.

NOTE

If the STAC-bus Export/Import feature is used, the clock ratio of eTPU : eMIOS should be 1:1 or 2:1, such as the following:

- eTPU 100MHz : eMIOS 100MHz
- eTPU 200MHz : eMIOS 100MHz

This applies whether the STAC-bus Export/Import is used from eTPU to eMIOS or used between eTPUs.

42.1.5 Output Disable Inputs

There are four output disable inputs to eMIOS200 module. The source of these inputs can be either eMIOS200 Flag output or Reaction channels output. The source selection is controlled through the following SIUL_MSCRn registers:

- eMIOS_0: SIUL2_MSCR606, SIUL2_MSCR607, SIUL2_MSCR608, and SIUL2_MSCR609
- eMIOS_1: SIUL2_MSCR632, SIUL2_MSCR633, SIUL2_MSCR634, and SIUL2_MSCR635

Please refer to the Device Input Multiplexing Tables (Excel file).

42.2 eMIOS200 Introduction

42.2.1 Overview

The configurable enhanced Modular Input/Output Subsystem (eMIOS200) provides functionality to generate or measure time events. Its overall architecture resembles that of the MIOS timer.

The predecessor MIOS timer block provides a framework in which a set of sublocks with different timer functions are assembled to attend to the specific needs of an SoC. The eMIOS200 builds on this concept by using a Unified Channel module that provides a superset of the functionality of all the individual MIOS channels, while also providing a consistent user interface. This allows more flexibility as each Unified Channel can be programmed for different functions in different applications of the SoC. In addition, eMIOS200 architecture allows the use of Dedicated Channels which perform specific functions not included in MIOS inheritance.

The block diagram of the configurable eMIOS200 block is shown below.

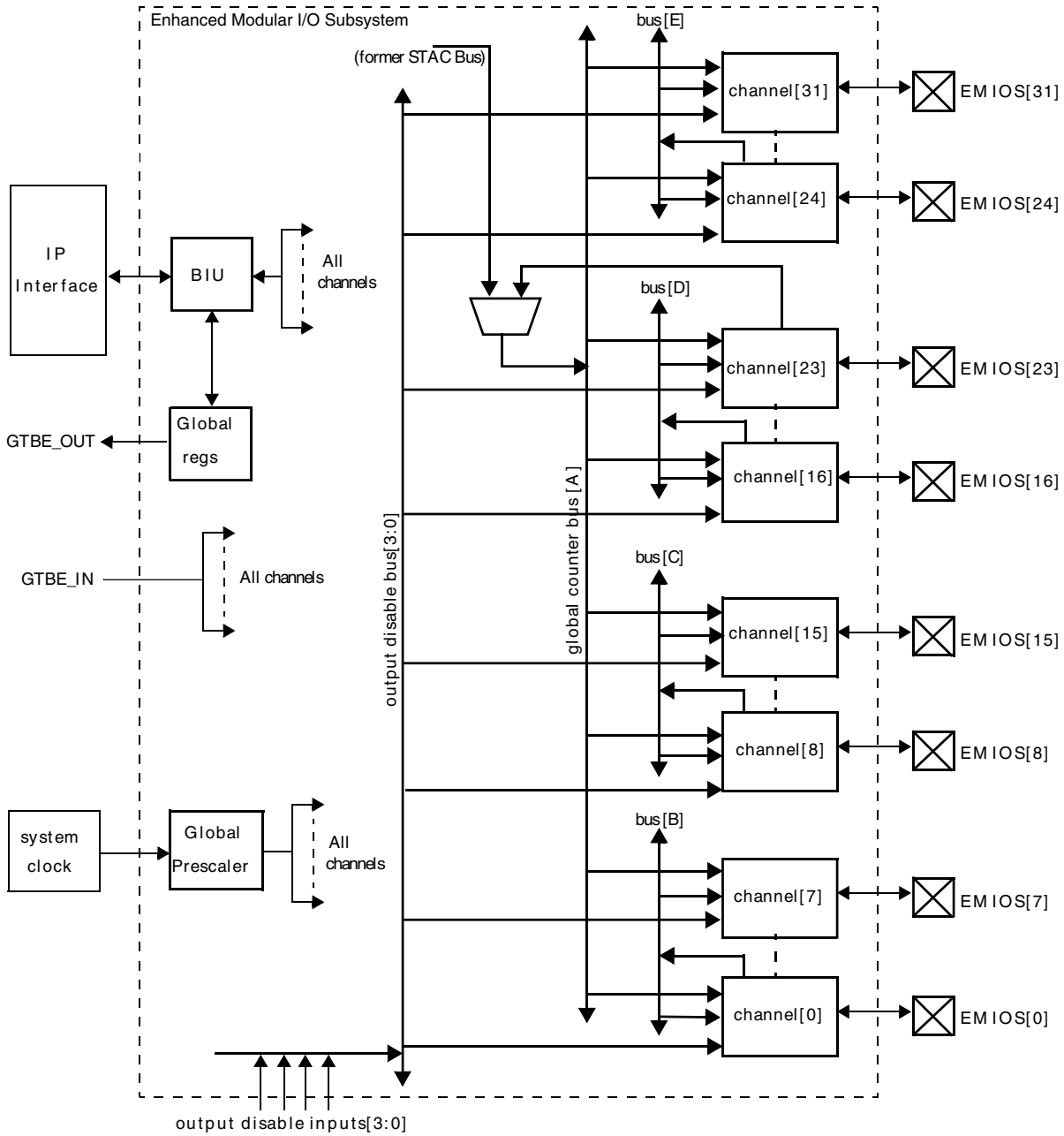


Figure 42-1. eMIOS200 full channel configuration using Unified Channels only (see Configuration section)

42.2.2 Features

The eMIOS200 has the following features:

- Up to 32 channels chosen among Unified or Dedicated Channels, not necessarily numbered in a continuous sequence. (See Configuration information)
- Data registers of either 8-, 16-, 24-, or 32-bit width. (See Configuration information)
- Counter buses B, C, D, and E can be driven by Unified Channels 0, 8, 16, and 24, respectively. (See Configuration information)
- Counter bus A can be driven by Unified Channel 23 or by the external shared timer bus(Red-Line Client).
- Each channel has its own timebase, alternative to the counter buses.
- One global prescaler
- One prescaler per channel (CP)
- Timebases shared through the counter buses
- One STAC Client (STAC)
- Control and status bits grouped in separated registers
- Synchronization among timebases
- Shadow FLAG register
- State of the Unified Channels can be frozen for debug purposes
- Motor control capability

42.2.3 Modes of operation

The Unified Channels can be configured to operate in the following modes:

- General-Purpose Input/Output (GPIO)
- Single-Action Input Capture (SAIC)
- Single-Action Output Compare (SAOC)
- Input Pulse-Width Measurement (IPWM)
- Input Period Measurement (IPM)
- Double-Action Output Compare (DAOC)
- Pulse/Edge Accumulation (PEA)
- Pulse/Edge Counting (PEC)

- Quadrature Decode (QDEC)
- Windowed Programmable Time Accumulation (WPTA)
- Modulus Counter (MC)
- Modulus Counter Buffered (MCB)
- Output Pulse-Width and Frequency Modulation (OPWFM)
- Output Pulse-Width and Frequency Modulation Buffered (OPWFMB)
- Center Aligned Output Pulse Width Modulation with Dead Time Insertion (OPWMC)
- Center Aligned Output Pulse Width Modulation with Dead Time Insertion Buffered (OPWMCB)
- Output Pulse-Width Modulation (OPWM)
- Output Pulse-Width Modulation Buffered (OPWMB)
- Output Pulse-Width Modulation with Trigger (OPWMT)

These modes are described in Unified Channel modes of operation.

Each channel can have a specific set of modes.

42.3 Memory map / register definition

42.3.1 Memory map

The overall address map organization is shown in the table below.

Whenever an access to either an absent register, absent channel, or reserved address space is performed, the transfer is terminated with an error.

Table 42-2. eMIOS200 Memory Map

eMIOS200[n] Base Address	Description
0x000	Module Configuration register (EMIOSMCR)
0x003	
0x004	Global FLAG register (EMIOSGFLAG)
0x007	
0x008	Output Update Disable (EMIOSOUDIS)
0x00B	

Table continues on the next page...

Table 42-2. eMIOS200 Memory Map (continued)

eMIOS200[n] Base Address	Description
0x00C	Disable Channel (EMIOSUCDIS)
0x00F	
0x01F	Reserved
0x020	Channel 0 to Channel 7
0x11F	
0x120	Channel 8 to Channel 15
0x21F	
0x220	Channel 16 to Channel 23
0x31F	
0x320	Channel 24 to Channel 31
0x41F	
0x420	Reserved
0xFFF	

42.3.1.1 Unified channel memory map

Addresses of Unified Channel registers are specified as offsets from the channel's base address, otherwise the eMIOS200 base address is used as reference.

The following table describes the Unified Channel memory map.

Table 42-3. Unified channel memory map

UC[n] Base Address	Description
UC[n]_BASE + 0x000	A register (EMIOSA[n])
UC[n]_BASE + 0x004	B register (EMIOSB[n])
UC[n]_BASE + 0x008	Counter register (EMIOSCNT[n])
UC[n]_BASE + 0x00C	Control register (EMIOSC[n])
UC[n]_BASE + 0x010	Status register (EMIOSS[n])
UC[n]_BASE + 0x014	Alternate A register (EMIOSALTA[n])
UC[n]_BASE + 0x018	Control 2 register (EMIOSC2[n])
UC[n]_BASE + 0x01C to UC[n]_BASE + 0x01F	Reserved

42.4 Control Memory Map and Registers

All control registers are 32 bits wide. This document illustrates the eMIOS200 with 24 Unified Channels and 24-bit wide data registers.

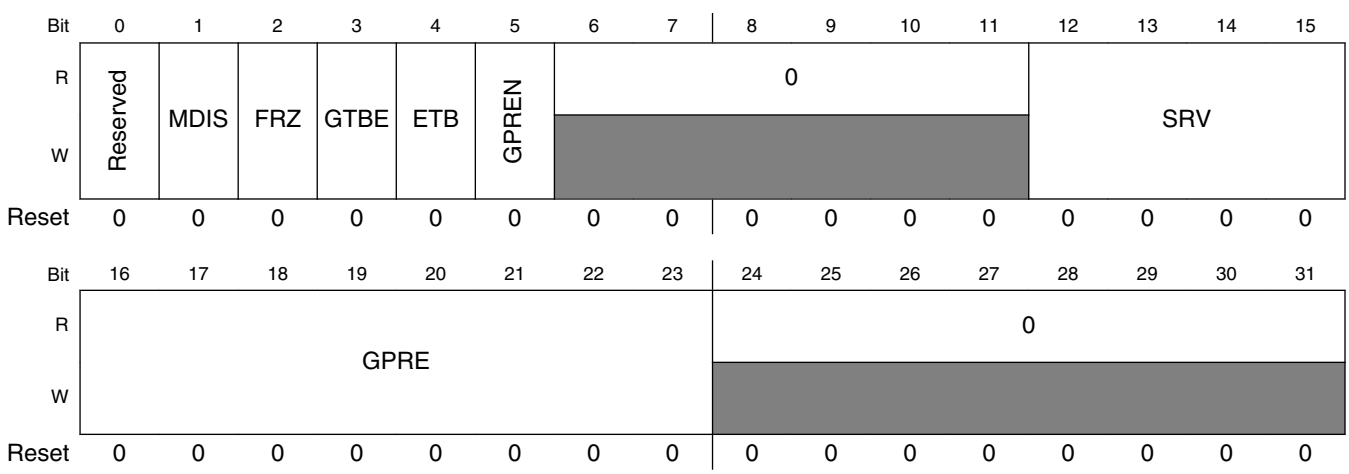
eMIOS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	eMIOS Module Configuration Register (eMIOS_MCR)	32	R/W	0000_0000h	42.4.1/1242
4	eMIOS Global FLAG Register (eMIOS_GFLAG)	32	R	0000_0000h	42.4.2/1244
8	eMIOS Output Update Disable register (eMIOS_OUDIS)	32	R/W	0000_0000h	42.4.3/1244
C	eMIOS Disable Channel register (eMIOS_UCDIS)	32	R/W	0000_0000h	42.4.4/1245

42.4.1 eMIOS Module Configuration Register (eMIOS_MCR)

The MCR register contains Global Control bits for the eMIOS200 block.

Address: 0h base + 0h offset = 0h



eMIOS_MCR field descriptions

Field	Description
0 Reserved	Reserved. This field is reserved. Only write this field with the reset value.

Table continues on the next page...

eMIOS_MCR field descriptions (continued)

Field	Description
1 MDIS	Module Disable bit. Puts the eMIOS200 into low power mode. The MDIS bit stops the clock of the block, except the access to registers EMIOSMCR, EMIOSOUDIS and EMIOSUCDIS. 0 Clock is running 1 Enter low power mode
2 FRZ	Freeze bit. This bit enables the eMIOS200 to freeze the registers of the Unified Channels when Debug Mode is requested at MCU level. Each Unified Channel should have FREN bit set in order to enter freeze state. While in Freeze state, the eMIOS200 continues to operate to allow the MCU access to the Unified Channels registers. The Unified Channel will remain frozen until the FRZ bit is written to zero or the MCU exits Debug mode or the Unified Channel FREN bit is cleared. 0 Exit freeze state 1 Stops Unified Channels operation when in Debug mode and the FREN bit is set in the EMIOSC[n] register
3 GTBE	Global Time Base Enable bit. The GTBE bit is used to export a Global Time Base Enable from the module and provide a method to start time bases of several blocks simultaneously. **NOTE:** The Global Time Base Enable input pin controls the internal counters. When asserted, Internal counters are enabled. When negated, Internal counters disabled. 0 Global Time Base Enable Out signal negated 1 Global Time Base Enable Out signal asserted
4 ETB	External Time Base bit The ETB bit selects the time base source that drives counter bus[A]. 0 Counter bus[A] assigned to Unified Channel. 1 Counter bus[A] assigned to STAC.
5 GPREN	Global Prescaler Enable bit. The GPREN bit enables the prescaler counter. 0 Prescaler disabled (no clock) and prescaler counter is cleared 1 Prescaler enabled
6–11 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
12–15 SRV	Server Time Slot bits. The SRV bits select the address of a specific STAC server to which the STAC is assigned.
16–23 GPRE	Global Prescaler bits. The GPRE[0:7] bits select the clock divider value for the global prescaler, as shown in the following table. GPRE values and corresponding divide ratio values are: 00000000 1

Table continues on the next page...

eMIOS_MCR field descriptions (continued)

Field	Description
	00000001 2 00000010 3 00000011 4 ------- - 11111110 255 11111111 256
24–31 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

42.4.2 eMIOS Global FLAG Register (eMIOS_GFLAG)

GFLAG is a read-only register that groups the FLAG bits from all channels. Each bit relates to one channel.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								F23_F0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eMIOS_GFLAG field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 F23_F0	Channel [n] mirrored FLAG bit. For Unified Channels these bits are mirrors of the FLAG bits in the EMIOSS[n] register.

42.4.3 eMIOS Output Update Disable register (eMIOS_OUDIS)

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								OU23_OU0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eMIOS_OUDIS field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 OU23_OU0	Channel [n] Output Update Disable bit. When running MC, MCB or an output mode, values are written to registers A2 and B2. OU[n] bits are used to disable transfers from registers A2 to A1 and B2 to B1. Each bit controls one channel. 0 Transfer enabled. Depending on the operation mode, transfer may occur immediately or in the next period. Unless stated otherwise, transfer occurs immediately. 1 Transfers disabled.

42.4.4 eMIOS Disable Channel register (eMIOS_UCDIS)

NOTE

See the Configuration information for the reset value of this register.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								CHDIS23_CHDIS0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eMIOS_UCDIS field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 CHDIS23_ CHDIS0	Enable Channel n bit. The CHDISn bit is used to disable each of the channels by stopping its respective clock. 0 Channel n enabled 1 Channel n disabled

42.5 Unified Channels Memory Map and Registers

The Unified Channel registers are as follows:

eMIOS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
120	eMIOS UC A register n (eMIOS_A8)	32	R/W	0000_0000h	42.5.1/1248
124	eMIOS UC B register n (eMIOS_B8)	32	R/W	0000_0000h	42.5.2/1249
128	eMIOS CNT register n (eMIOS_CNT8)	32	R/W	0000_0000h	42.5.3/1250
12C	eMIOS UC Control register n (eMIOS_C8)	32	R/W	0000_0000h	42.5.4/1251
130	eMIOS UC Status register n (eMIOS_S8)	32	R/W	See section	42.5.5/1257
134	eMIOS Alternate Address register n (eMIOS_ALTA8)	32	R/W	0000_0000h	42.5.6/1259
140	eMIOS UC A register n (eMIOS_A9)	32	R/W	0000_0000h	42.5.1/1248
144	eMIOS UC B register n (eMIOS_B9)	32	R/W	0000_0000h	42.5.2/1249
148	eMIOS CNT register n (eMIOS_CNT9)	32	R/W	0000_0000h	42.5.3/1250
14C	eMIOS UC Control register n (eMIOS_C9)	32	R/W	0000_0000h	42.5.4/1251
150	eMIOS UC Status register n (eMIOS_S9)	32	R/W	See section	42.5.5/1257
154	eMIOS Alternate Address register n (eMIOS_ALTA9)	32	R/W	0000_0000h	42.5.6/1259
160	eMIOS UC A register n (eMIOS_A10)	32	R/W	0000_0000h	42.5.1/1248
164	eMIOS UC B register n (eMIOS_B10)	32	R/W	0000_0000h	42.5.2/1249
168	eMIOS CNT register n (eMIOS_CNT10)	32	R/W	0000_0000h	42.5.3/1250
16C	eMIOS UC Control register n (eMIOS_C10)	32	R/W	0000_0000h	42.5.4/1251
170	eMIOS UC Status register n (eMIOS_S10)	32	R/W	See section	42.5.5/1257
174	eMIOS Alternate Address register n (eMIOS_ALTA10)	32	R/W	0000_0000h	42.5.6/1259
180	eMIOS UC A register n (eMIOS_A11)	32	R/W	0000_0000h	42.5.1/1248
184	eMIOS UC B register n (eMIOS_B11)	32	R/W	0000_0000h	42.5.2/1249
188	eMIOS CNT register n (eMIOS_CNT11)	32	R/W	0000_0000h	42.5.3/1250
18C	eMIOS UC Control register n (eMIOS_C11)	32	R/W	0000_0000h	42.5.4/1251
190	eMIOS UC Status register n (eMIOS_S11)	32	R/W	See section	42.5.5/1257
194	eMIOS Alternate Address register n (eMIOS_ALTA11)	32	R/W	0000_0000h	42.5.6/1259
1A0	eMIOS UC A register n (eMIOS_A12)	32	R/W	0000_0000h	42.5.1/1248
1A4	eMIOS UC B register n (eMIOS_B12)	32	R/W	0000_0000h	42.5.2/1249
1A8	eMIOS CNT register n (eMIOS_CNT12)	32	R/W	0000_0000h	42.5.3/1250
1AC	eMIOS UC Control register n (eMIOS_C12)	32	R/W	0000_0000h	42.5.4/1251
1B0	eMIOS UC Status register n (eMIOS_S12)	32	R/W	See section	42.5.5/1257
1B4	eMIOS Alternate Address register n (eMIOS_ALTA12)	32	R/W	0000_0000h	42.5.6/1259
1C0	eMIOS UC A register n (eMIOS_A13)	32	R/W	0000_0000h	42.5.1/1248
1C4	eMIOS UC B register n (eMIOS_B13)	32	R/W	0000_0000h	42.5.2/1249

Table continues on the next page...

eMIOS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1C8	eMIOS CNT register n (eMIOS_CNT13)	32	R/W	0000_0000h	42.5.3/1250
1CC	eMIOS UC Control register n (eMIOS_C13)	32	R/W	0000_0000h	42.5.4/1251
1D0	eMIOS UC Status register n (eMIOS_S13)	32	R/W	See section	42.5.5/1257
1D4	eMIOS Alternate Address register n (eMIOS_ALTA13)	32	R/W	0000_0000h	42.5.6/1259
1E0	eMIOS UC A register n (eMIOS_A14)	32	R/W	0000_0000h	42.5.1/1248
1E4	eMIOS UC B register n (eMIOS_B14)	32	R/W	0000_0000h	42.5.2/1249
1E8	eMIOS CNT register n (eMIOS_CNT14)	32	R/W	0000_0000h	42.5.3/1250
1EC	eMIOS UC Control register n (eMIOS_C14)	32	R/W	0000_0000h	42.5.4/1251
1F0	eMIOS UC Status register n (eMIOS_S14)	32	R/W	See section	42.5.5/1257
1F4	eMIOS Alternate Address register n (eMIOS_ALTA14)	32	R/W	0000_0000h	42.5.6/1259
200	eMIOS UC A register n (eMIOS_A15)	32	R/W	0000_0000h	42.5.1/1248
204	eMIOS UC B register n (eMIOS_B15)	32	R/W	0000_0000h	42.5.2/1249
208	eMIOS CNT register n (eMIOS_CNT15)	32	R/W	0000_0000h	42.5.3/1250
20C	eMIOS UC Control register n (eMIOS_C15)	32	R/W	0000_0000h	42.5.4/1251
210	eMIOS UC Status register n (eMIOS_S15)	32	R/W	See section	42.5.5/1257
214	eMIOS Alternate Address register n (eMIOS_ALTA15)	32	R/W	0000_0000h	42.5.6/1259
220	eMIOS UC A register n (eMIOS_A16)	32	R/W	0000_0000h	42.5.1/1248
224	eMIOS UC B register n (eMIOS_B16)	32	R/W	0000_0000h	42.5.2/1249
228	eMIOS CNT register n (eMIOS_CNT16)	32	R/W	0000_0000h	42.5.3/1250
22C	eMIOS UC Control register n (eMIOS_C16)	32	R/W	0000_0000h	42.5.4/1251
230	eMIOS UC Status register n (eMIOS_S16)	32	R/W	See section	42.5.5/1257
234	eMIOS Alternate Address register n (eMIOS_ALTA16)	32	R/W	0000_0000h	42.5.6/1259
240	eMIOS UC A register n (eMIOS_A17)	32	R/W	0000_0000h	42.5.1/1248
244	eMIOS UC B register n (eMIOS_B17)	32	R/W	0000_0000h	42.5.2/1249
248	eMIOS CNT register n (eMIOS_CNT17)	32	R/W	0000_0000h	42.5.3/1250
24C	eMIOS UC Control register n (eMIOS_C17)	32	R/W	0000_0000h	42.5.4/1251
250	eMIOS UC Status register n (eMIOS_S17)	32	R/W	See section	42.5.5/1257
254	eMIOS Alternate Address register n (eMIOS_ALTA17)	32	R/W	0000_0000h	42.5.6/1259
260	eMIOS UC A register n (eMIOS_A18)	32	R/W	0000_0000h	42.5.1/1248
264	eMIOS UC B register n (eMIOS_B18)	32	R/W	0000_0000h	42.5.2/1249
268	eMIOS CNT register n (eMIOS_CNT18)	32	R/W	0000_0000h	42.5.3/1250
26C	eMIOS UC Control register n (eMIOS_C18)	32	R/W	0000_0000h	42.5.4/1251
270	eMIOS UC Status register n (eMIOS_S18)	32	R/W	See section	42.5.5/1257
274	eMIOS Alternate Address register n (eMIOS_ALTA18)	32	R/W	0000_0000h	42.5.6/1259
280	eMIOS UC A register n (eMIOS_A19)	32	R/W	0000_0000h	42.5.1/1248
284	eMIOS UC B register n (eMIOS_B19)	32	R/W	0000_0000h	42.5.2/1249
288	eMIOS CNT register n (eMIOS_CNT19)	32	R/W	0000_0000h	42.5.3/1250
28C	eMIOS UC Control register n (eMIOS_C19)	32	R/W	0000_0000h	42.5.4/1251
290	eMIOS UC Status register n (eMIOS_S19)	32	R/W	See section	42.5.5/1257

Table continues on the next page...

eMIOS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
294	eMIOS Alternate Address register n (eMIOS_ALTA19)	32	R/W	0000_0000h	42.5.6/1259
2A0	eMIOS UC A register n (eMIOS_A20)	32	R/W	0000_0000h	42.5.1/1248
2A4	eMIOS UC B register n (eMIOS_B20)	32	R/W	0000_0000h	42.5.2/1249
2A8	eMIOS CNT register n (eMIOS_CNT20)	32	R/W	0000_0000h	42.5.3/1250
2AC	eMIOS UC Control register n (eMIOS_C20)	32	R/W	0000_0000h	42.5.4/1251
2B0	eMIOS UC Status register n (eMIOS_S20)	32	R/W	See section	42.5.5/1257
2B4	eMIOS Alternate Address register n (eMIOS_ALTA20)	32	R/W	0000_0000h	42.5.6/1259
2C0	eMIOS UC A register n (eMIOS_A21)	32	R/W	0000_0000h	42.5.1/1248
2C4	eMIOS UC B register n (eMIOS_B21)	32	R/W	0000_0000h	42.5.2/1249
2C8	eMIOS CNT register n (eMIOS_CNT21)	32	R/W	0000_0000h	42.5.3/1250
2CC	eMIOS UC Control register n (eMIOS_C21)	32	R/W	0000_0000h	42.5.4/1251
2D0	eMIOS UC Status register n (eMIOS_S21)	32	R/W	See section	42.5.5/1257
2D4	eMIOS Alternate Address register n (eMIOS_ALTA21)	32	R/W	0000_0000h	42.5.6/1259
2E0	eMIOS UC A register n (eMIOS_A22)	32	R/W	0000_0000h	42.5.1/1248
2E4	eMIOS UC B register n (eMIOS_B22)	32	R/W	0000_0000h	42.5.2/1249
2E8	eMIOS CNT register n (eMIOS_CNT22)	32	R/W	0000_0000h	42.5.3/1250
2EC	eMIOS UC Control register n (eMIOS_C22)	32	R/W	0000_0000h	42.5.4/1251
2F0	eMIOS UC Status register n (eMIOS_S22)	32	R/W	See section	42.5.5/1257
2F4	eMIOS Alternate Address register n (eMIOS_ALTA22)	32	R/W	0000_0000h	42.5.6/1259
300	eMIOS UC A register n (eMIOS_A23)	32	R/W	0000_0000h	42.5.1/1248
304	eMIOS UC B register n (eMIOS_B23)	32	R/W	0000_0000h	42.5.2/1249
308	eMIOS CNT register n (eMIOS_CNT23)	32	R/W	0000_0000h	42.5.3/1250
30C	eMIOS UC Control register n (eMIOS_C23)	32	R/W	0000_0000h	42.5.4/1251
310	eMIOS UC Status register n (eMIOS_S23)	32	R/W	See section	42.5.5/1257
314	eMIOS Alternate Address register n (eMIOS_ALTA23)	32	R/W	0000_0000h	42.5.6/1259

42.5.1 eMIOS UC A register n (eMIOS_A*n*)

Address: 0h base + 120h offset + (32d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																A															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eMIOS_A*n* field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved.

Table continues on the next page...

eMIOS_A*n* field descriptions (continued)

Field	Description
	This read-only field is reserved and always has the value 0.
8–31 A	Depending on the mode of operation, internal registers A1[0:23] or A2[0:23], used for matches and captures, can be assigned to address EMIOS_An. Both A1 and A2 are cleared by reset. The table in the EMIOS_Bn register description summarizes the EMIOS_An writing and reading accesses for all operation modes. Values not listed are reserved. For more information, see the “UC Modes of Operation” section.

42.5.2 eMIOS UC B register n (eMIOS_B*n*)

Depending on the mode of operation, internal registers B1[0:23] or B2[0:23] can be assigned to address EMIOS_Bn. Both B1 and B2 are cleared by reset. The table below summarizes the EMIOS_Bn writing and reading accesses for all operation modes. Values not listed are reserved. For more information see the “UC Modes of Operation” section.

Depending on the channel configuration, it may have an EMIOS_B register or not. EMIOS_B register is required for the following modes: DAOC, IPM, IPWM, OPWM, OPWMB, OPWFM, OPWMT, OPWFMB, OPWMC, OPWMCB, MC, MCB, PEA, PEC, WPTA.

The following table summarizes the EMIOS_An, EMIOS_Bn, and EMIOS_ALTAn writing and reading accesses for all operation modes. Values not listed are reserved. For more information, see the “UC Modes of Operation” section.

Table 42-13. EMIOS_An, EMIOS_Bn and EMIOS_ALTAn values assignment

Operation mode	Register access					
	Write	Read	Write	Read	Alt write	Alt read
GPIO	A1, A2	A1	B1, B2	B1	A2	A2
SAIC[1, 1]	–	A2	B2	B2	–	–
SAOC[1]	A2	A1	B2	B2	–	–
IPWM	–	A2	–	B1	–	–
IPM	–	A2	–	B1	–	–
DAOC	A2	A1	B2	B1	–	–
PEA	A1	A2	–	B1	–	–
PEC[1]	A1	A1	B1	B1	–	A2
QDEC[1]	A1	A1	B2	B2	–	–
WPTA	A1	A1	B1	B1	–	A2
MC[1]	A2	A1	B2	B2	–	–
OPWFM	A2	A1	B2	B1	–	–
OPWMC	A2	A1	B2	B1	–	–
OPWM	A2	A1	B2	B1	–	–

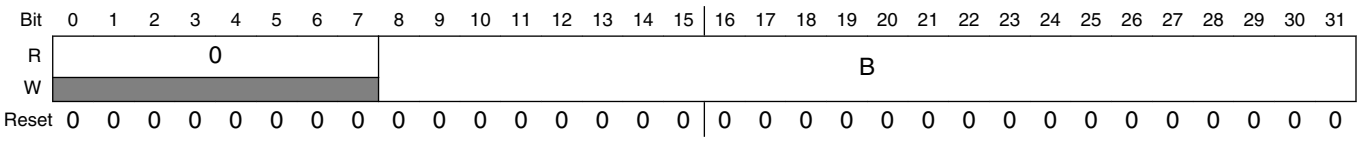
Table continues on the next page...

Table 42-13. EMIOS_An, EMIOS_Bn and EMIOS_ALTAn values assignment (continued)

Operation mode	Register access					
	Write	Read	Write	Read	Alt write	Alt read
OPWMT	A2	A1	B2	B1	A2	A2
MCB[1]	A2	A1	B2	B1	–	–
OPWFMB	A2	A1	B2	B1	–	–
OPWMCB	A2	A1	B2	B1	–	–
OPWMB	A2	A1	B2	B1	–	–

1. In these modes, the register EMIOS_Bn is not used, but B2 can be accessed.

Address: 0h base + 124h offset + (32d × i), where i=0d to 15d



eMIOS_B*n* field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 B	See text above register figure.

1. In these modes, the register EMIOS_Bn is not used, but B2 can be accessed.

42.5.3 eMIOS CNT register n (eMIOS_CNT*n*)

When GPIO mode is selected or the channel is frozen, the EMIOS_CNTn register is read/write. For all others modes, the EMIOS_CNTn is a read-only register. When entering some operation modes, this register is automatically cleared (refer to the “UC Modes of Operation” section for details).

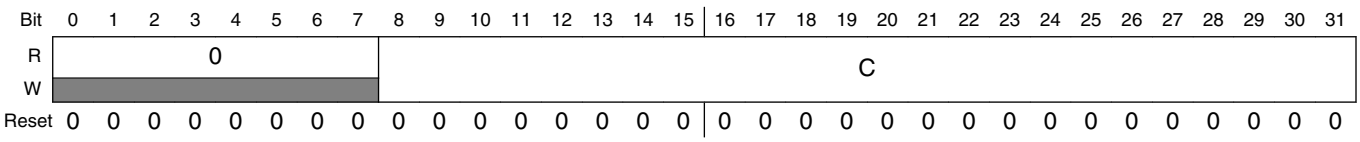
EMIOS_CNTn register is required for the following modes:

- OPWFM
- OPWFMB
- OPWMC
- OPWMCB
- PEA
- PEC
- WPTA

- QDEC
- MC
- MCB

For other modes, the internal counter is not used. Therefore, depending on the modes available in a channel, the internal counter may be absent.

Address: 0h base + 128h offset + (32d × i), where i=0d to 15d



eMIOS_CNT*n* field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 C	The C field contains the value of the internal counter.

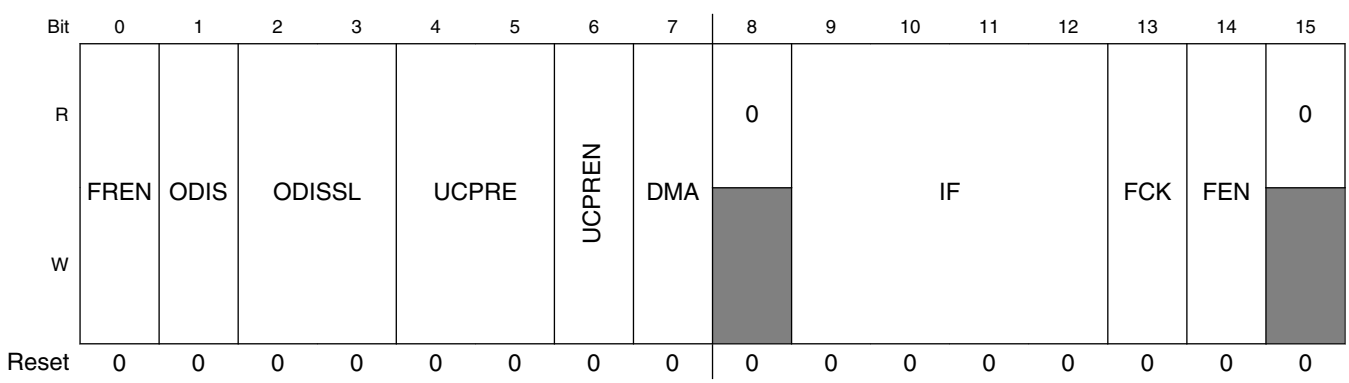
42.5.4 eMIOS UC Control register n (eMIOS_C*n*)

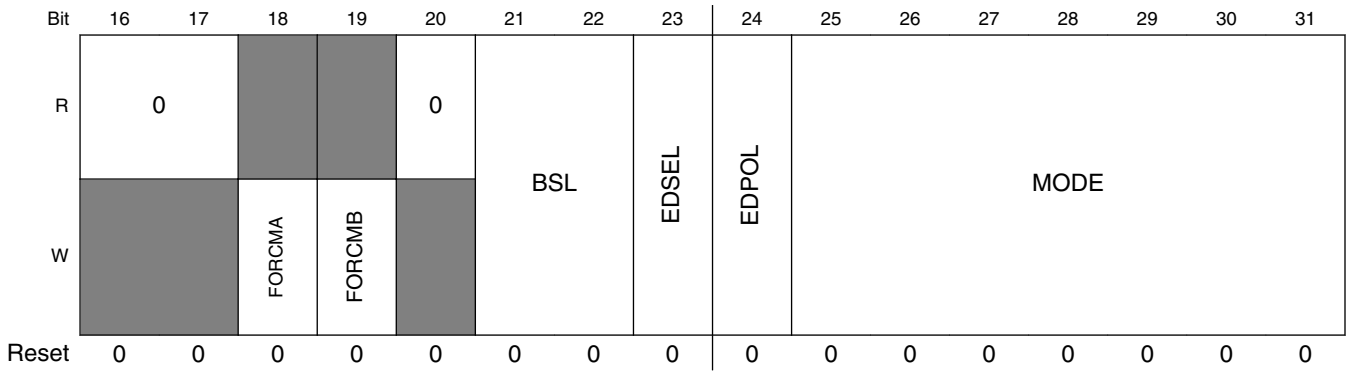
The UC Control register gathers several read/write control bits affecting the UC operation conditions.

NOTE

This register does not include a FORCMB bit for Channel Types A and C.

Address: 0h base + 12Ch offset + (32d × i), where i=0d to 15d





eMIOS_C*n* field descriptions

Field	Description
0 FREN	Freeze Enable bit. The FREN bit, if set and validated by FRZ bit in EMIOS_MCR register allows the channel to enter freeze state, freezing all registers values when in debug mode and allowing the MCU to perform debug functions. 0 Normal operation 1 Freeze UC registers values
1 ODIS	Output Disable bit. The ODIS bit allows disabling the output pin when running any of the output modes with the exception of GPIO mode. 0 The output pin operates normally 1 If the selected Output Disable Input signal is asserted, the output pin goes to EDPOL for OPWFMB and OPWMB modes and to the complement of EDPOL for other output modes, but the Unified Channel continues to operate normally, i.e., it continues to produce FLAG and matches. When the selected Output Disable Input signal is negated, the output pin operates normally.
2–3 ODISSL	Output Disable select bits. The ODISSL[0:1] bits select one of the four output disable input signals, as shown in the following table. **Table 42-23. UC ODISSL selection** ODISSL[0:1] → input signal 00 → Output Disable Input 0 01 → Output Disable Input 1 10 → Output Disable Input 2 11 → Output Disable Input 3
4–5 UCPRE	Prescaler bits. The UCPRE[0:1] bits select the clock divider value for the internal prescaler of Unified Channel, as shown in the following table. **Table 42-22. UC Internal Prescaler clock divider** UCPRE[0:1] → Divide ratio 00 → 1

Table continues on the next page...

eMIOS_C*n* field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
Table 42-22. UC Internal Prescaler clock divider (continued)
<table>
<tr><th>UCPRE[0:1]</th><th>Divide ratio</th></tr>
<tr><td>01</td><td>2</td></tr>
<tr><td>10</td><td>3</td></tr>
<tr><td>11</td><td>4</td></tr>
</table>
NOTE: The UCPRE bits are mirrored in the two LSBs of the UCEXTPRE field in register EMIOS_Cn. Any write operations to UCPRE bits will affect the respective bits at EMIOS_C2n register.
</td></tr>
<tr><td>6
UCPREN</td><td>Prescaler Enable bit.

The UCPREN bit enables the prescaler counter.

0 Prescaler disabled (no clock)
1 Prescaler enabled</td></tr>
<tr><td>7
DMA</td><td>Direct Memory Access bit.

The DMA bit selects if FLAG generation will be used as an interrupt or as a DMA request.

0 Flag/overrun assigned to Interrupt request.
1 Flag/overrun assigned to DMA request.</td></tr>
<tr><td>8
Reserved</td><td>Reserved.

This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>9–12
IF</td><td>Input Filter bits.

The IF[0:3] bits control the programmable input filter, selecting the minimum input pulse width that can pass through the filter, as shown in the following table. For output modes, these bits have no meaning.
Table 42-21. UC Input Filter bits
<table>
<tr><th>IF[0:3] [1]</th><th>Minimum input Pulse width [filter clock periods]</th></tr>
<tr><td>0000</td><td>bypassed[2]</td></tr>
<tr><td>0001</td><td>02</td></tr>
<tr><td>0010</td><td>04</td></tr>
<tr><td>0100</td><td>08</td></tr>
<tr><td>1000</td><td>16</td></tr>
<tr><td>All others</td><td>Reserved</td></tr>
</table>
1. Filter latency (the difference in time between the input and the response) is 3 clock edges.
2. The input signal is synchronized before arriving to the digital filter.</td></tr>
<tr><td>13
FCK</td><td>Filter Clock select bit.

The FCK bit selects the clock source for the programmable input filter.

0 Prescaled clock
1 eMIOS module clock</td></tr>
</table>

Table continues on the next page...

eMIOS_C*n* field descriptions (continued)

Field	Description
14 FEN	FLAG Enable bit. The FEN bit allows the Unified Channel FLAG bit to generate an interrupt signal or a DMA request signal (the type of signal to be generated is defined by the DMA bit). 0 Disable (FLAG does not generate an interrupt request) 1 Enable (FLAG generates an interrupt request)
15–17 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
18 FORCMA	Force Match A bit. For output modes, the FORCMA bit is equivalent to a successful comparison on comparator A (except that the FLAG bit is not set). This bit is cleared by reset and is always read as zero. This bit is valid for every output operation mode which uses comparator A, otherwise it has no effect. For input modes, the FORCMA bit is not used and writing to it has no effect. 0 Has no effect. 1 Force a match at comparator A
19 FORCMB	Force Match B bit. For output modes, the FORCMB bit is equivalent to a successful comparison on comparator B (except that the FLAG bit is not set). This bit is cleared by reset and is always read as zero. This bit is valid for every output operation mode which uses comparator B, otherwise it has no effect. 0 Has not effect 1 Force a match at comparator B
20 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
21–22 BSL	Bus Select bits. The BSL[0:1] bits are used to select either one of the counter buses or the internal counter to be used by the Unified Channel , as shown in the following tables.

Table 42-20. UC BSL bits

BSL[0:1]	Selected bus[1]
00	All channels: counter bus[A]
01	Channels 0 to 7: counter bus[B]
	Channels 8 to 15: counter bus[C]
	Channels 16 to 23: counter bus[D]
	Channels 24 to 31: counter bus[E]
10	Reserved
11	All channels: internal counter

1. Not all devices support all counter buses. See the Device Configuration chapter for details.

23 EDSEL	Edge Selection bit.

Table continues on the next page...

eMIOS_C*n* field descriptions (continued)

Field	Description
	For input modes, the EDSEL bit selects whether the internal counter is triggered by both edges of a pulse or just by a single edge as defined by the EDPOL bit . When not shown in the mode of operation description, this bit has no effect. 0 – Single edge triggering defined by the EDPOL bit 1 – Both edges triggering{mts_ipi_slv_saic} For GPIO in mode, the EDSEL bit selects if a FLAG can be generated. 0 – A FLAG is generated as defined by the EDPOL bit 1 – No FLAG is generated For SAOC mode, the EDSEL bit selects the behavior of the output flip-flop at each match. 0 – The EDPOL value is transferred to the output flip-flop 1 – The output flip-flop is toggled
24 EDPOL	Edge Polarity bit. For input modes (except QDEC mode), the EDPOL bit asserts which edge triggers either the internal counter or an input capture or a FLAG . When not shown in the mode of operation description, this bit has no effect. 0 – Trigger on a falling edge 1 – Trigger on a rising edge **NOTE:** UC[n-1] EDPOL bit selects which edge clocks the internal counter of UC[n] **NOTE:** In order to operate properly, EDPOL bit must contain the same value in UC[n] and UC[n-1] For output modes, the EDPOL bit is used to select the logic level on the output pin . 0 – A match on comparator A clears the output flip-flop, while a match on comparator B sets it 1 – A match on comparator A sets the output flip-flop, while a match on comparator B clears it
25–31 MODE	Mode selection bits. The MODE bits select the mode of operation of the Unified Channel, as shown in the following table. Values not listed are reserved. **NOTE:** If a reserved value is written to the mode bit field, the results are unpredictable.

Table 42-19. UC MODE bits

MODE[0:6]	Mode of operation[1]
0000000	General purpose Input/Output mode (input)
0000001	General purpose Input/Output mode (output)
0000010	Single Action Input Capture
0000011	Single Action Output Compare
0000100	Input Pulse Width Measurement
0000101	Input Period Measurement
0000110	Double Action Output compare (with FLAG set on B match)
0000111	Double Action Output compare (with FLAG set on both match)
0001000	Pulse/Edge Accumulation (continuous)
0001001	Pulse/Edge Accumulation (single shot)

Table continues on the next page...

eMIOS_C*n* field descriptions (continued)

Field	Description

Table 42-19. UC MODE bits (continued)

MODE[0:6]	Mode of operation[1]
0001010	Pulse/Edge Counting (continuous)
0001011	Pulse/Edge Counting (single shot)
0001100	Quadrature Decode (for count and direction encoders type)
0001101	Quadrature Decode (for phase_A and phase_B encoders type)
0001110	Windowed Programmable Time Accumulation
0001111	Reserved
001000b[2, 2]	Modulus Counter (Up counter with clear on match start)
001001b	Modulus Counter (Up counter with clear on match end)
00101bb[2]	Modulus Counter (Up/Down counter)
00110b0[2]	Output Pulse Width and Frequency Modulation (immediate update)
00110b1[2]	Output Pulse Width and Frequency Modulation (next period update)
00111b0[2]	Center Aligned Output Pulse Width Modulation (with trail edge dead-time)
00111b1[2]	Center Aligned Output Pulse Width Modulation (with lead edge dead-time)
01000b0[2]	Output Pulse Width Modulation (immediate update)
01000b1[2]	Output Pulse Width Modulation (next period update)
0100100 through 0100101	Reserved
0100110	Output Pulse Width Modulation with Trigger
0100111 through 1001111	Reserved
101000b[2]	Modulus Counter Buffered (Up counter)
101001b[2]	Reserved
10101bb[2]	Modulus Counter Buffered (Up/Down counter)
10110b0[2]	Output Pulse Width and Frequency Modulation Buffered
10110b1[2]	Reserved
10111b0[2]	Center Aligned Output Pulse Width Modulation Buffered (with trail edge dead-time)
10111b1[2]	Center Aligned Output Pulse Width Modulation Buffered (with lead edge dead-time)
11000b0[2]	Output Pulse Width Modulation Buffered
1100001 through 1111111	Reserved

1. Not all modes of operation are supported. See the configuration information for details.
2. b = adjust parameters for the mode of operation. Refer to “UC Modes of Operation” section for details.

1. Filter latency (the difference in time between the input and the response) is 3 clock edges.
2. The input signal is synchronized before arriving to the digital filter.
3. Not all devices support all counter buses. See the Device Configuration chapter for details.

4. Not all modes of operation are supported. See the configuration information for details.
5. b = adjust parameters for the mode of operation. Refer to “UC Modes of Operation” section for details.

42.5.5 eMIOS UC Status register n (eMIOS_S*n*)

The UC Status register EMIOS_Sn gathers bits reflecting the status of the UC input/ output signals and the overflow condition of the internal counter.

Address: 0h base + 130h offset + (32d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OVR	0														
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OVFL	0												UCIN	UCOUT	FLAG
W	w1c															w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	*	0	0

* Notes:
- UCIN field: Reset value is undefined.

eMIOS_S*n* field descriptions

Field	Description
0 OVR	Overrun bit. The OVR bit indicates that FLAG generation occurred when the FLAG bit was already set . The OVR bit can be cleared either by clearing the FLAG bit or by writing a 1 to the OVR bit. 0 Overrun has not occurred 1 Overrun has occurred

Table continues on the next page...

eMIOS_S*n* field descriptions (continued)

Field	Description
1–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16 OVFL	Overflow bit. The OVFL bit indicates that an overflow has occurred in the internal counter. The OVFL bit must be cleared by software writing a 1 to it. 0 No overflow 1 An overflow had occurred
17–28 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
29 UCIN	Unified Channel Input pin bit. The UCIN bit reflects the input pin state after being filtered and synchronized.
30 UCOUT	Unified Channel Output pin bit. The UCOUT bit reflects the output pin state .
31 FLAG	FLAG bit. The FLAG bit is set when an input capture or a match event in the comparators occurred. The FLAG bit must be cleared by writing a 1 to it. 0 FLAG cleared 1 FLAG set event has occurred

42.5.6 eMIOS Alternate Address register n (eMIOS_ALTA*n*)

The eMIOS_ALTAn register provides an alternate address to access A2 channel registers in restricted modes (GPIO, PEC, WPTA, OPWMT)only. In modes where the function of eMIOS_ALTAn register is not specified, any write access is ignored and any read access returns unspecified data. If eMIOS_An register is used along with eMIOS_ALTAn, both A1 and A2 registers can be accessed in these modes. The table in the eMIOS_Bn register's description summarizes the EMIOS_ALTAn writing and reading accesses for all EMIOS_ALTAn operation modes. See the “General purpose Input/Output mode (GPIO) Mode” section, the “Pulse/Edge Counting (PEC) Mode” section, the “Windowed Programmable Time Accumulation (WPTA) Mode” section, and the “Output Pulse Width Modulation with Trigger (OPWMT) Mode” section for a more detailed description of the use of eMIOS_ALTAn register.

Address: 0h base + 134h offset + (32d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								ALTA																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eMIOS_ALTA*n* field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 ALTA	See text above register figure.

42.6 Functional Description

The eMIOS200 provides independent channels (Unified) that can be configured and accessed by a host MCU. Up to five time bases can be shared by the channels.

The eMIOS200 module is based on a multi-bus timer architecture in which several timer channels are used to drive counter buses that are shared among the channels. Up to five timebases can be shared by the channels through five counter buses (counter bus A or F, B, C, D and E) and each channel can generate its own timebase. There are five counter buses in the module: one global counter bus, shared by all channels and selectable among

time bases A or F, and four local counter buses (B, C, D, and E), each one dedicated to a slice of 8 channels. Counter bus A is referred to as the global counter buse. Counter buses B, C, D, and E are the local counter buses.

The eMIOS200 counter buses are driven by channels in specific locations. Global counter bus A is driven by the channel in channel slot [23].. Counter buses B, C, D, and E are driven by channels in slots 0, 8, 16, and 24, respectively. Counter bus A or F drives all channels. Counter bus B drives channels in slots from 0 through 7. Counter bus C drives channels in slots from 8 through 15. Counter bus D drives channels in slots from 16 through 23. Counter bus E drives channels from 24 through 31.

The eMIOS200 block is reset asynchronously. All registers are cleared on reset.

The following figure shows an eMIOS200 block configured with a mix of Unified Channels. In this configuration example, each one of the channels can be driven by the respective local bus—B for the [7:0] slice and D for the [23:16] slice—or by the global counter bus A. Note that there are Unified Channels with no external pin associated. These channels are used to trigger internal SoC signals such as interrupt requests or DMA transfer requests. Since the channel flags are also connected to the block top-level pins, they can be used as trigger signals for other blocks at chip level.

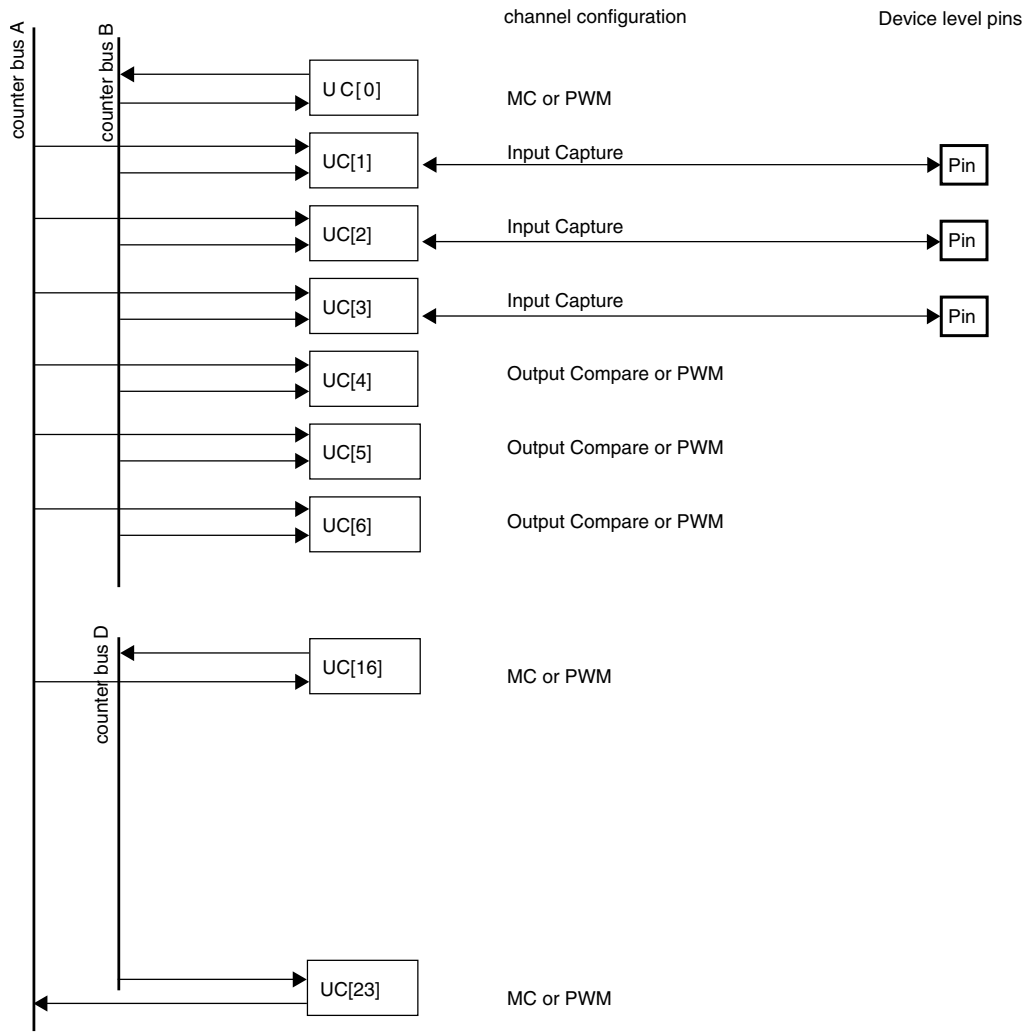


Figure 42-108. eMIOS200 example configuration using Unified Channels

42.6.1 Unified Channel (UC)

Figure 42-109 shows the Unified Channel block diagram. Each Unified Channel consists of the following:

- A counter bus selector, which selects the timebase to be used by the channel for all timing functions
- A programmable clock prescaler
- Two double-buffered data registers, A and B, that allow up to two input capture and/or output compare events to occur before software intervention is needed
- Two comparators (equal only), A and B, that compare the selected counter bus with the value in the data registers

- An internal counter, which can be used as a local time base or to count input events
- Programmable input filter, which ensures that only valid pin transitions are received by channel
- Programmable input edge detector, which detects the rising, falling, or either edge
- An output flip-flop, which holds the logic level to be applied to the output pin
- eMIOS200 Status and Control register
- An Output Disable Input selector, which selects the Output Disable Input signal that will be used as output disable

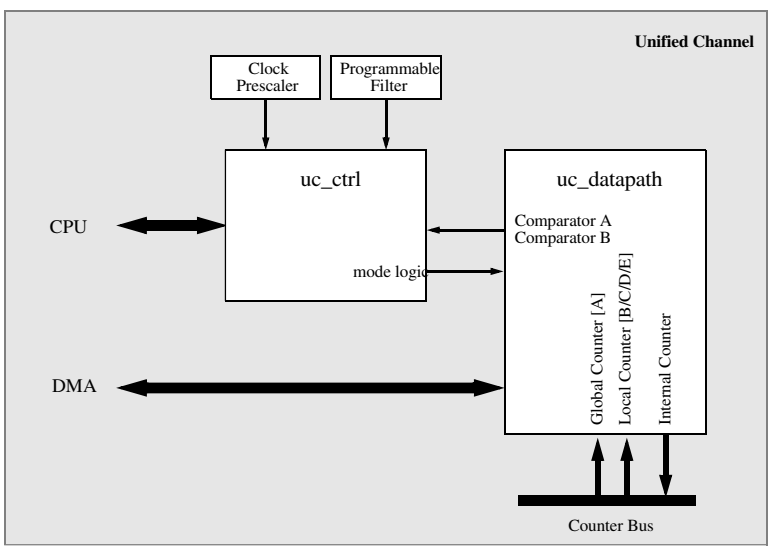


Figure 42-109. Unified Channel block diagram

Figure 42-110 shows both the Unified Channel control and datapath block diagrams. The control sub-block is responsible for the generation of signals to control the multiplexers in the datapath sub-block. Each mode is implemented by a dedicated logic independent from others modes, thus allowing to optimize the logic by disabling the mode and therefore its associated logic. Targeting the logic optimization, a set of registers is shared by the modes thus providing sequential events to be stored.

The datapath block provides the channel A and B registers, the internal timebase and comparators. Multiplexers select the input of comparators and data for the registers inputs, thus configuring the datapath to implement the channel modes. The outputs of the A and B comparators are connected to the uc_ctrl control block.

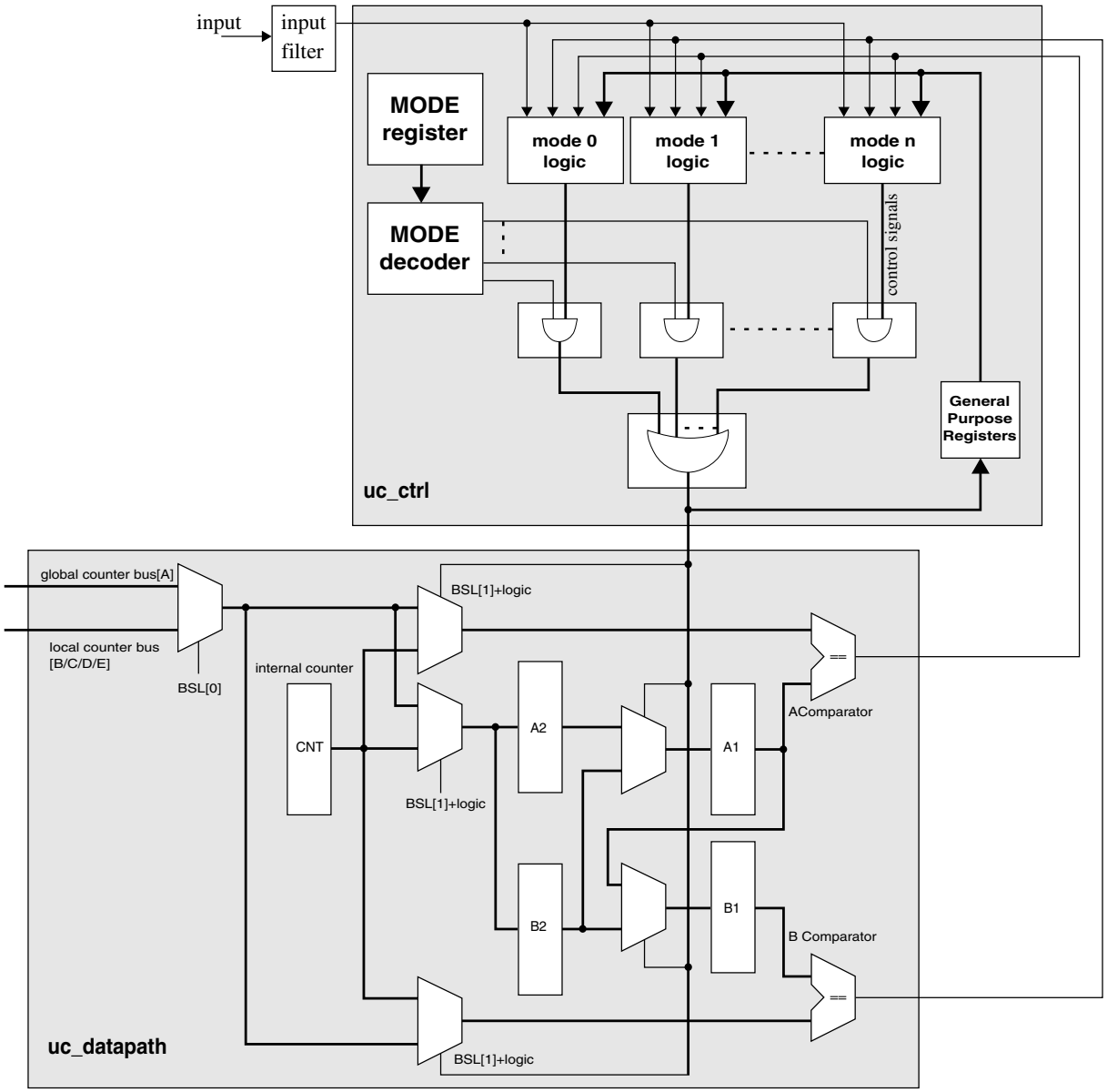


Figure 42-110. Unified Channel control and datapath block diagrams

42.6.1.1 Unified Channel modes of operation

The mode of operation of the Unified Channel is determined by the mode select bits MODE[0:6] in the EMIOS_C_n register.

When entering an output mode (except for GPIO mode), the output flip-flop is set to the disabled state according to EMIOSCn[ODIS].

As the internal counter, EMIOSCNTn, runs in all modes (except for GPIO mode) and can be used as a timebase if the resource is not used in the current mode.

In order to provide smooth waveform generation even when A and B registers are immediately changed, it is available with the MCB, OPWFMB, OPWMB and OPWFMB modes. In these modes, A and B registers are double buffered. They are presented in separate sections because there are several basic differences between these and each corresponding nonbuffered mode.

42.6.1.1.1 General-Purpose Input/Output (GPIO) mode

In GPIO mode, all input capture and output compare functions of the Unified Channel are disabled, and the internal counter (EMIOSCNTn register) is cleared and disabled. All control bits remain accessible. In order to prepare the UC for a new operation mode, writing to registers EMIOSAn or EMIOSBn stores the same value in registers A1/A2 or B1/B2, respectively. Writing to the EMIOSALTAn register stores a value only in register A2.

MODE[6] bit selects between input (MODE[6] = 0) and output (MODE[6] = 1) modes.

When changing MODE[0:6], the application software must first enter GPIO mode to reset the Unified Channel's internal functions properly. Failure to do this could lead to invalid and unexpected output compare or input capture results or the FLAGs being set incorrectly.

In GPIO input mode (MODE[0:6] = 0000000), FLAG generation is determined according to EDPOL and EDSEL bits and the input pin status can be determined by reading the UCIN bit.

In GPIO output mode (MODE[0:6] = 0000001), the Unified Channel is used as a single output port pin and the value of the EDPOL bit is permanently transferred to the output flip-flop.

42.6.1.1.2 Single Action Input Capture (SAIC) Mode

In SAIC mode (MODE[0:6]=0000010), when a triggering event occurs on the input pin, the value on the selected time base is captured into register A2. The FLAG bit is set along with the capture event to indicate that an input capture has occurred. Register EMIOSA[n] returns the value of register A2. As soon as the SAIC mode is entered coming out from GPIO mode the channel is ready to capture events. The events are captured as soon as they occur thus reading register A always returns the value of the latest captured event. Subsequent captures are enabled with no need of further reads from EMIOSA[n] register. The FLAG is set at any time a new event is captured.

The input capture is triggered by a rising, falling or either edges in the input pin, as configured by EDPOL and EDSEL bits in EMIOSC[n] register.

Figure 42-111 and Figure 42-112 show how the Unified Channel can be used for input capture.

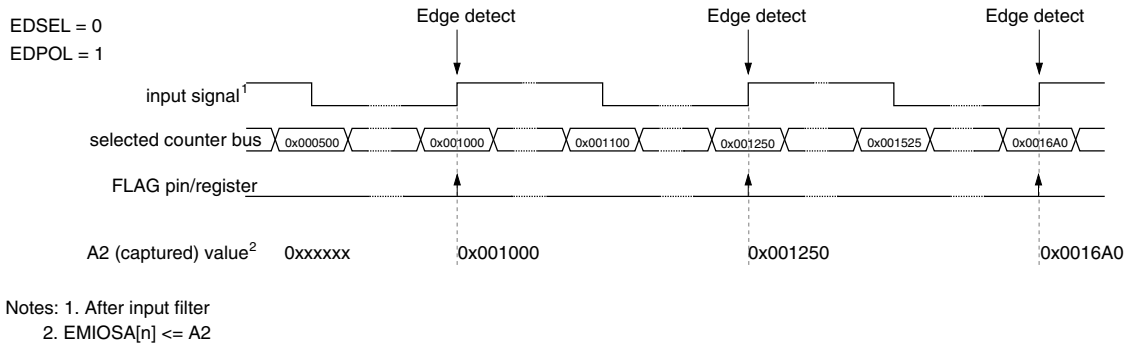


Figure 42-111. Single Action Input Capture with rising edge triggering example

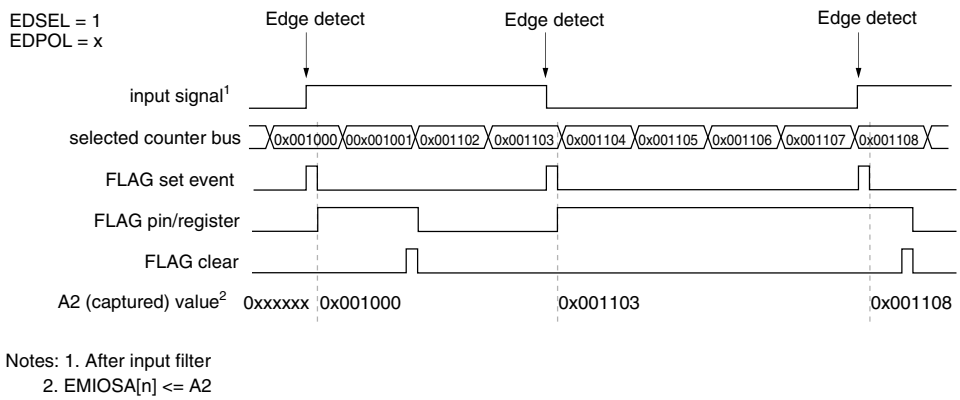


Figure 42-112. Single Action Input Capture with both edges triggering example

42.6.1.1.3 Single Action Output Compare (SAOC) Mode

In SAOC mode (MODE[0:6]=0000011) a match value is loaded in register A2 and then immediately transferred to register A1 to be compared with the selected time base. When a match occurs, the EDSEL bit selects whether the output flip-flop is toggled or the value in EDPOL is transferred to it. Along with the match the FLAG bit is set to indicate that the output compare match has occurred. Writing to register EMIOSA[n] stores the value in register A2 and reading to register EMIOSA[n] returns the value of register A1.

An output compare match can be simulated in software by setting the FORCMA bit in EMIOSC[n] register. In this case, the FLAG bit is not set.

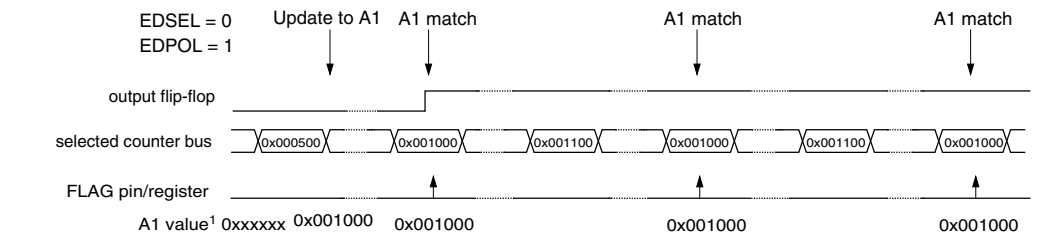
When SAOC mode is entered coming out from GPIO mode, the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC[n] register.

Counter bus can be either internal or external and is selected through BSL[0:1] bits.

Figure 42-113 and Figure 42-114 show how the Unified Channel can be used to perform a single output compare with EDPOL value being transferred to the output flip-flop and toggling the output flip-flop at each match, respectively. Note that once in SAOC mode the matches are enabled thus the desired match value on register A1 must be written before the mode is entered. A1 register can be updated at any time thus modifying the match value which will reflect in the output signal generated by the channel. Subsequent matches are enabled with no need of further writes to EMIOSA[n] register. The FLAG is set at the same time a match occurs (see Figure 42-115).

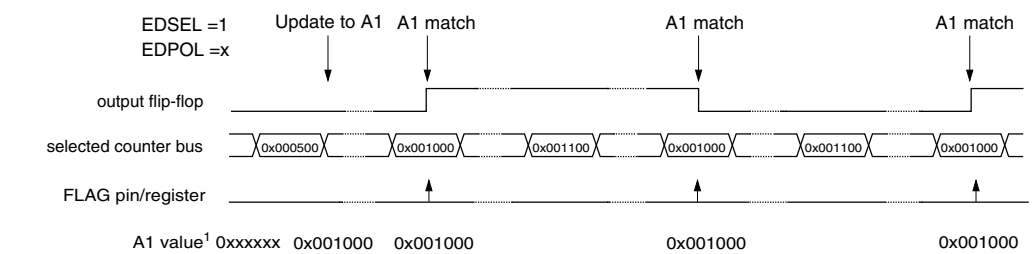
Note

The channel internal counter in SAOC mode is free-running. It starts counting as soon as the SAOC mode is entered.



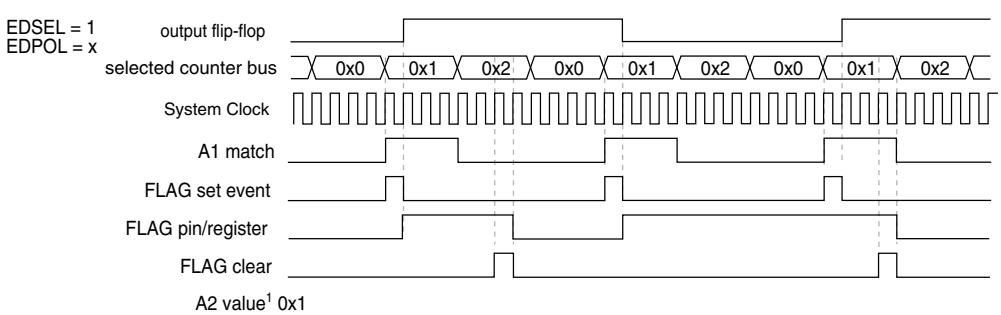
Notes: 1. EMIOSA[n] = A2
A2 = A1 according to OU[n] bit

Figure 42-113. SAOC example with EDPOL value being transferred to the output flip-flop



Notes: 1. EMIOSA[n] = A2
A2 = A1 according to OU[n] bit

Figure 42-114. SAOC example toggling the output flip-flop



Note: 1. EMIOSA[n] <= A2

Figure 42-115. SAOC example with flag behavior

42.6.1.1.4 Input Pulse Width Measurement (IPWM) Mode

The IPWM mode (MODE[0:6]=0000100) allows the measurement of the width of a positive or negative pulse by capturing the leading edge on register B1 and the trailing edge on register A2. Successive captures are done on consecutive edges of opposite polarity. The leading edge sensitivity (i.e., pulse polarity) is selected by EDPOL bit in the EMIOSC[n] register. Registers EMIOSA[n] and EMIOSB[n] return the values in register A2 and B1, respectively.

The capture function of register A2 remains disabled until the first leading edge triggers the first input capture on register B2. When this leading edge is detected, the count value of the selected time base is latched into register B2; the FLAG bit is not set. When the trailing edge is detected, the count value of the selected time base is latched into register A2 and, at the same time, the FLAG bit is set and the content of register B2 is transferred to register B1 and to register A1.

If subsequent input capture events occur while the corresponding FLAG bit is set, registers A2, B1 and A1 will be updated with the latest captured values and the FLAG will remain set. Registers EMIOSA[n] and EMIOSB[n] return the value in registers A2 and B1, respectively.

In order to guarantee coherent access, reading EMIOSA[n] forces B1 be updated with the content of register A1. At the same time transfers between B2 and B1 are disabled until the next read of EMIOSB[n] register. Reading EMIOSB[n] register forces B1 be updated with A1 register content and re-enables transfers from B2 to B1, to take effect at the next trailing edge capture. Transfers from B2 to A1 are not blocked at any time.

The input pulse width is calculated by subtracting the value in B1 from A2.

The following figure shows how the Unified Channel can be used for input pulse width measurement.

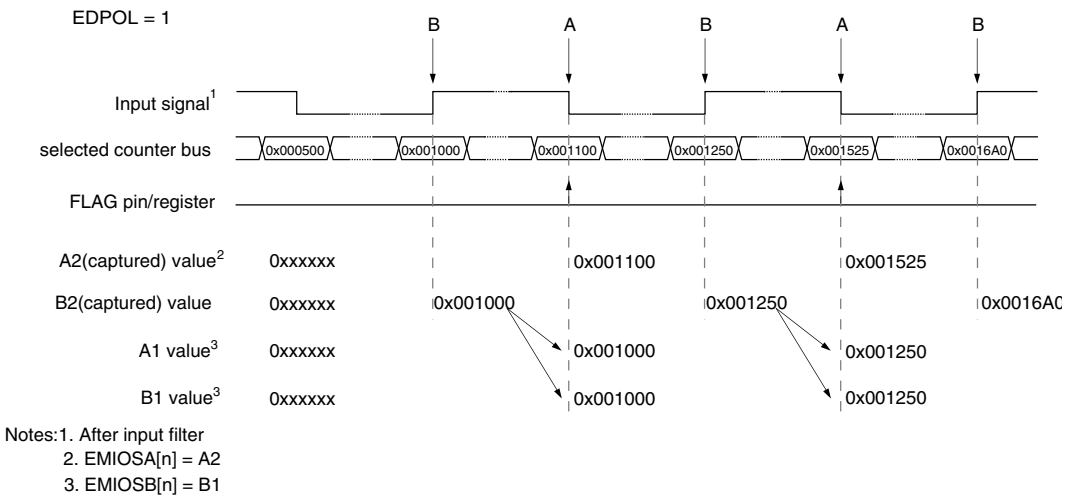


Figure 42-116. Input Pulse Width Measurement example

The following figure shows the A1 and B1 updates when EMIOSA[n] and EMIOSB[n] register reads occur. Note that A1 register has always coherent data related to A2 register. Note also that when EMIOSA[n] read is performed B1 register is loaded with A1 register content. This guarantee that the data in register B1 has always the coherent data related to the last EMIOSA[n] read. The B1 register updates remains locked until EMIOSB[n] read occurs. If EMIOSA[n] read is performed B1 is updated with A1 register content even if B1 update is locked by a previous EMIOSA[n] read operation.

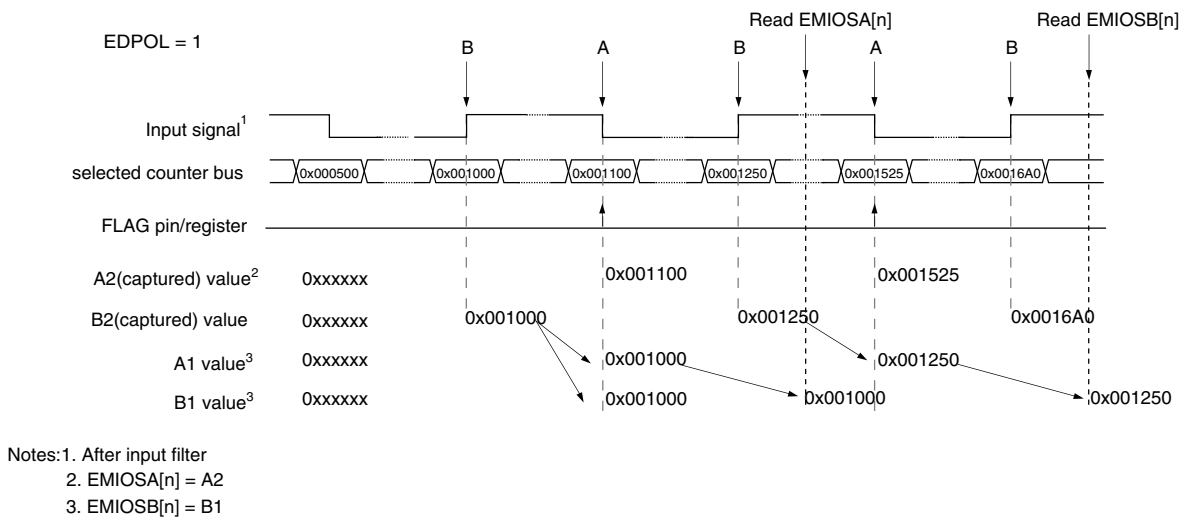


Figure 42-117. B1 and A1 updates at EMIOSA[n] and EMIOSB[n] reads

Reading EMIOS_A_n followed by EMIOS_B_n always provides coherent data. If coherent data is not required for any reason, the sequence of reads should be inverted, therefore EMIOSB[n] should be read prior to EMIOSA[n] register. Note that even in this case B1 register updates will be blocked after EMIOSA[n] read, thus a second EMIOSB[n] is required in order to release B1 register updates.

42.6.1.1.5 Input Period Measurement (IPM) Mode

The IPM mode (MODE[0:6]=0000101) allows the measurement of the period of an input signal by capturing two consecutive rising edges or two consecutive falling edges. Successive input captures are done on consecutive edges of the same polarity. The edge polarity is defined by the EDPOL bit in the EMIOSC[n] register.

Note

The input signal must have at least four system clock cycles period in order to be properly captured by the synchronization logic at the channel input even if the input filter is in by-pass mode.

When the first edge of selected polarity is detected, the selected time base is latched into the registers A2 and B2, and the data previously held in register B2 is transferred to register B1. On this first capture the FLAG line is not set, and the values in registers B1 is meaningless. On the second and subsequent captures, the FLAG line is set and data in register B2 is transferred to register B1.

When the second edge of the same polarity is detected, the counter bus value is latched into registers A2 and B2, the data previously held in register B2 is transferred to data register B1 and to register A1. The FLAG bit is set to indicate the start and end points of a complete period have been captured. This sequence of events is repeated for each subsequent capture. Registers EMIOSA[n] and EMIOSB[n] return the values in register A2 and B1, respectively.

In order to allow coherent data, reading EMIOSA[n] forces A1 content be transferred to B1 register and disables transfers between B2 and B1. These transfers are disabled until the next read of the EMIOSB[n] register. Reading EMIOSB[n] register forces A1 content to be transferred to B1 and re-enables transfers from B2 to B1, to take effect at the next edge capture.

The input pulse period is calculated by subtracting the value in B1 from A2.

The following figure shows how the Unified Channel can be used for input period measurement.

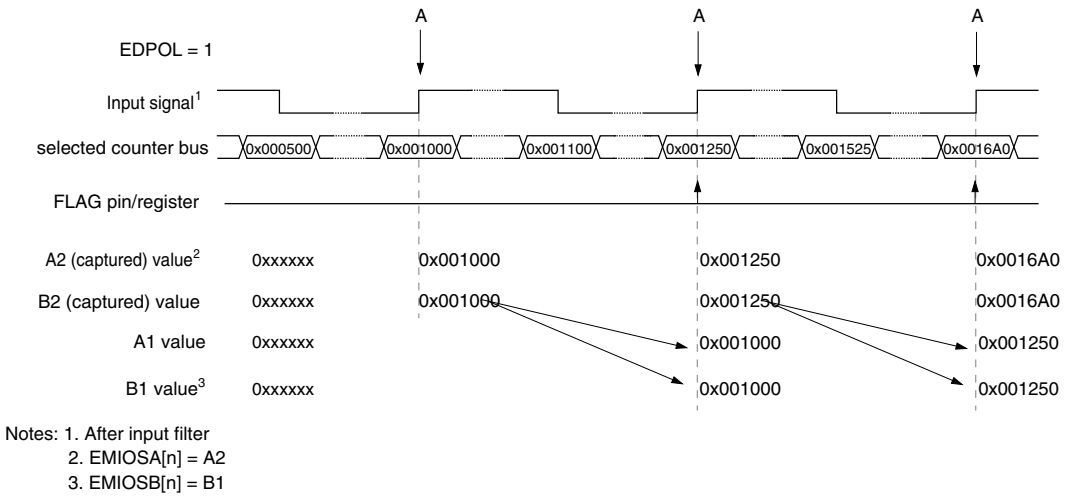


Figure 42-118. Input Period Measurement example

Figure 42-119 describes the A1 and B1 register updates when EMIOSA[n] and EMIOSB[n] read operations are performed. When EMIOSA[n] read occurs the content of A1 is transferred to B1 thus providing coherent data in A2 and B1 registers. Transfers from B2 to B1 are then blocked until EMIOSB[n] is read. After EMIOSB[n] is read, register A1 content is transferred to register B1 and the transfers from B2 to B1 are re-enabled to occur at the transfer edges, which is the leading edge in the following example figure.

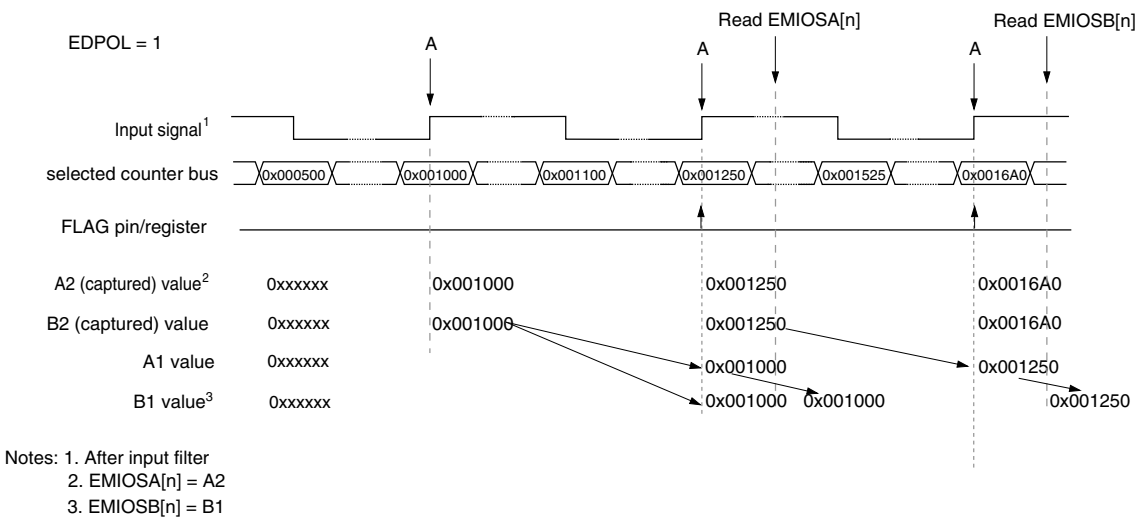


Figure 42-119. A1 and B1 updates at EMIOSA[n] and EMIOSB[n] reads

42.6.1.1.6 Double Action Output Compare (DAOC) Mode

In the DAOC mode, the leading and trailing edges of the variable pulse width output are generated by matches occurring on comparators A and B. There is no restriction concerning the order in which A and B matches occur.

When the DAOC mode is entered, coming out from GPIO mode both comparators are disabled and the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC[n] register.

Data written to A2 and B2 are transferred to A1 and B1, respectively, on the next system clock cycle if OU[n] bit of EMIOSOUDIS register is cleared (see Figure 42-146). The transfer is blocked if OU[n] bit is set. Comparator A is enabled only after the transfer to A1 register occurs and is disabled on the next A match. Comparator B is enabled only after the transfer to B1 register occurs and is disabled on the next B match. Comparators A and B are enabled and disabled independently.

The output flip-flop is set to the value of EDPOL when a match occurs on comparator A and to the complement of EDPOL when a match occurs on comparator B.

MODE[6] controls if the FLAG is set on both matches (MODE[0:6]=0000111) or just on the B match (MODE[0:6]=0000110). FLAG bit assertion depends on comparator enabling.

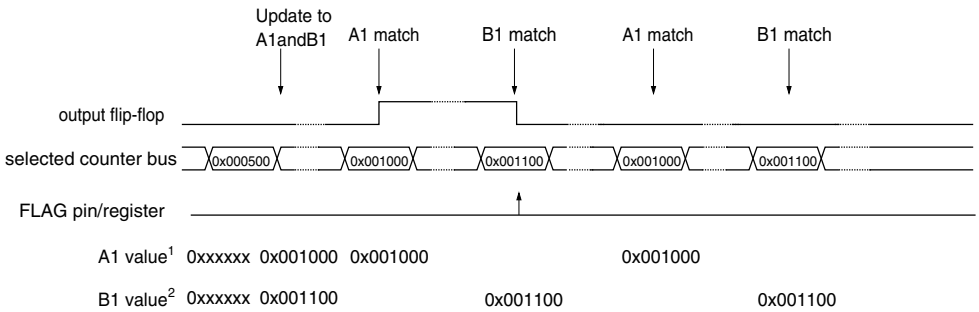
If subsequent enabled output compares occur on registers A1 and B1, pulses will continue to be generated, regardless of the state of the FLAG bit.

At any time, the FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a comparison event in comparator A or B, respectively. Note that the FLAG bit is not affected by these forced operations.

Note

If both registers (A1 and B1) are loaded with the same value, the B match prevails concerning the output pin state (output flip-flop is set to the complement of EDPOL), the FLAG bit is set and both comparators are disabled.

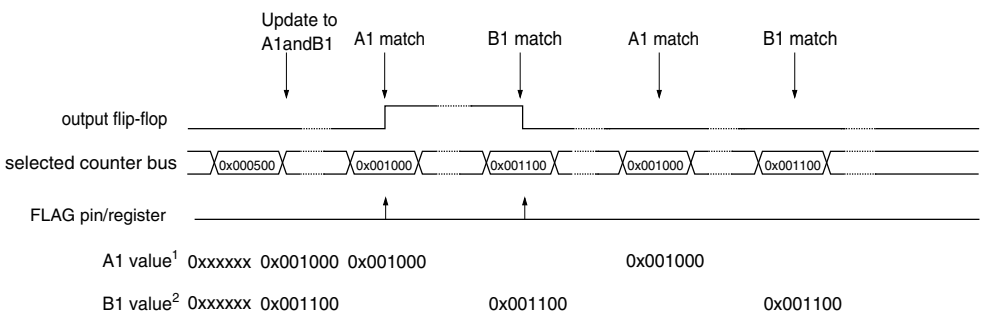
The following two tables show how the Unified Channel can be used to generate a single output pulse with FLAG bit being set on the second match or on both matches, respectively.



Notes:1. EMIOSA[n] = A1 (when reading)
2. EMIOSB[n] = B1 (when reading)

A2 = A1 according to OU[n] bit
B2 = B1 according to OU[n] bit

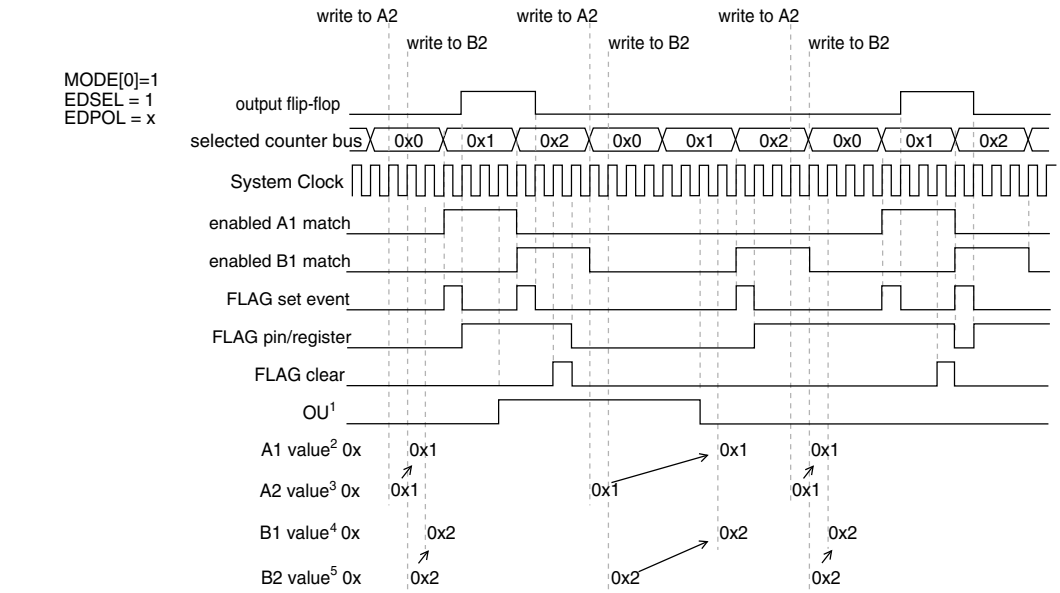
Figure 42-120. Double Action Output Compare with FLAG set on the second match, MODE[0] bit is 0



Notes:1. EMIOSA[n] = A1 (when reading)
2. EMIOSB[n] = B1 (when reading)

A2 = A1 according to OU[n] bit
B2 = B1 according to OU[n] bit

Figure 42-121. Double Action Output Compare with FLAG set on both matches, MODE[0] bit is 1



Note: 1. OU[n] bit of EMIOSOUDIS register
2. EMIOSA[n] = A1 (when reading)
3. EMIOSA[n] = A2 (when writing)
4. EMIOSB[n] = B1 (when reading)
5. EMIOSB[n] = B2 (when writing)

Figure 42-122. DAOC with transfer disabling example

42.6.1.1.7 Pulse/Edge Accumulation (PEA) Mode

The PEA mode returns the time taken to detect a desired number of input events. MODE[6] bit selects between continuous or single shot operation.

After writing to register A1, the internal counter is cleared on the first input event, ready to start counting input events and the selected timebase is latched into register B2. On the match between the internal counter and register A1, a counter bus capture is triggered to register A2 and B2. The data previously held in register B2 is transferred to register B1 and the FLAG bit is set to indicate that an event has occurred. The desired time interval can be determined by subtracting register B1 from A2. Registers EMIOSA[n] and EMIOSB[n] return the values in register A2 and B1, respectively.

As part of the coherency mechanism, reading EMIOSA[n] disables transfers from B2 to B1. These transfers are disabled until the next read of the EMIOS_B_n register. Reading the EMIOSB[n] register re-enables transfers from B2 to B1, to take effect at the next transfer event, as described above. [2]

In order to have coherent data in continuous mode of operation the following steps should be performed, assuming FLAG is initially cleared:

2. If B1 was not updated due to B2 to B1 transfer being disabled after reading register EMIOS_A_n, further EMIOS_A_n] and EMIOS_B_n reads will not return coherent data until a new bus capture is triggered to registers A2 and B2. This capture event is indicated by the channel FLAG being asserted. If enabled, the FLAG also generates an interrupt.

1. Wait for FLAG assertion;
2. Read EMIOSA[n] register;
3. Read EMIOSB[n] register;
4. Clear FLAG bit;
5. Return to step 1.

Accumulation cycles may be lost if the read is not performed in a timely manner. Whenever Overrun bit is asserted it means that one or more cycles have been lost.

Triggering of the counter clock (input event) is done by a rising or falling edge or both edges on the input pin. The polarity of the triggering edge is selected by the EDSEL and EDPOL bits in EMIOS_C_n register.

For continuous operation mode (MODE[6] cleared, MODE[0:6]=0001000), the counter is cleared on the next input event after a FLAG generation and continues to operate as described above.

For single shot operation (MODE[6] set, MODE[0:6]=0001001), the counter is not cleared or incremented after a FLAG generation, until a new writing operation to register A is performed.

Figure 42-123 and Figure 42-124 show how the Unified Channel can be used for continuous and single shot pulse/edge accumulation mode.

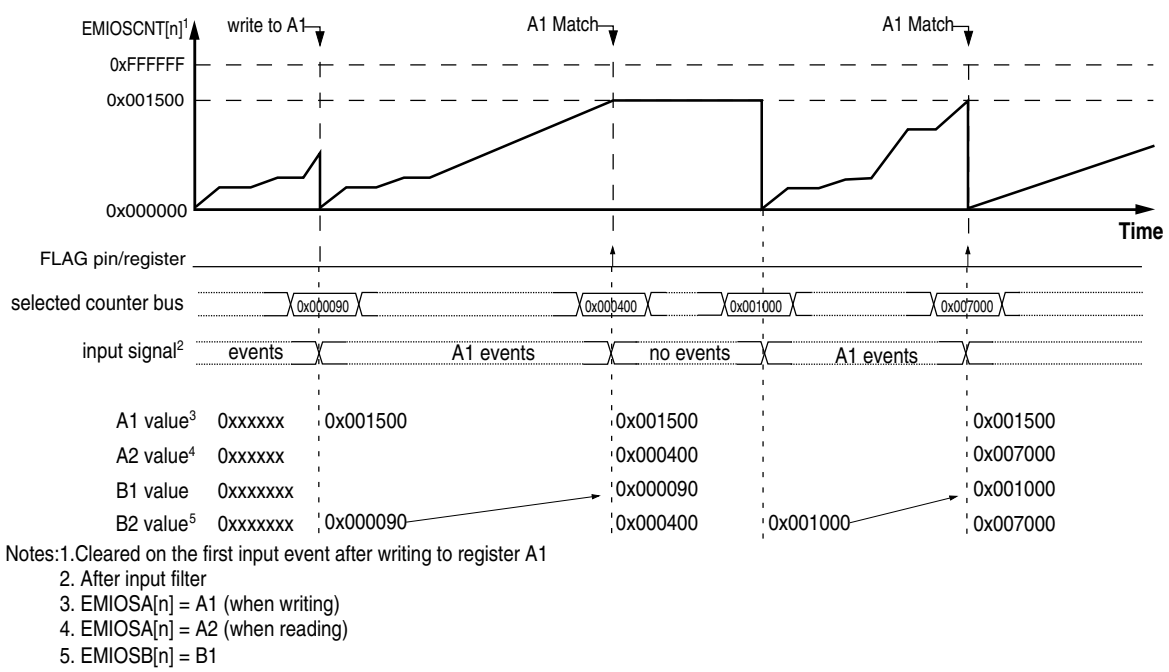


Figure 42-123. Pulse/Edge Accumulation continuous mode example (MODE[6] bit is 0b)

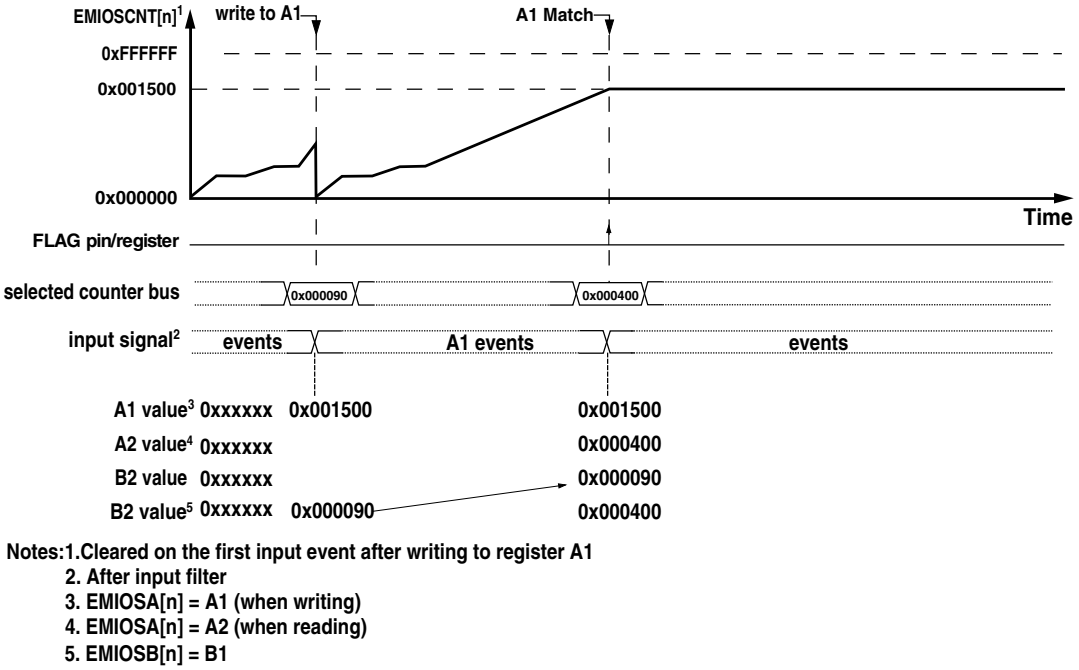


Figure 42-124. Pulse/Edge Accumulation single-shot mode example (MODE[6] bit is 1b)

42.6.1.1.8 Pulse/Edge Counting (PEC) Mode

The PEC mode returns the amount of pulses or edges detected on the input for a desired time window. MODE[6] bit selects between continuous or single shot operation.

Triggering of the internal counter is done by a rising or falling edge or both edges on the input signal. The polarity and the triggering edge is selected by EDSEL and EDPOL bits in EMIOSC[n] register.

Register A1 holds the start time and register B1 holds the stop time for the time window. After writing to register A1, when a match occur between comparator A and the selected timebase, the internal counter is cleared and it is ready to start counting input events. When the time base matches comparator B, the internal counter is disabled and its content is transferred to register A2. At the same time the FLAG bit is set. Reading registers EMIOSCNT[n] or A2 returns the amount of detected pulses.

For continuous operation (MODE[6] cleared, MODE[0:6]=0001010), the next match between comparator A and the selected time base clears the internal counter and counting is enabled again. In order to guarantee coherent measurements when reading EMIOSCNT[n] after the FLAG is set, the software must check if the time base value is out of the time interval defined by registers A1 and B1. Alternatively register A2 always holds the latest available measurement providing coherent data at any time after the first FLAG had occurred. This register is addressed by the alternate address EMIOSALTA[n].

For single shot operation (MODE[6] set, MODE[0:6]=0001011), the next match between comparator A and the selected time base has no effect, until a new write to register A is performed. The EMIOSCNT content is also transferred to register A2 when a match in the B comparator occurs.

Figure 42-125 and Figure 42-126 show how the Unified Channel can be used for continuous or single shot pulse/edge counting mode.

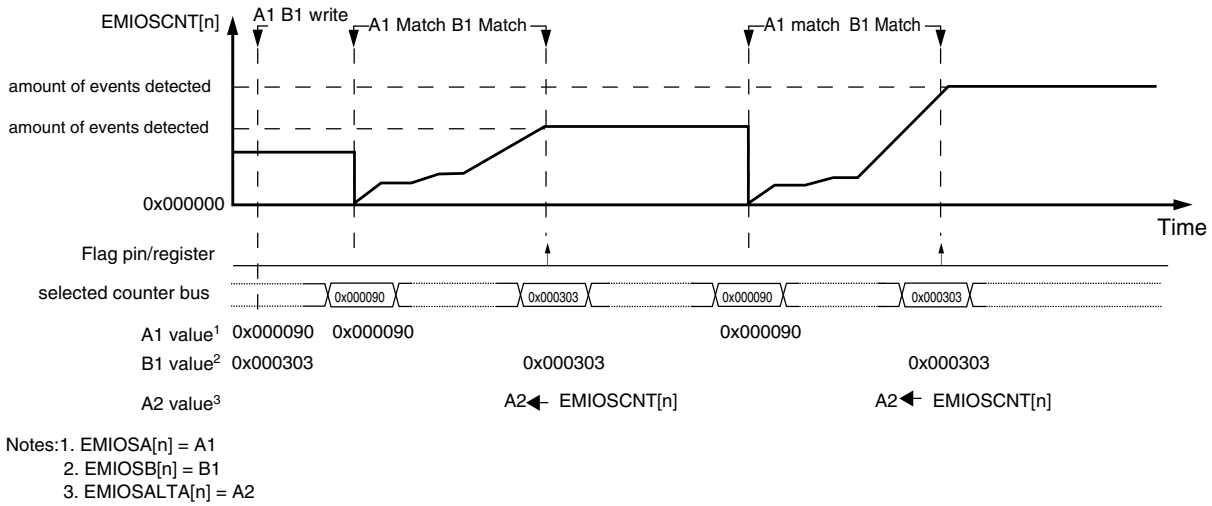


Figure 42-125. Pulse/Edge Counting continuous mode example

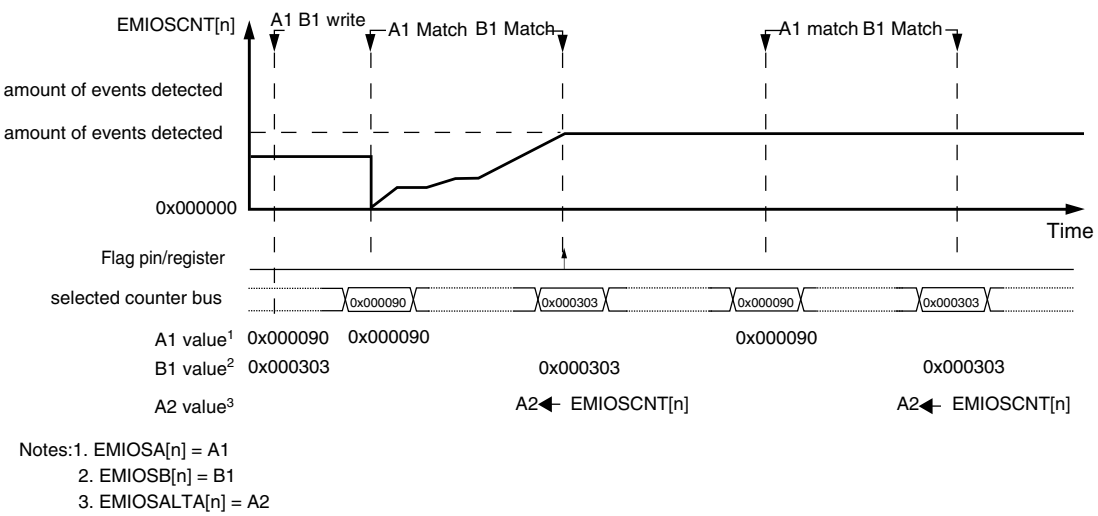


Figure 42-126. Pulse/Edge Counting single-shot mode example

42.6.1.1.9 Quadrature Decode (QDEC) Mode

Quadrature decode mode uses UC[n] operating in QDEC mode and the input programmable filter (IPF) from UC[n-1]. Note that UC[n-1] can be configured, at the same time, to an operation mode that does not use I/O pins, such as MC mode (modulus counter). The connection among the UCs is circular, i.e., when UC[0] is running in QDEC mode, the input programmable filter from UC[23] is being used, for the case of a 24-channel configuration.

This mode generates a FLAG every time the internal counter matches A1 register. The internal counter is automatically selected and is not cleared when entering this mode.

MODE[6] bit selects which type of encoder will be used: *count & direction* encoder or *phase_A and phase_B* encoders.

When operating with *count and direction* encoder (MODE[6] cleared), UC[n] input pin must be connected to the *direction* signal and UC[n-1] input pin must be connected to the *count* signal of the quadrature encoder. UC[n] EDPOL bit selects count direction according to *direction* signal and UC[n-1] EDPOL bit selects if the internal counter is clocked by the rising or falling edge of the *count* signal.

When operating with *phase_A and phase_B* encoder (MODE[6] set), UC[n] input pin must be connected to the *phase_A* signal and UC[n-1] input pin must be connected to the *phase_B* signal of the quadrature encoder. EDPOL bit selects the count direction according to the phase difference between *phase_A and phase_B* signals.

Figure 42-127 and Figure 42-128 show two Unified Channels configured to quadrature decode mode for *count & direction* encoder and *phase_A and phase_B* encoders, respectively.

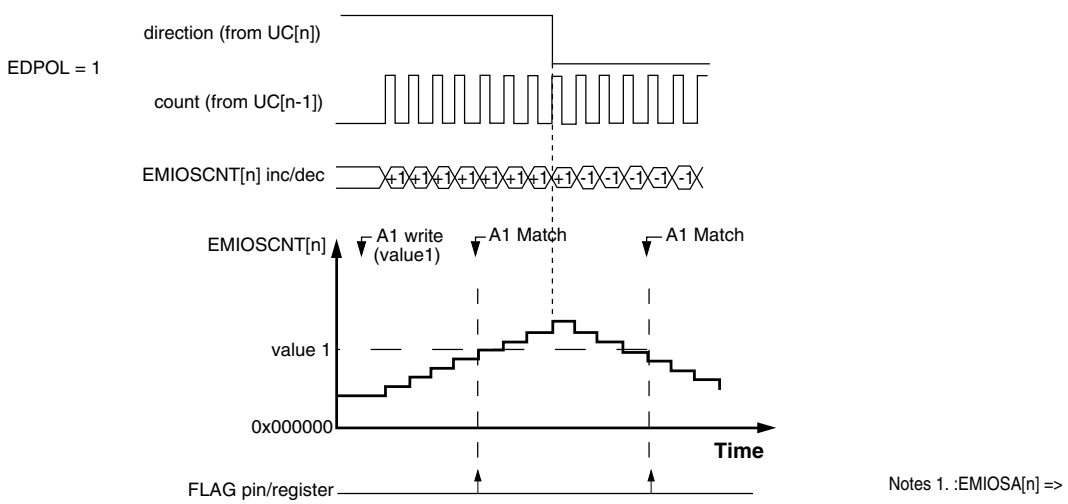


Figure 42-127. Quadrature Decode mode example with *count and direction* encoder

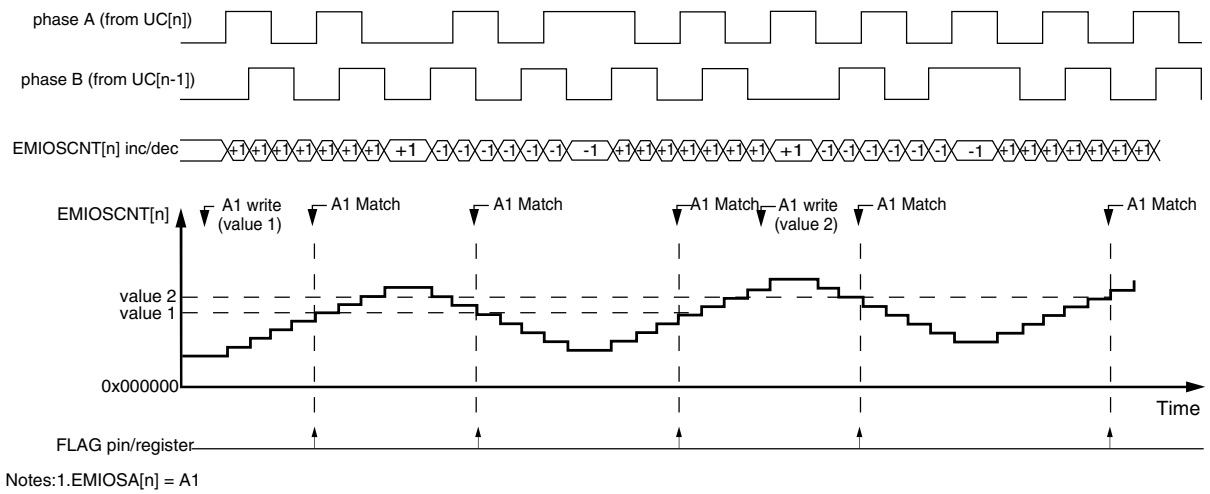


Figure 42-128. Quadrature Decode mode example with *phase_A and phase_B encoder*

42.6.1.1.10 Windowed Programmable Time Accumulation (WPTA) Mode

The WPTA mode (MODE[0:6]=0001110) accumulates the sum of the total high time or low time of an input signal over a programmable interval (time window).

The prescaler bits UCPRE[0:1] in EMIOSC[n] register define the increment rate of the internal counter.

Register A1 holds the start time and register B1 holds the stop time of the programmable time interval. When a match occurs between register A and the selected timebase, the internal counter is cleared and it is ready to start counting. The internal counter is used as a time accumulator, i.e., it counts up when the input signal has the same polarity of EDPOL bit in EMIOSC[n] register and does not count otherwise. When a match occurs in comparator B, the internal counter is disabled regardless of the input signal polarity and the FLAG bit is set. At the same time the content of EMIOSCNT[n] is transferred to register A2. Reading registers EMIOSCNT[n] or A2 returns the high or low time of the input signal.

Note that EMIOSCNT[n] is stable only outside the time window defined from A1 to B1 matches, otherwise its content reflects a count in progress and not the final value. Alternatively to EMIOSCNT register A2 returns the latest available measurement. Since this register is updated only at comparator B matches it always contains stable and up-to-date data. In this mode this register is accessible through the alternate register address EMIOSALTA[n].

The following figure shows how the Unified Channel can be used to accumulate high time.

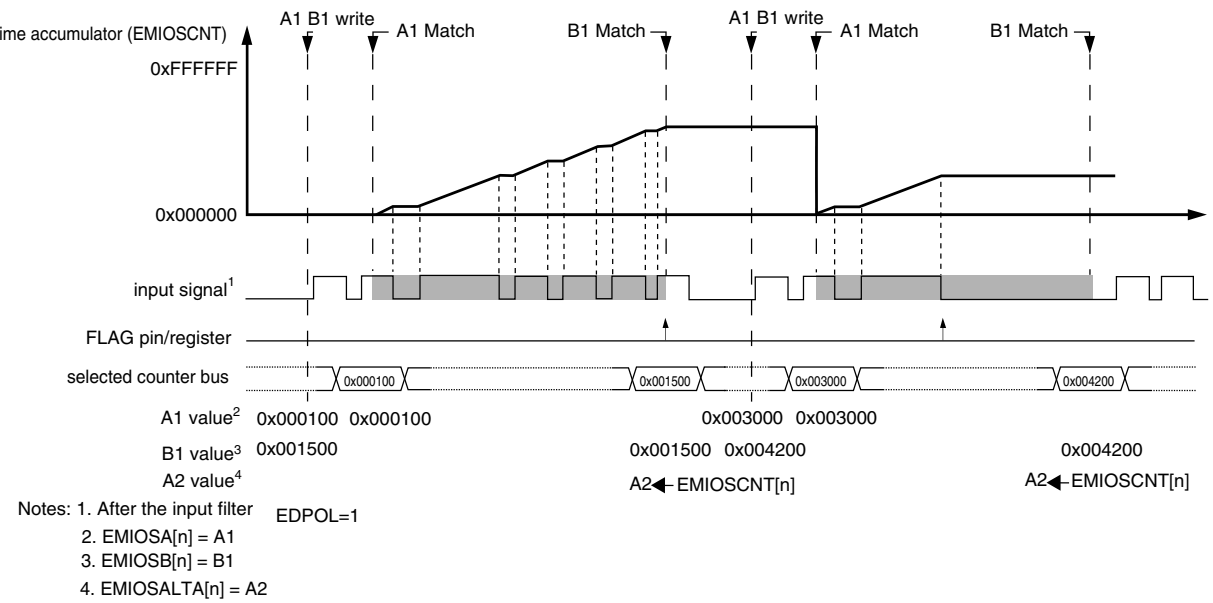


Figure 42-129. Windowed Programmable Time Accumulation example

42.6.1.1.11 Modulus Counter (MC) Mode

The MC mode can be used to provide a time base for a counter bus or as a general purpose timer.

MODE[6] bit selects internal or external clock source when cleared or set, respectively. When external clock is selected, the input signal pin is used as the source and the triggering polarity edge is selected by the EDPOL and EDSEL in the EMIOSC[n] register.

The internal counter counts up from the current value until it matches the value in register A1. Register B1 is cleared and is not accessible to the MCU. MODE[4] bit selects up mode or up/down mode, when cleared or set, respectively.

When in up count mode, a match between the internal counter and register A1 sets the FLAG and clears the internal counter. The timing of those events varies according to the MC mode setup as follows:

- Internal counter clearing on match start (MODE[0:6]=001000b)

- External clock is selected if MODE[0] is set. In this case the internal counter clears as soon as the match signal occurs. The channel FLAG is set at the same time the match occurs. Note that by having the internal counter cleared as soon as the match occurs and incremented at the next input event a shorter zero count is generated. See Figure 42-162 and Figure 42-163.
- Internal clock source is selected if MODE[0] is cleared. In this case the counter clears as soon as the match signal occurs. The channel FLAG is set at the same time the match occurs. At the next prescaler tick after the match the internal counter remains at zero and only resumes counting on the following tick. See Figure 42-162 and Figure 42-164.

- Internal counter clearing on match end (MODE[0:6]=001001b)
  - External clock is selected if MODE[0] is set. In this case the internal counter clears when the match signal is asserted and the input event occurs. The channel FLAG is set at the same time the counter is cleared. See Figure 42-162 and Figure 42-165.
  - Internal clock source is selected if MODE[0] is cleared. In this case the internal counter clears when the match signal is asserted and the prescaler tick occurs. The channel FLAG is set at the same time the counter is cleared. See Figure 42-162 and Figure 42-165.

Note

If internal clock source is selected and the prescaler of the internal counter is set to 1 the MC mode behaves the same way even in Clear on Match Start or Clear on Match End sub-modes.

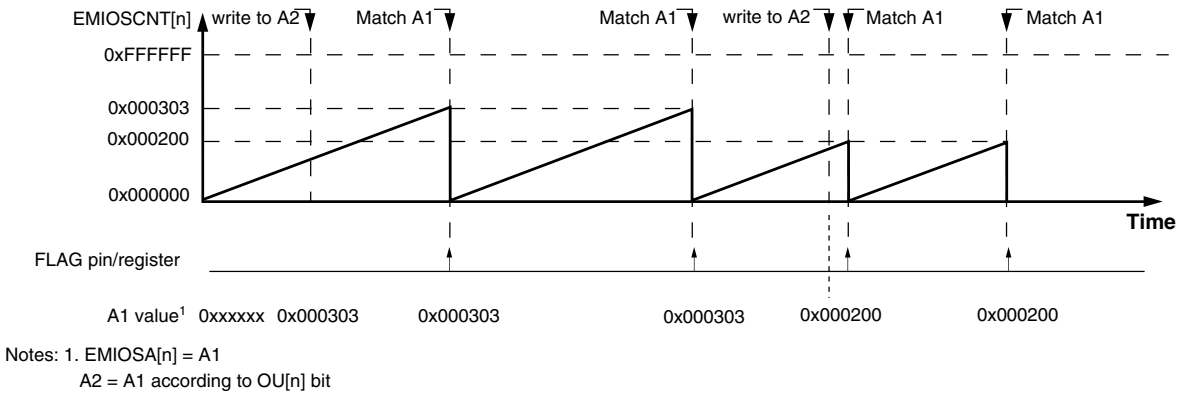
When in up/down count mode (MODE[0:6]=00101bb), a match between the internal counter and register A1 sets the FLAG and changes the counter direction from increment to decrement. A match between register B1 and the internal counter changes the counter direction from decrement to increment and sets the FLAG only if MODE[5] bit is set.

Only values different than 0x0 must be written at A register. Loading 0x0 leads to unpredictable results.

Updates on register A or counter in MC mode may cause loss of match in the current cycle if the transfer occurs near the match. In this case, the counter may rollover and resume operation in the next cycle.

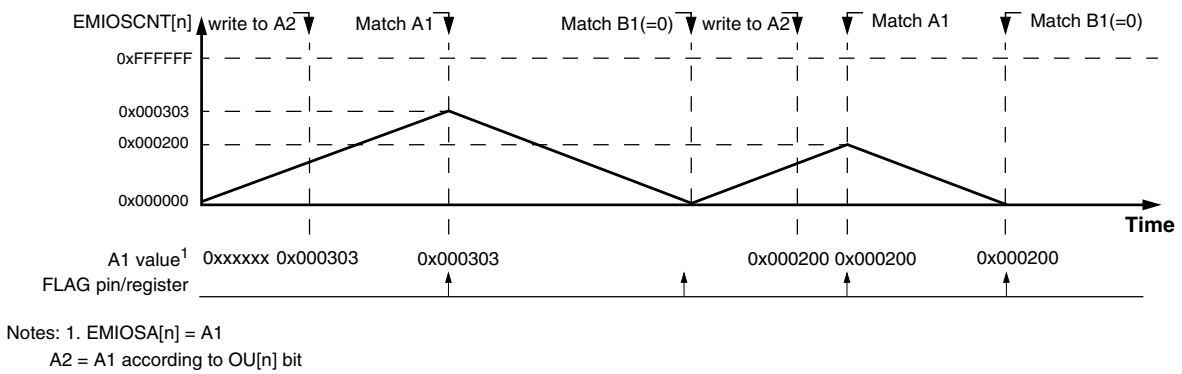
Register B2 has no effect in MC mode. Nevertheless, register B2 can be accessed for reads and writes by addressing EMIOSB.

The following two figures show how the Unified Channel can be used as modulus counter in up mode and up/down mode, respectively.



Notes: 1. EMIOSA[n] = A1
A2 = A1 according to OU[n] bit

Figure 42-130. Modulus Counter up mode example (MODE[4] bit is 0)



Notes: 1. EMIOSA[n] = A1
A2 = A1 according to OU[n] bit

Figure 42-131. Modulus Counter up/down mode example (MODE[4] bit is 1)

42.6.1.1.12 Modulus Counter Buffered (MCB) Mode

The MCB mode provides a time base which can be shared with other channels through the internal counter buses. Register A1 is double buffered thus allowing smooth transitions between cycles when changing A2 register value on the fly. Register A1 is updated at the cycle boundary, which is defined as when the internal counter transitions to 0x1.

The internal counter values operates within a range from 0x1 up to register A1 value. If when entering MCB mode coming out from GPIO mode the internal counter value is not within that range then the A match will not occur causing the channel internal counter to wrap at the maximum counter value which is 0xff_ffff for a 24-bit counter. After the counter wrap occurs it returns to 0x1 and resume normal MCB mode operation. Thus in order to avoid the counter wrap condition make sure its value is within the 0x1 to register A1 value range when the MCB mode is entered.

MODE[6] bit selects internal clock source if cleared or external if set. When external clock is selected the input channel pin is used as the channel clock source. The active edge of this clock is defined by EDPOL and EDSEL bits in the EMIOSC[n] channel register.

When entering in MCB mode, if up counter is selected by MODE[4]=0 (MODE[0:6]=101000b), the internal counter starts counting from its current value to up direction until A1 match occurs. The internal counter is set to 0x1 when its value matches A1 value and a clock tick occurs (either prescaled clock or input pin event).

If up/down counter is selected by setting MODE[4]=1, the counter changes direction at A1 match and counts down until it reaches the value 0x1. After it has reached 0x1 it is set to count in up direction again. Register B1 is used to generate a match in order to set the internal counter in up-count direction if up/down mode is selected. Register B1 cannot be changed while this mode is selected.

Note that differently from the MC mode, the MCB mode counts between 0x1 and A1 register value. Only values greater than 0x1 must be written at register A1. Loading values other than those leads to unpredictable results. The counter cycle period is equal to A1 value in up counter mode. If in up/down counter mode the period is defined by the expression: (2*A1)-2.

Figure 42-132 describes the counter cycle for several A1 values. Register A1 is loaded with A2 register value at the cycle boundary. Thus any value written to A2 register within cycle **n** will be updated to A1 at the next cycle boundary and therefore will be used on cycle **n+1**. The cycle boundary between cycle **n** and cycle **n+1** is defined as when the internal counter transitions from A1 value in cycle **n** to 0x1 in cycle **n+1** . Note that the FLAG is generated at the cycle boundary and has a synchronous operation, meaning that it is asserted one system clock cycle after the FLAG set event.

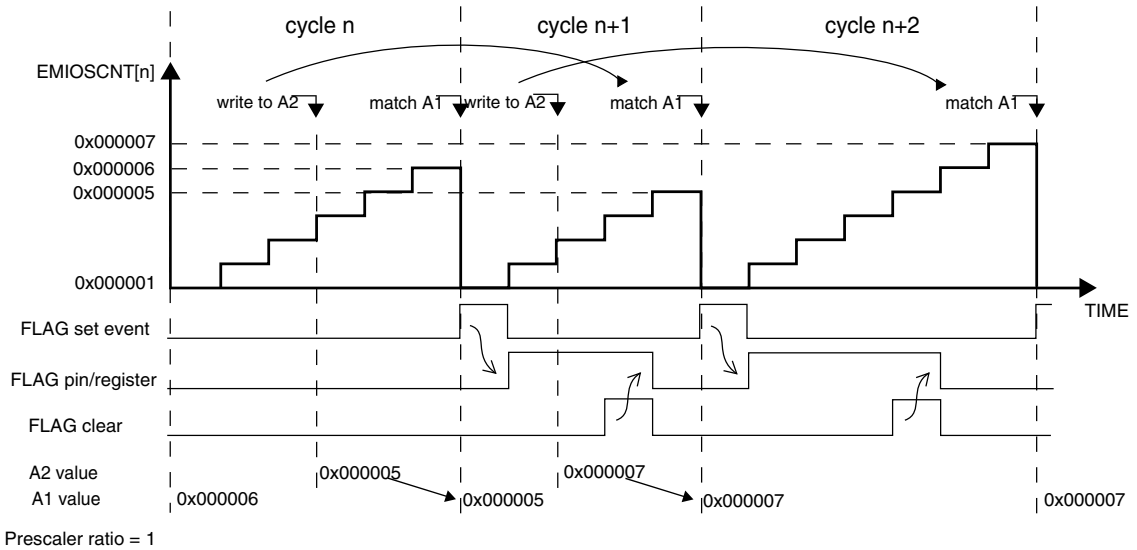


Figure 42-132. Modulus Counter Buffered (MCB) Up Count mode

Figure 42-133 describes the MCB in up/down counter mode (MODE[0:6]=10101bb). Register A1 is updated at the cycle boundary. If A2 is written in cycle **n**, this new value will be used in cycle **n+1** for A1 match. Flags are generated only at A1 match start if MODE[5] is 0. If MODE[5] is set to 1 flags are also generated at the cycle boundary.

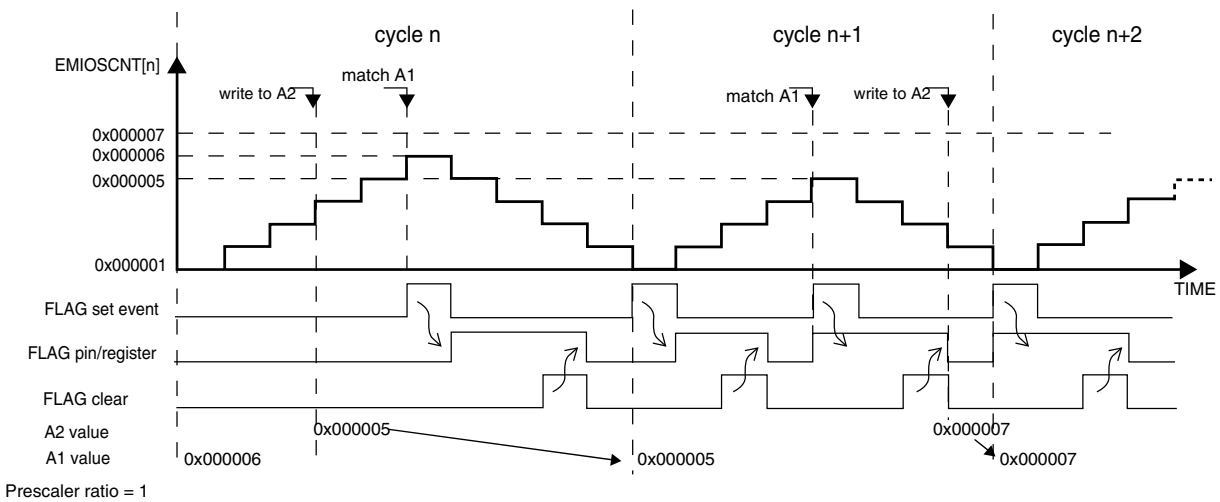


Figure 42-133. Modulus Counter Buffered (MCB) Up/Down Mode

The figure below describes in more detail, the register A1 update process in up counter mode. The A1 load signal is generated at the last system clock period of a counter cycle. Thus, A1 is updated with A2 value at the same time that the counter (EMIOSCNT[n]) is loaded with 0x1. The load signal pulse has the duration of one system clock period. If A2 is written within cycle **n** its value is available at A1 at the first clock of cycle **n+1** and the new value is used for match at cycle **n+1**. The update disable bits OU[n] of EMIOSOUDIS register can be used to control the update of this register, thus allowing to delay the A1 register update for synchronization purposes.

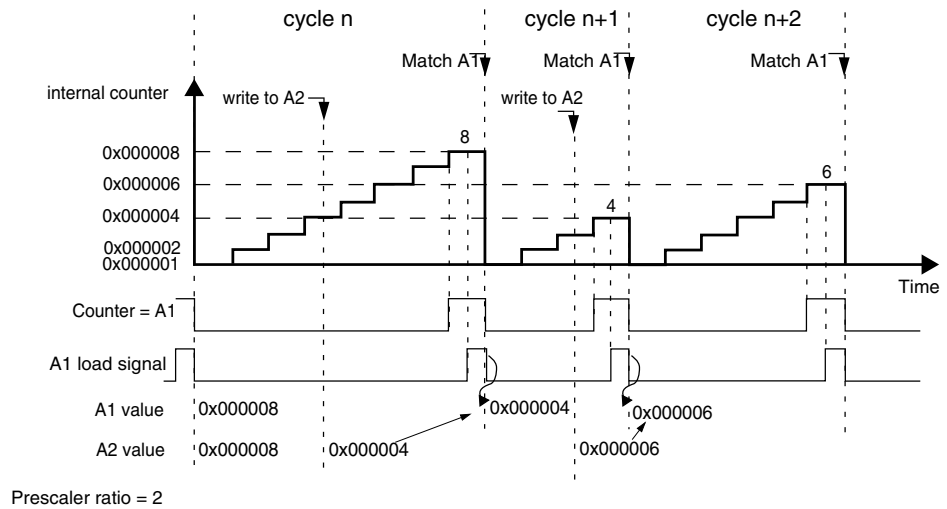


Figure 42-134. MCB Mode Register A1 Update in Up Counter Mode

The figure below describes the register A1 update in up/down counter mode. Note that A2 can be written at any time within cycle **n** in order to be used in cycle **n+1**. Thus A1 receives this new value at the next cycle boundary. Note that the update disable bits OU[n] of EMIOSOUDIS register can be used to disable the update of register A1.

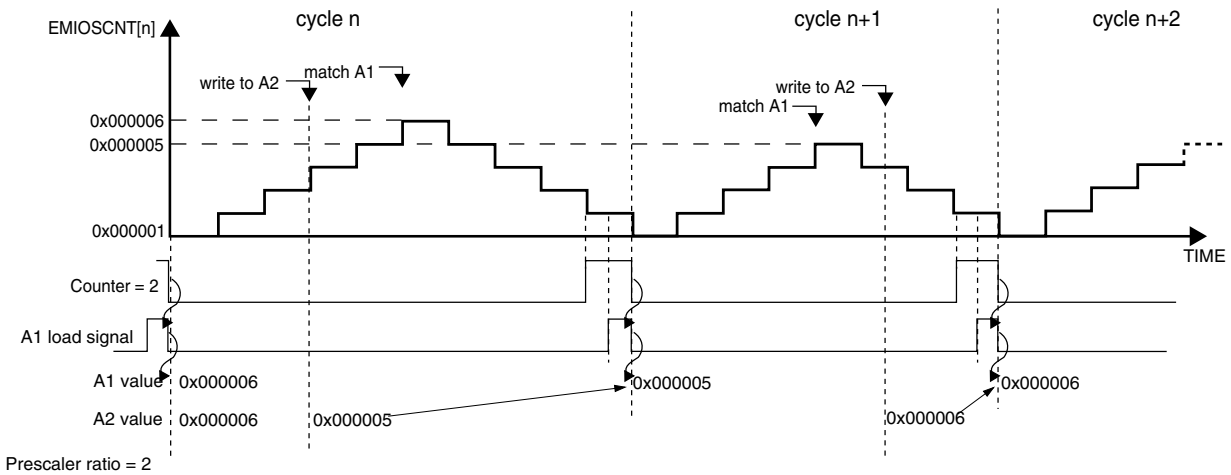


Figure 42-135. MCB Mode Register A1 Update in Up/Down Counter Mode

42.6.1.1.13 Output Pulse Width and Frequency Modulation (OPWFM) Mode

In this mode, duty cycle of output signal is the value defined in register A1 plus one and the period is the value defined in register B1 plus one. MODE[6] bit controls the transfer from register B2 to B1, which can be done either immediately (MODE[6] cleared, MODE[0:6]=00110b0), providing the fastest change in the duty cycle, or at every match of register A1 (MODE[6] set, MODE[0:6]=00110b1).

When OPWFM mode is entered, coming out from GPIO mode, the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC[n] register.

The internal counter is automatically selected as a time base, therefore the BSL[0:1] bits in register EMIOSC[n] has no meaning. When a match on comparator A occurs, the output flip-flop is set to the value of the EDPOL bit. When a match occurs on comparator B, the output flip-flop is set to the complement of the EDPOL bit and the internal counter is cleared.

A FLAG can be generated at match B, when MODE[5] is cleared, or in both matches, when MODE[5] is set.

At any time, the FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on A or B respectively. Also, FORCMB clears the internal counter. Note that the FLAG bit is not set by the FORCMA or FORCMB operations.

If subsequent comparisons occur on comparators A and B, the PWFM pulses continue to be output, regardless of the state of the FLAG bit.

In order to achieve 100% duty cycle, both registers A1 and B1 must be set to the same value. When a simultaneous match occurs on comparators A and B, the output flip-flop is set to the value of EDPOL bit. 0% duty cycle is possible by writing 0x0 to register A. When a match occurs, the output flip-flop is set at every period to the complement of EDPOL bit. The transfer from register B2 to B1 is still controlled by MODE[6] bit.

Note

Writing 0x0 to A1 and B1 produces a duty cycle of 0%.

Figure 42-136 shows the Unified Channel running in OPWFM mode with immediate register update and Figure 42-137 shows the Unified Channel running in OPWFM mode with next period update.

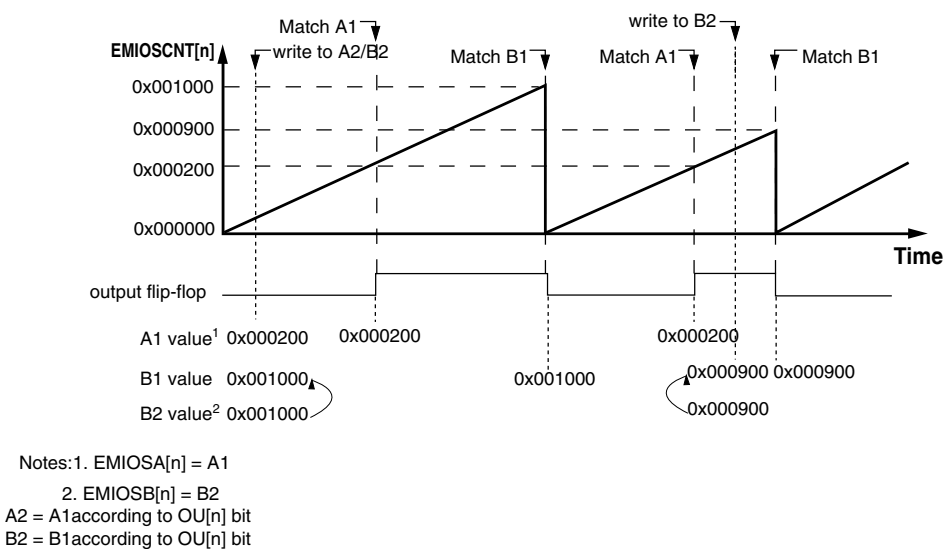


Figure 42-136. OPWFM with immediate update (MODE[6] bit is 0b)

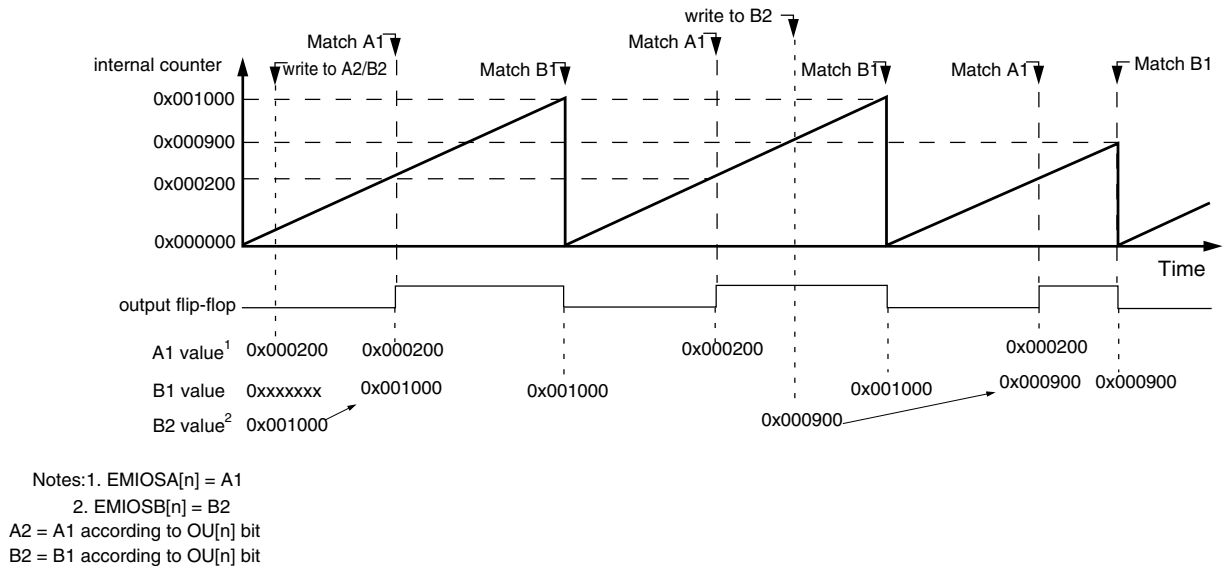


Figure 42-137. OPWFM with next period update (MODE[6] bit is 1b)

42.6.1.1.14 Output Pulse Width and Frequency Modulation Buffered (OPWFMB) Mode

This mode (MODE[0:6]=10110b0) provides waveforms with variable duty cycle and frequency. The internal channel counter is automatically selected as the time base when this mode is selected. Register A1 indicates the duty cycle and B1 register the frequency. Both registers A1 and B1 are double buffered to allow smooth signal generation when changing the registers values.. 0% and 100% duty cycles are supported.

At OPWFMB mode entry, the output flip-flop is set to the value of the EDPOL bit in the EMIOSC[n] register.

In order to provide smooth and consistent channel operation this mode differs substantially from the OPWFM mode. The main differences reside in the A1 and B1 registers update, on the delay from the A1 match to the output pin transition and on the range of the internal counter values which starts from 0x1 up to register B1 value.

When entering OPWFMB mode coming out of GPIO mode, the internal counter value is not within that range then the B match will not occur causing the channel internal counter to wrap at the maximum counter value which is 0xff_ffff for a 24-bit counter. After the counter wrap occurs it returns to 0x1 and resume normal OPWFMB mode operation. Thus in order to avoid the counter wrap condition make sure its value is within the 0x1 to B1 register value range when the OPWFMB mode is entered.

When a match on comparator A occurs the output register is set to the value of EDPOL. When a match on comparator B occurs the output register is set to the complement of EDPOL. B1 match also causes the internal counter to transition to 0x1, thus restarting the counter cycle.

Only values greater than 0x1 are allowed to be written to register B1. Loading values other than those leads to unpredictable results.

Figure 42-138 describes the operation of the OPWFMB mode regarding output pin transitions and A1/B1 registers match events. Note that the output pin transition occurs when the A1 or B1 match signal is deasserted which is indicated by the A1 match negedge detection signal. If register A1 is set to 0x4 the output pin transitions 4 counter periods after the cycle had started, plus one system clock cycle. Note that in the example shown in Figure 42-138 the internal counter prescaler has a ratio of two.

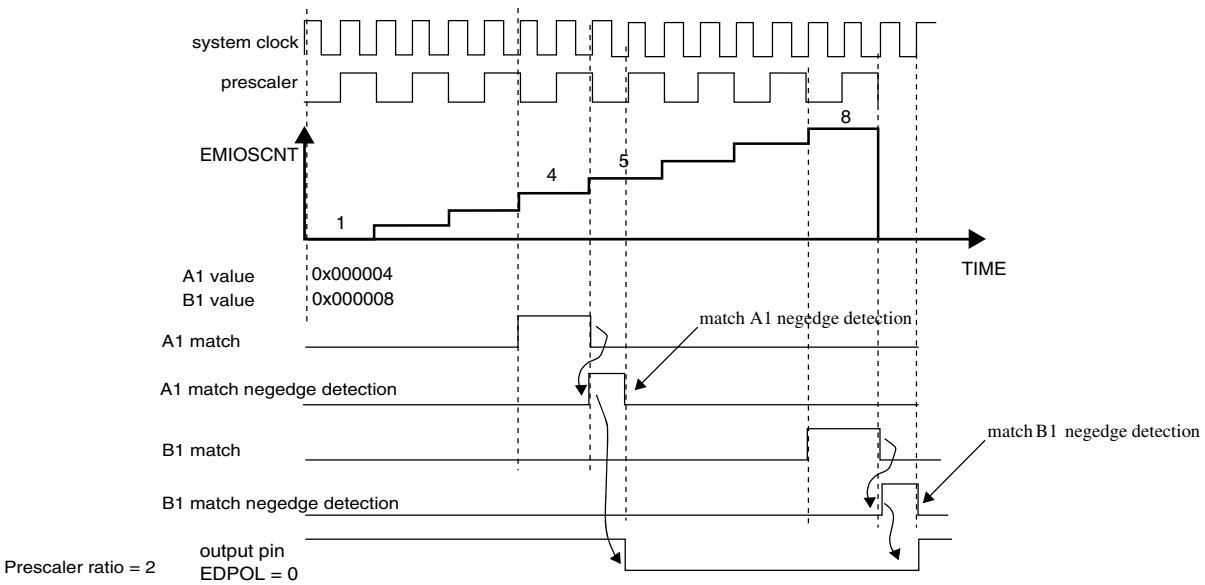


Figure 42-138. OPWFMB A1 and B1 match to Output Register Delay

Figure 42-139 describes the generated output signal if A1 is set to 0x0. Since the counter does not reach zero in this mode, the channel internal logic infers a match as if A1=0x1 with the difference that in this case, the posedge of the match signal is used to trigger the output pin transition instead of the negedge used when A1=0x1. Note that A1 posedge match signal from cycle **n+1** occurs at the same time as B1 negedge match signal from cycle **n** . This allows to use the A1 posedge match to mask the B1 negedge match when they occur at the same time. The result is that no transition occurs on the output flip-flop and a 0% duty cycle is generated.

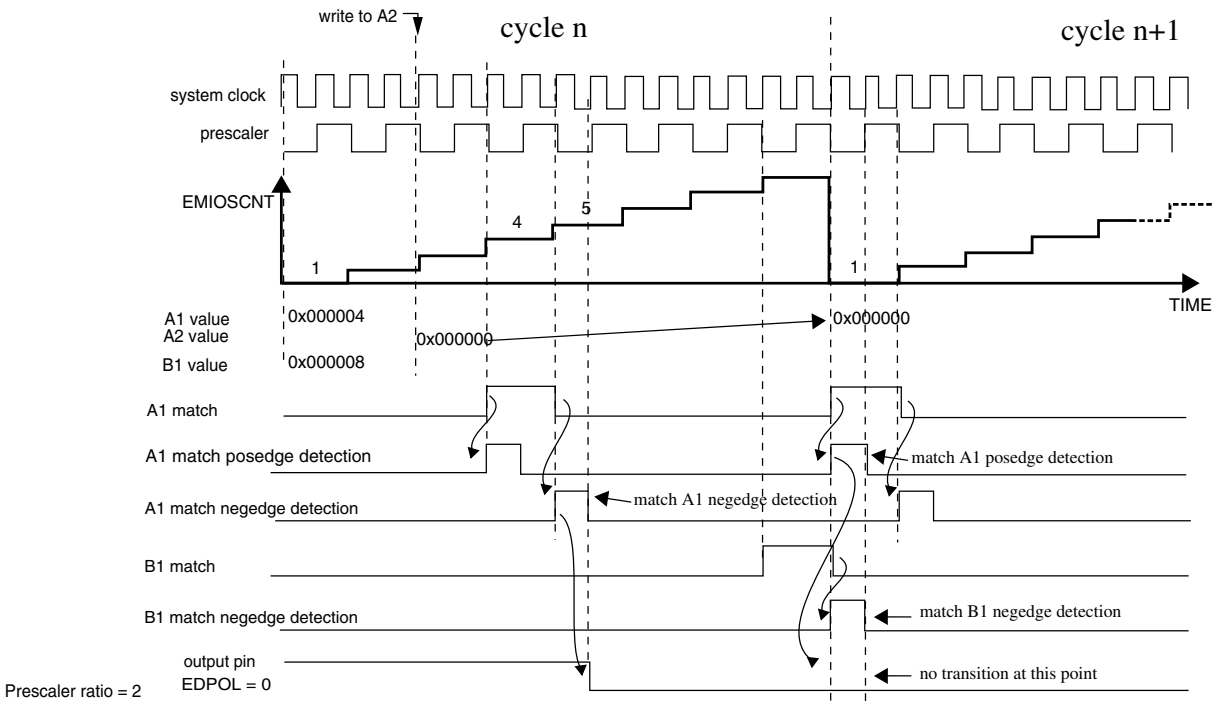


Figure 42-139. OPWFMB Mode with A1 = 0 (0% duty cycle)

Figure 42-140 describes the timing for the A1 and B1 registers load. The A1 and B1 load use the same signal which is generated at the last system clock period of a counter cycle. Thus, A1 and B1 are updated respectively with A2 and B2 values at the same time that the counter (EMIOSCNT[n]) is loaded with 0x1. This event is defined as the cycle boundary. The load signal pulse has the duration of one system clock period. If A2 and B2 are written within cycle **n** their values are available at A1 and B1, respectively, at the first clock of cycle **n+1** and the new values are used for matches at cycle **n+1**. The update disable bits OU[n] of EMIOSOUDIS register can be used to control the update of these registers, thus allowing to delay the A1 and B1 registers update for synchronization purposes.

In Figure 42-140, it is assumed that both the channel and global prescalers are set to 0x1 (each divide ratio is two), meaning that the channel internal counter transitions at every four system clock cycles. FLAGs can be generated only on B1 matches when MODE[5] is cleared, or on both A1 and B1 matches when MODE[5] is set. Since B1 flag occurs at the cycle boundary, this flag can be used to indicate that A2 or B2 data written on cycle **n** were loaded to A1 or B1, respectively, thus generating matches in cycle **n+1**. Note that the FLAG has a synchronous operation, meaning that it is asserted one system clock cycle after the FLAG set event.

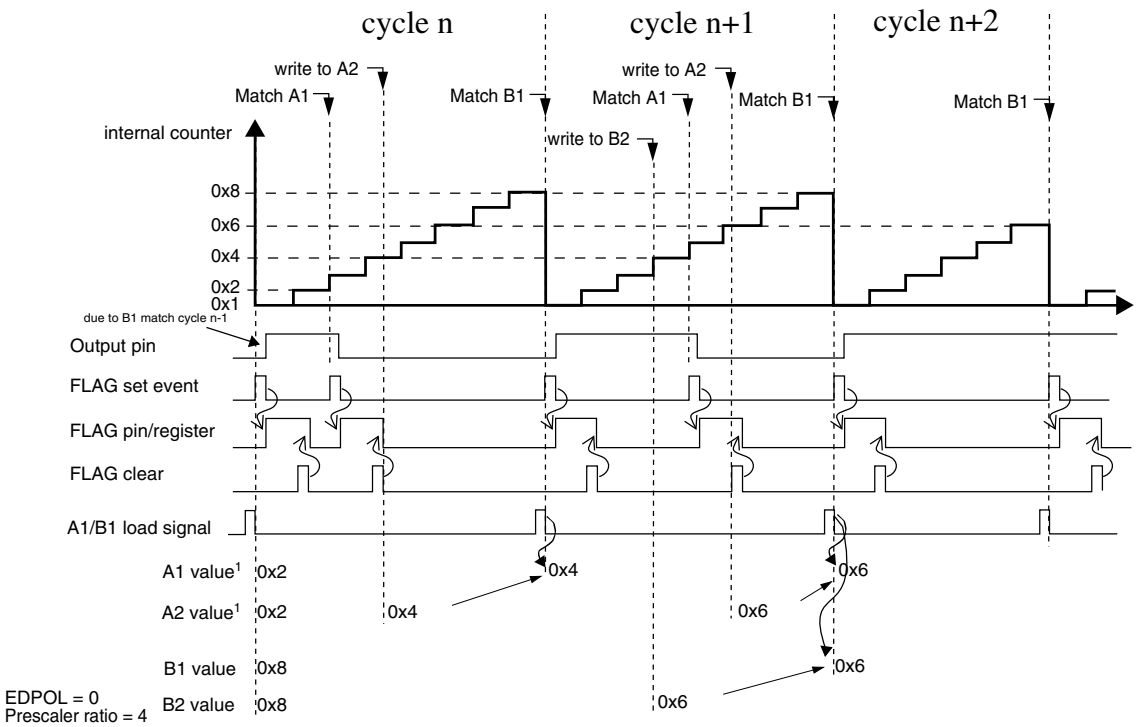


Figure 42-140. OPWFMB A1 and B1 Registers Update and Flags (MODE6] bit is 0b)

Figure 42-141 describes the operation of the Output Disable feature in OPWFMB mode. Differently from the OPWFM mode, the output disable forces the channel output flip-flop to EDPOL bit value. This functionality targets applications that use active high signals and a high to low transition at A1 match. In this case EDPOL should be set to 0. Note that both the channel and global prescalers are set to 0x0 (each divide ratio is one), meaning that the channel internal counter transitions at every system clock cycle.

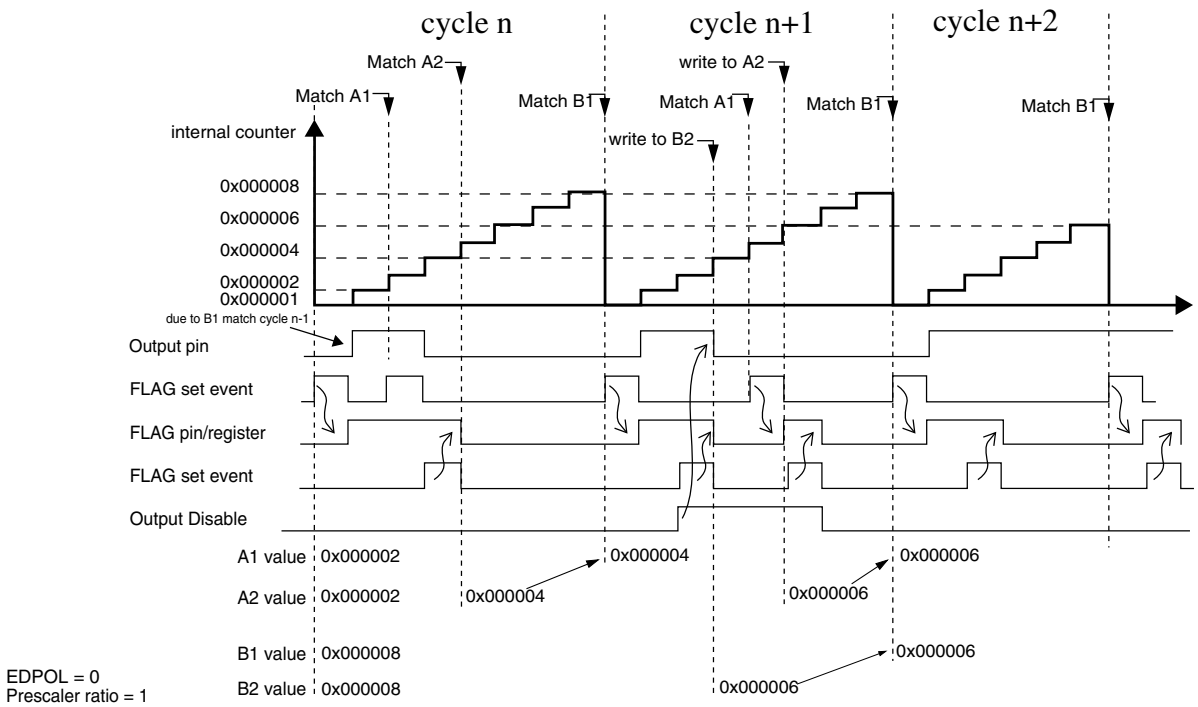


Figure 42-141. OPWFMB Mode with Active Output Disable (MODE[6] bit is 1b)

Note that the output disable has a synchronous operation, meaning that the assertion of the Output Disable input pin causes the channel output flip-flop to transition to EDPOL at the next system clock cycle. If the Output Disable input is deasserted the output pin transition at the following A1 or B1 match.

In Figure 42-141, it is assumed that the Output Disable input is enabled and selected for the Channel. Please, refer to UC Control Register (EMIOSCn) for a detailed description about the ODIS and ODISSL bits, respectively enable and selection of the Output Disable inputs.

The FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on comparators A or B respectively. Similarly to a B1 match FORCMB sets the internal counter to 0x1. The FLAG bit is not set by the FORCMA or FORCMB bits being asserted.

Figure 42-142 describes the generation of 100% and 0% duty cycle signals. It is assumed EDPOL =0 and the resultant prescaler value is 1. Initially, A1=0x8 and B1=0x8. In this case, B1 match has precedence over A1 match, thus the output flip-flop is set to the complement of EDPOL bit. This cycle corresponds to a 100% duty cycle signal. The same output signal can be generated for any A1 value greater or equal to B1.

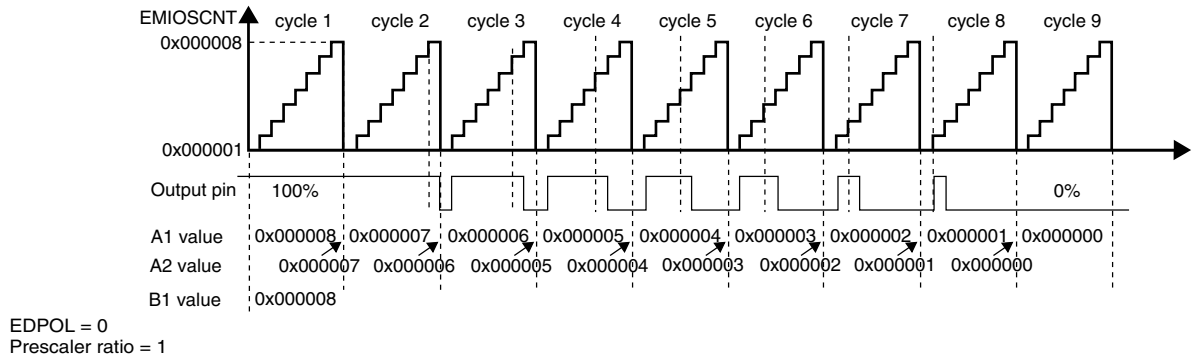


Figure 42-142. OPWFMB Mode from 100% to 0% Duty Cycle

A 0% duty cycle signal is generated if A1=0x0 as shown in Figure 42-142 cycle 9. In this case B1=0x8 match from cycle 8 occurs at the same time as the A1=0x0 match from cycle 9. Please, refer to Figure 42-139 for a description of the A1 and B1 match generation. In this case A1 match has precedence over B1 match and the output signal transitions to EDPOL.

42.6.1.1.15 Center Aligned Output Pulse Width Modulation with Dead Time Insertion (OPWMC) Mode

This operation mode generates a center aligned PWM with dead time insertion in the leading (MODE[0:6]=00111b1) or trailing edge (MODE[0:6]=00111b0).

The selected counter bus must be running an up/down time base, as shown in the 'Modulus Counter up/down mode example' figure. BSL[0:1] bits select the time base. Register A1 contains the ideal duty cycle for the PWM signal and is compared with the selected time base. Register B1 contains the dead time value and is compared with the internal counter. For a leading edge dead time insertion, the output PWM duty cycle is equal to the difference between register A1 and register B1, and for a trailing edge dead time insertion, the output PWM duty cycle is equal to the sum of register A1 and register B1. Mode[6] bit selects between trailing and leading dead time insertion, respectively.

Note

The internal counter may be running in the internal prescaler ratio, while the selected time base may be running in a different prescaler ratio. The output signal may produce an unexpected output if the dead time interval is greater than the duty cycle of the PWM signal.

When OPWMC mode is entered, coming out from GPIO mode, the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC[n] register.

When operating with leading edge dead time insertion, the first match between A1 and the selected time base clears the internal counter and switches the selected time base to the internal counter. When a match occurs between register B1 and the selected time base, the output flip-flop is set to the value of the EDPOL bit and the time base is switched to the selected counter bus. In the next match between register A1 and the selected time base, the output flip-flop is set to the complement of the EDPOL bit. This sequence repeats continuously.

When operating with trailing edge dead time insertion, the first match between A1 and the selected time base sets the output flip-flop to the value of the EDPOL bit. In the next match between register A1 and the selected time base, the internal counter is cleared and the selected time base is switched to the internal counter. When a match occurs between register B1 and the selected time base, the output flip-flop is set to the complement of the EDPOL bit and the time base is switched to the selected counter bus. This sequence repeats continuously.

FLAG can be generated in the trailing edge of the output PWM signal when MODE[5] is cleared, or in both edges, when MODE[5] is set.

At any time, the FORCMA or FORCMB bits are equivalent to a successful comparison on comparator A or B with the exception that the FLAG bit is not set.

Note

When in freeze state, the FORCMA or FORCMB bits only allow the software to force the output flip-flop to the level corresponding of a match on A or B respectively.

If subsequent matches occur on comparators A and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

In order to achieve a duty cycle of 100%, both registers A1 and B1 must be set to the same value. When a simultaneous match occurs between the selected time base and registers A1 and B1, the output flip-flop is set to the value of EDPOL bit and the selected time base switches to the selected counter bus, allowing a new cycle to begin at any time, as previously described. 0% duty cycle is possible by writing 0x0 to register A. When a match occurs, the output flip-flop is set to the complement of EDPOL bit and the selected time base switches to the selected counter bus, allowing a new cycle to begin at any time, as previously described. In both cases, FLAG is generated regardless of MODE[6] bit.

Note

If A1 and B1 are set to the 0x0, a 0% duty cycle waveform is produced.

Figure 42-143 and Figure 42-144 show the Unified Channel running in OPWMC with leading and trailing dead time, respectively.

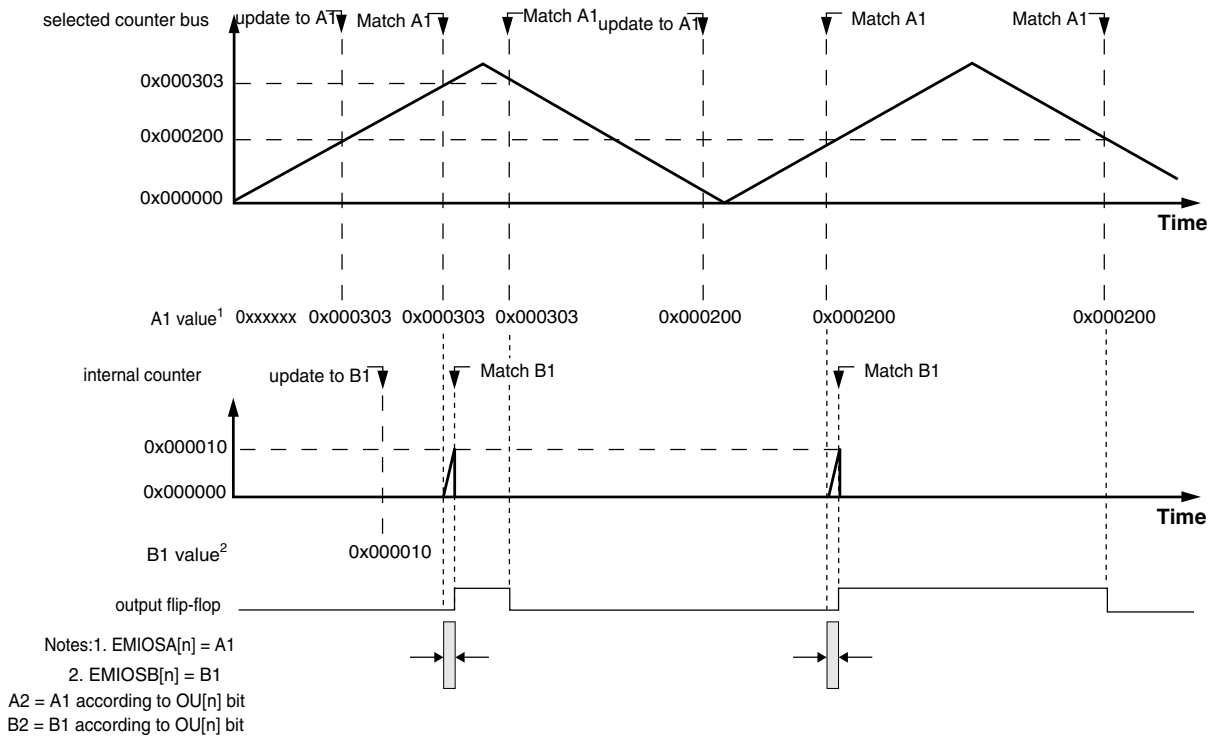


Figure 42-143. OPWMC with leading dead time insertion (MODE [6] bit is 1b)

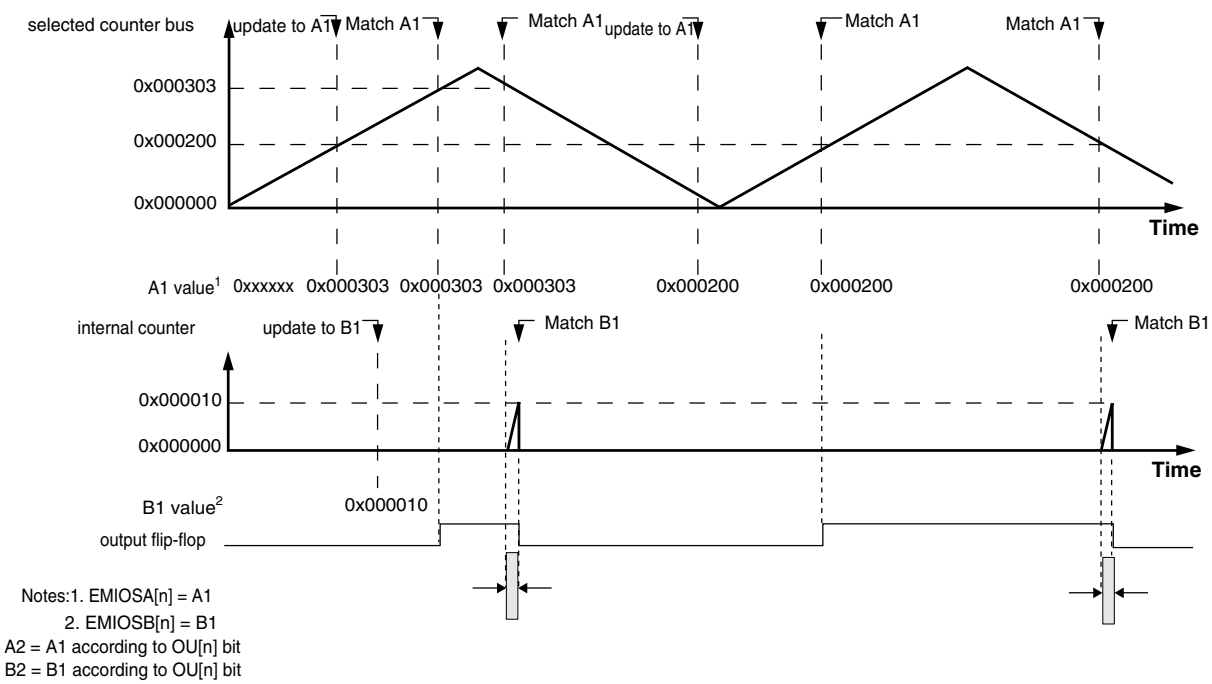


Figure 42-144. OPWMC with trailing dead time insertion (MODE [6] bit is 0b)

42.6.1.1.16 Center Aligned Output Pulse Width Modulation with Dead Time Insertion Buffered (OPWMCB) Mode

This operation mode generates a center aligned PWM with dead time insertion to the leading (MODE[0:6]=10111b1) or trailing edge (MODE[0:6]=10111b0). A1 and B1 registers are double buffered to allow smooth output signal generation when changing A2 or B2 registers values.

BSL[0:1] bits select the time base. The time base selected for a channel configured to OPWMCB mode should be a channel configured to MCB Up/Down mode, as shown in Figure 42-133 . It is recommended to start the MCB channel time base after the OPWMCB mode is entered in order to avoid missing A matches at the very first duty cycle.

Register A1 contains the ideal duty cycle for the PWM signal and is compared with the selected time base.

Register B1 contains the dead time value and is compared against the internal counter. For a leading edge dead time insertion, the output PWM duty cycle is equal to the difference between register A1 and register B1, and for a trailing edge dead time insertion, the output PWM duty cycle is equal to the sum of register A1 and register B1. Mode[6] bit selects between trailing and leading dead time insertion, respectively.

Note

The internal counter runs in the internal prescaler ratio, while the selected time base may be running in a different prescaler ratio.

When OPWMCB mode is entered, coming out from GPIO mode, the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC[n] register.

The following basic steps summarize proper OPWMCB startup, assuming the channels are initially in GPIO mode:

1. *[global]* Disable Global Prescaler.
2. *[MCB channel]* Disable Channel Prescaler.
3. *[MCB channel]* Write 0x1 at internal counter.
4. *[MCB channel]* Set A register.
5. *[MCB channel]* Set channel to MCB Up mode.
6. *[MCB channel]* Set prescaler ratio.
7. *[MCB channel]* Enable Channel Prescaler.

8. *[OPWMCB channel]* Disable Channel Prescaler.
9. *[OPWMCB channel]* Set A register.
10. *[OPWMCB channel]* Set B register.
11. *[OPWMCB channel]* Select time base input through BSL[1:0] bits.
12. *[OPWMCB channel]* Enter OPWMCB mode.
13. *[OPWMCB channel]* Set prescaler ratio.
14. *[OPWMCB channel]* Enable Channel Prescaler.
15. *[global]* Enable Global Prescaler.

The following figure describes the load of A1 and B1 registers which occurs when the selected counter bus transitions from 0x2 to 0x1 . This event defines the cycle boundary. Note that values written to A2 or B2 within cycle **n** are loaded into A1 or B1 registers, respectively, and used to generate matches in cycle **n+1** .

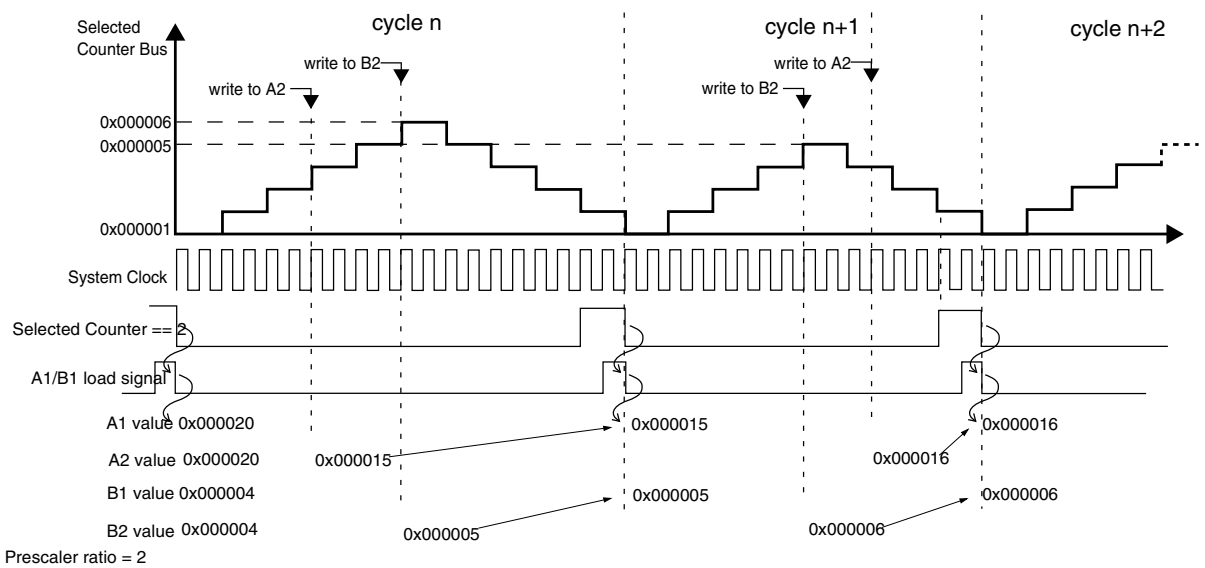


Figure 42-145. OPWMCB A1 and B1 registers load

The OU[n] bit of EMIOSOUDIS register can be used to disable the A1 and B1 updates, thus allowing to synchronize the load on these registers with the load of A1 or B1 registers in others channels. Note that using the update disable bit A1 and B1 registers can be updated at the same counter cycle thus allowing to change both registers at the same time.

In this mode, A1 matches always sets the internal counter to 0x1 . When operating with leading edge dead time insertion the first A1 match sets the internal counter to 0x1. When a match occurs between register B1 and the internal time base, the output flip-flop is set

to the value of the EDPOL bit. In the following match between register A1 and the selected time base, the output flip-flop is set to the complement of the EDPOL bit. This sequence repeats continuously. The internal counter should not reach 0x0 as consequence of a rollover. In order to avoid it, the user should not write to EMIOSB register a value greater than twice the difference between external count up limit and EMIOSA value.

The following figure shows two cycles of a Center Aligned PWM signal. Note that both A1 and B1 register values are changing within the same cycle which allows to vary at the same time the duty cycle and dead time values.

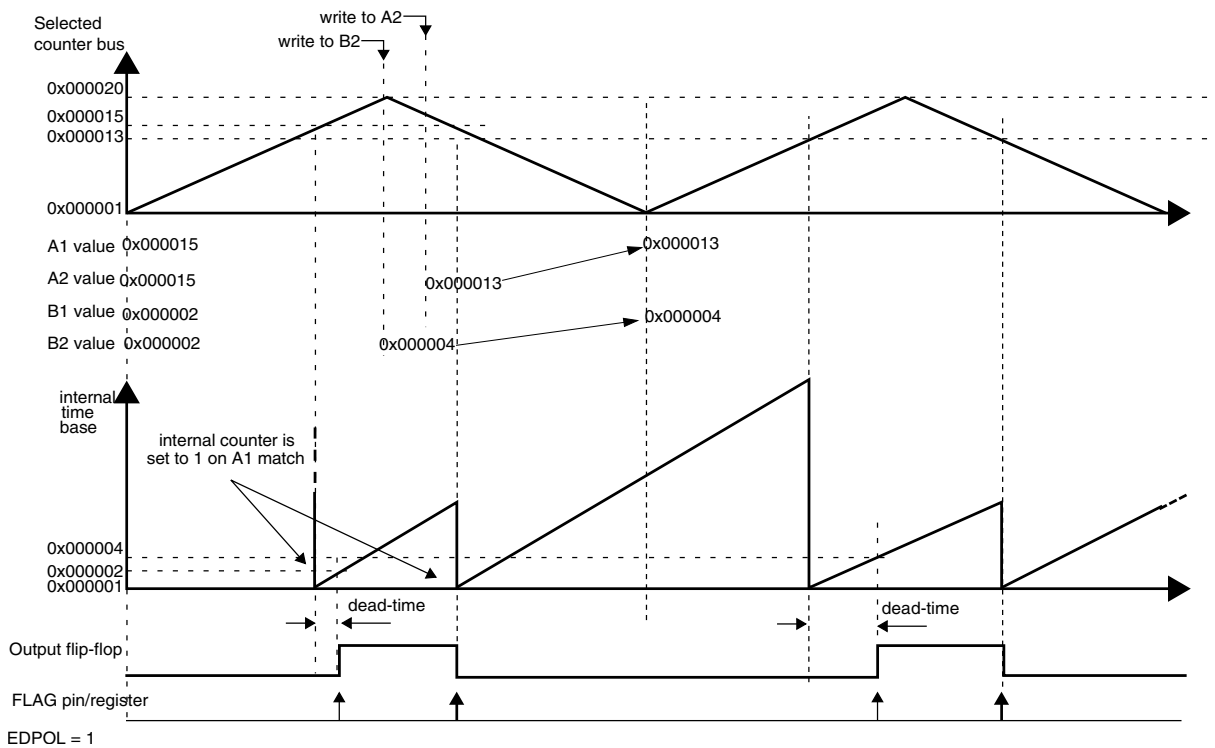


Figure 42-146. OPWMCB with Lead Dead Time Insertion

When operating with trailing edge dead time insertion, the first match between A1 and the selected time base sets the output flip-flop to the value of the EDPOL bit and sets the internal counter to 0x1. In the second match between register A1 and the selected time base, the internal counter is set to 0x1 and B1 matches are enabled. When the match between register B1 and the selected time base occurs the output flip-flop is set to the complement of the EDPOL bit. This sequence repeats continuously.

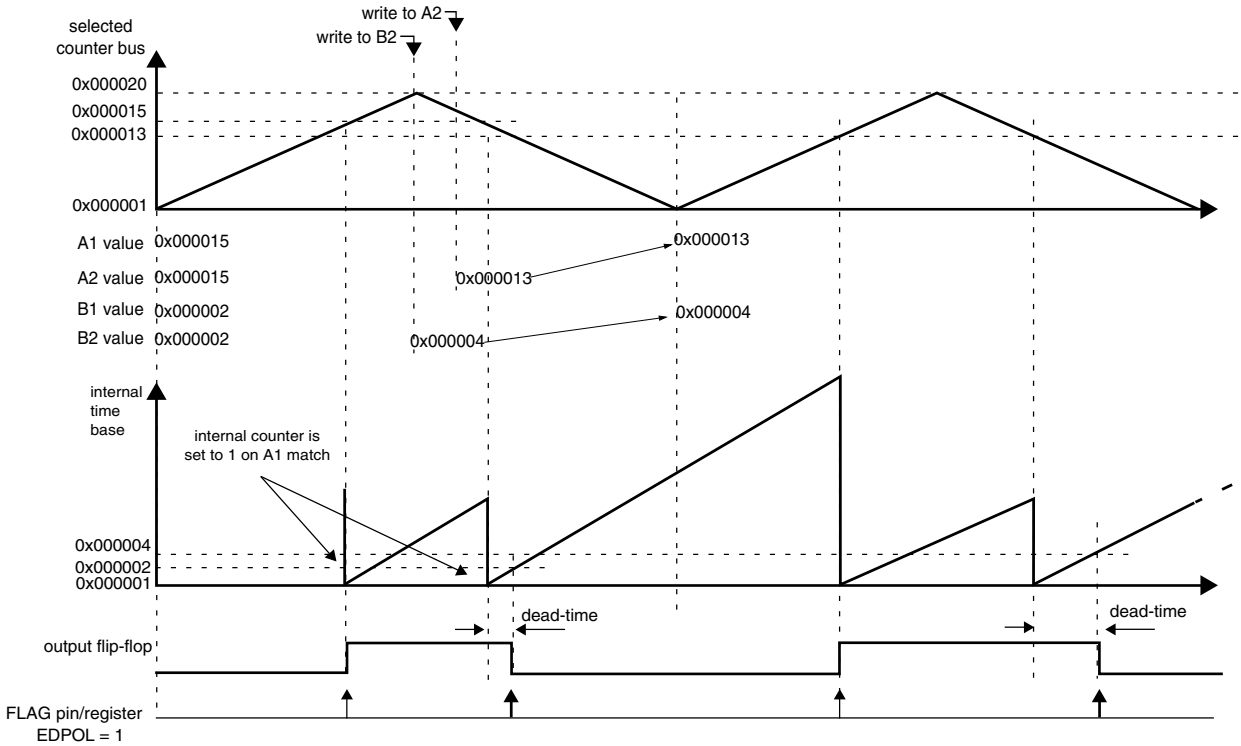


Figure 42-147. OPWMCB with Trail Dead Time Insertion

A FLAG can be generated in the trailing edge of the output PWM signal when MODE[5] is cleared, or in both edges, when MODE[5] is set. If subsequent matches occur on comparators A and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

Note

In OPWMCB mode FORCMA and FORCMB do not have the same behavior as a regular match. Instead they force the output flip-flop to constant value which depends upon the selected dead time insertion mode, lead or trail and the value of the EDPOL bit.

FORCMA has different behaviors depending upon the selected dead time insertion mode, lead or trail. In lead dead time insertion FORCMA force a transition in the output flip-flop to the opposite of EDPOL. In trail dead time insertion the output flip-flop is forced to the value of EDPOL bit.

If FORCMB bit is set, the output flip-flop value depends upon the selected dead time insertion mode. In lead dead time insertion FORCMB forces the output flip-flop to transition to EDPOL bit value. In trail dead time insertion the output flip-flop is forced to the opposite of EDPOL bit value.

Note

FORCMA bit set does not set the internal time-base to 0x1 as a regular A1 match.

The FLAG bit is not set either in case of a FORCMA or FORCMB or even if both forces are issued at the same time.

Note

FORCMA and FORCMB have the same behavior even in Freeze or normal mode regarding the output pin transition.

When *FORCMA* is issued along with *FORCMB* the output flip-flop is set to the opposite of EDPOL bit value. This is equivalent of saying that *FORCMA* has precedence over *FORCMB* when lead dead time insertion is selected and *FORCMB* has precedence over *FORCMA* when trail dead time insertion is selected.

A duty cycle from 0% to 100% can be generated by setting appropriate values to A1 and B1 registers relatively to the period of the external time base. Setting A1=0x1 generates a 100% duty cycle waveform. Assuming EDPOL is set to one and OPWMCB mode with trail dead time insertion, 100% duty cycle signals can be generated if B1 occurs at or after the cycle boundary (external counter = 1). If A1 is greater than the maximum value of the selected counter bus period, then a 0% duty cycle is produced, only if the pin starts the current cycle in the opposite of EDPOL value. In case of 100% duty cycle, the transition from EDPOL to the opposite of EDPOL may be obtained by forcing pin, using FORCMA and/or FORCMB.

Note

If A1 is set to 0x1 at OPWMCB entry the 100% duty cycle may not be obtained in the very first PWM cycle due to the pin condition at mode entry.

Only values different than 0x0 are allowed to be written to A1 register. If 0x0 is loaded to A1, the results are unpredictable.

Note

A special case occurs when A1 is set to (external counter bus period)/2, which is the maximum value of the external counter. In this case the output flip-flop is constantly set to the EDPOL bit value.

The internal channel logic prevents matches from one cycle to propagate to the next cycle. In trail dead time insertion B1 match from cycle **n** could eventually cross the cycle boundary and occur in cycle **n+1.** In this case B1 match is masked out and does not cause the output flip-flop to transition. Therefore matches in cycle **n+1** are not affected by the late B1 matches from cycle **n**.

The following figure shows a 100% duty cycle output signal generated by setting A1=0x4 and B1=0x3. In this case, the trailing edge is positioned at the boundary of cycle **n+1**, which is actually considered to belong to cycle **n+2** and therefore does not cause the output flip-flip to transition.

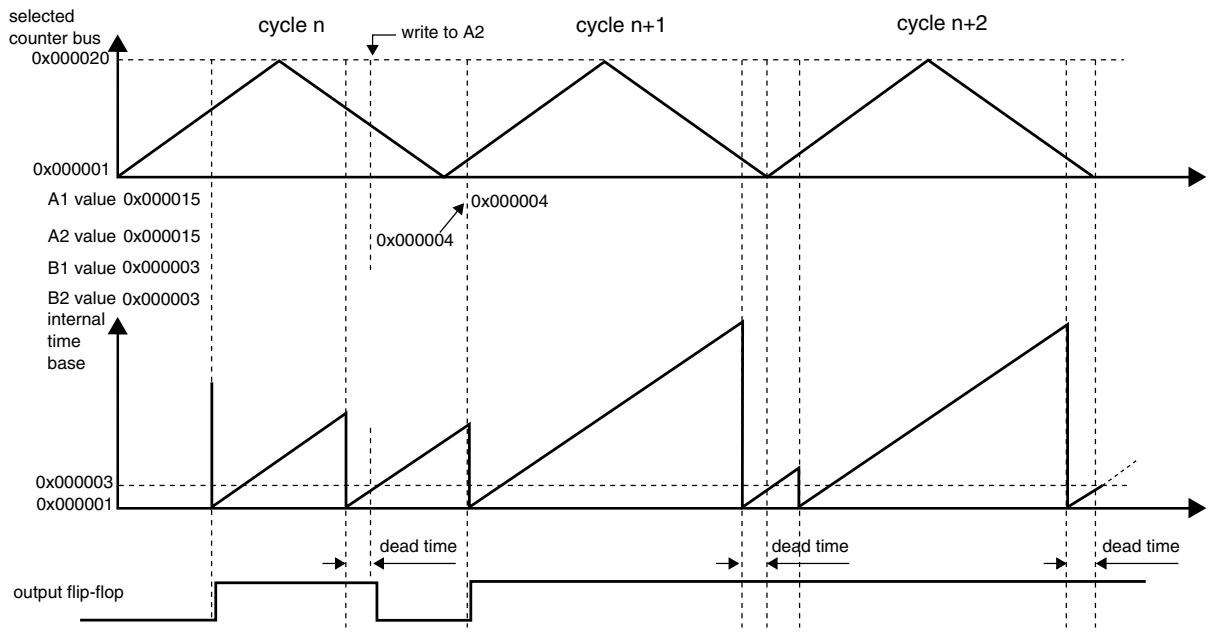


Figure 42-148. OPWMCB with 100% Duty Cycle (A1=4 and B1=3)

The output disable feature, if enabled, causes the output flip-flop to transition to the EDPOL inverted state. This feature allows an application to force the channel output pin to a "safe" state. The internal channel matches continue to occur even in this case, thus generating Flags. As soon as the output disable is deasserted, the channel output pin is again controlled by A1 and B1 matches. Note that this process is synchronous, meaning that the output channel pin transitions only on system clock edges.

It is important to notice that, such as in OPWMB and OPWFMB modes, the match signal used to set or clear the channel output flip-flop is generated on the deassertion of the channel combinational comparator output signal which compares the selected time base with A1 or B1 register values. Please refer to Figure 42-138, which describes the delay from matches to output flip-flop transition in OPWFMB mode. The operation of OPWMCB mode is similar to OPWFMB regarding matches and output pin transition.

42.6.1.1.17 Output Pulse Width Modulation (OPWM) Mode

Registers A1 and B1 define the leading and trailing edges of the PWM output pulse, respectively. MODE[6] bit controls the transfer from register B2 to B1, which can be done either immediately (MODE[6] cleared, MODE[0:6]=01000b0), providing the fastest change in the duty cycle, or at every match of register A1 (MODE[6] set, MODE[0:6]=01000b1).

The value loaded in register A1 is compared with the value on the selected time base. When a match on comparator A occurs, the output flip-flop is set to the value of the EDPOL bit. When a match occurs on comparator B, the output flip-flop is set to the complement of the EDPOL bit.

A FLAG can be generated at match B, when MODE[5] is cleared, or in both matches, when MODE[5] is set.

At any time, the FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on A or B respectively. Note that FLAG bit is not set by the FORCMA and FORCMB operations.

If subsequent matches occur on comparators A and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

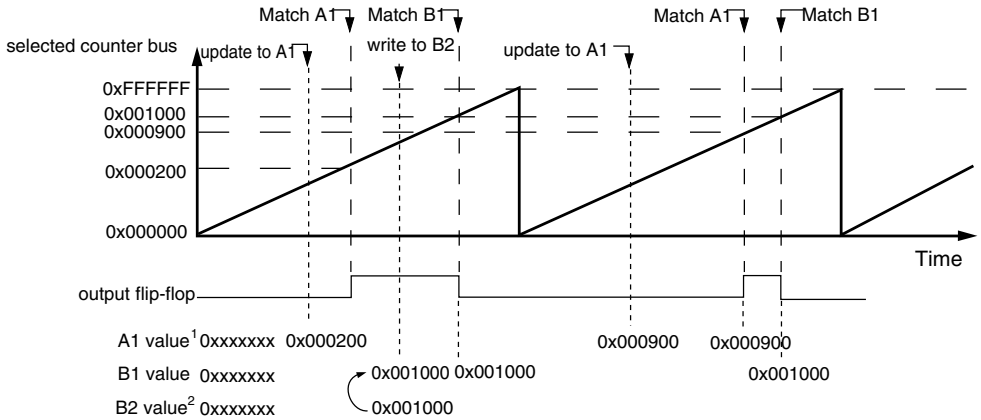
At OPWM mode entry the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC[n] register.

In order to achieve 100% duty cycle, both registers A1 and B1 must be set to the same value. When a simultaneous match on comparators A and B occur, the output flip-flop is set at every period to the value of EDPOL bit. 0% duty cycle is possible by writing 0x0 to register A. When a match occurs, the output flip-flop is set at every period to the complement of EDPOL bit. The transfer from register B2 to B1 is still controlled by the MODE[6] bit.

Note

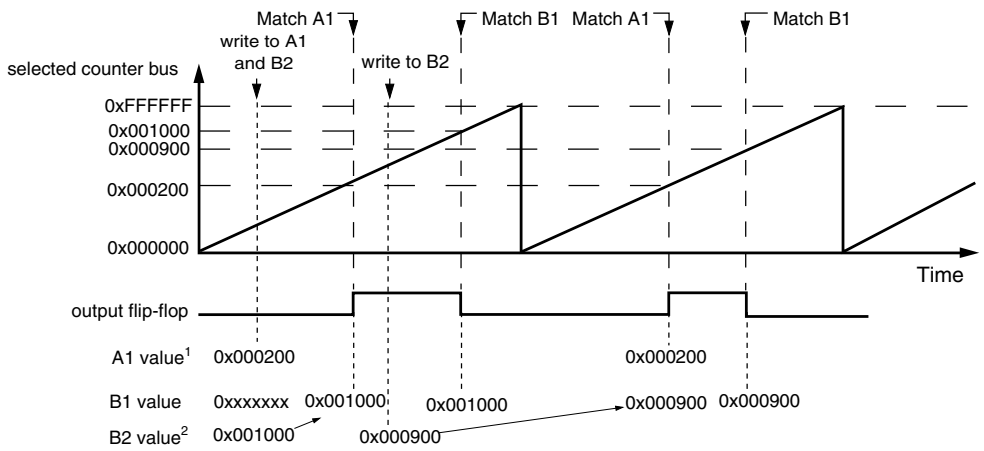
If A1 and B1 are set to the 0x0, a 0% duty cycle waveform is produced.

Figure 42-149 and Figure 42-150 show the Unified Channel running in OPWM with immediate update and next period update, respectively.



Notes:1. EMIOSA[n] = A1
2. EMIOSB[n] = B2
A2 = A1 according to OU[n] bit
B2 = B1 according to OU[n] bit

Figure 42-149. PWM with immediate update (MODE[6] is 0b)



Notes:1. EMIOSA[n] = A1
2. EMIOSB[n] = B2
A2 =A1 and A2= A1 according to OU[n] bit

Figure 42-150. PWM with next period update (MODE[6] is 1b)

42.6.1.1.18 Output Pulse Width Modulation Buffered (OPWMB) Mode

OPWMB mode (MODE[0:6]=11000b0) is used to generate pulses with programmable leading and trailing edge placement. An external counter driven in MCB Up mode must be selected from one of the counter buses. A1 register value defines the first edge and B1 the second edge. The output signal polarity is defined by the EDPOL bit. If EDPOL is zero, a negative edge occurs when A1 matches the selected counter bus and a positive edge occurs when B1 matches the selected counter bus.

The A1 and B1 registers are double buffered and updated from A2 and B2, respectively, at the cycle boundary. The load operation is similar to the OPWFMB mode. Please refer to Figure 42-140 for more information about A1 and B1 registers update.

FLAG can be generated at B1 matches, when MODE[5] is cleared, or in both A1 and B1 matches, when MODE[5] is set. If subsequent matches occur on comparators A and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on A1 or B1 respectively. The FLAG bit is not set by the FORCMA and FORCMB operations.

At OPWMB mode entry the output flip-flop is set to the value of the EDPOL bit in the EMIOSC[n] register.

Some rules applicable to the OPWMB mode are:

- B1 matches have precedence over A1 matches if they occur at the same time within the same counter cycle
- A1=0 match from cycle **n** has precedence over B1 match from cycle **n-1**
- A1 matches are masked out if they occur after B1 match within the same cycle
- Any value written to A2 or B2 on cycle **n** is loaded to A1 and B1 registers at the following cycle boundary (assuming OU[n] bit of EMIOSOUDIS register is not asserted). Thus the new values will be used for A1 and B1 matches in cycle **n+1**

The following figure describes the operation of the OPWMB mode regarding A1 and B1 matches and the transition of the channel output pin. In this example EDPOL is set to zero.

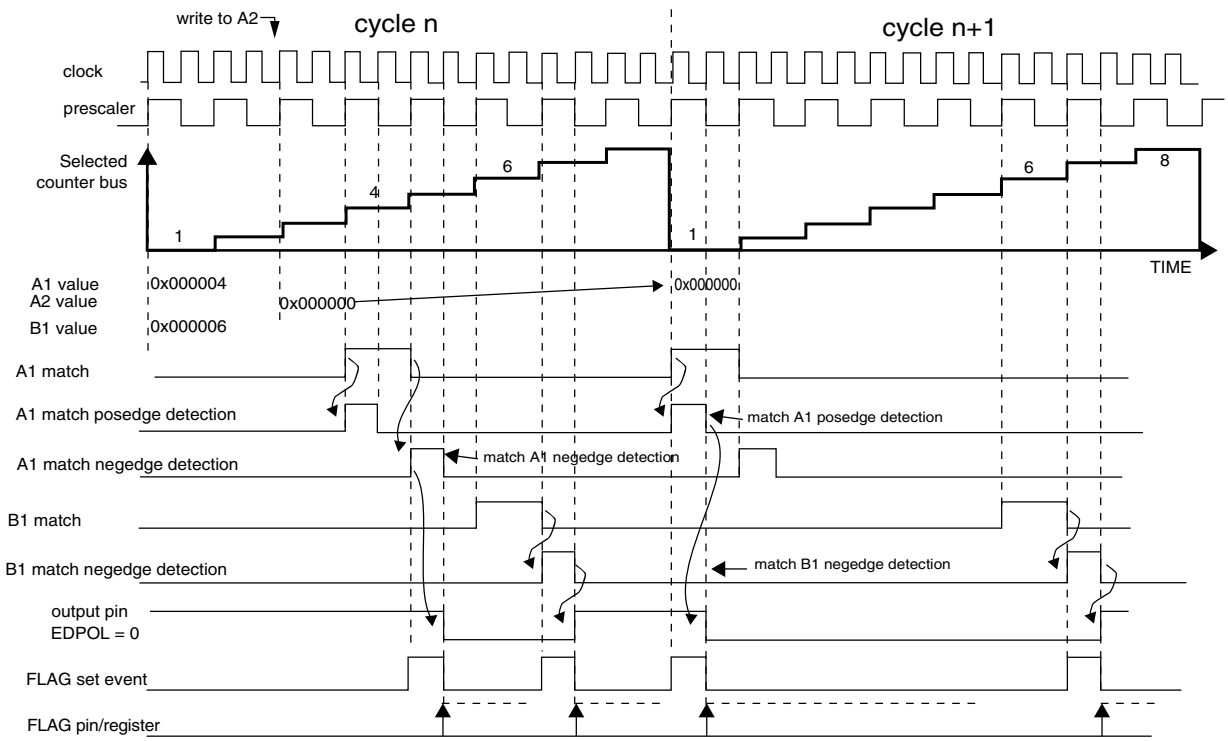


Figure 42-151. OPWMB Mode Matches and Flags

Note that the output pin transitions are based on the negedges of the A1 and B1 match signals. The figure above shows in cycle **n+1** the value of A1 register being set to zero. In this case the match posedge is used instead of the negedge to transition the output flip-flop.

The figure below describes the channel operation for 0% duty cycle. Note that the A1 match posedge signal occurs at the same time as the B1=0x8 negedge signal. In this case A1 match has precedence over B1 match, causing the output pin to remain at EDPOL bit value, thus generating a 0% duty cycle signal.

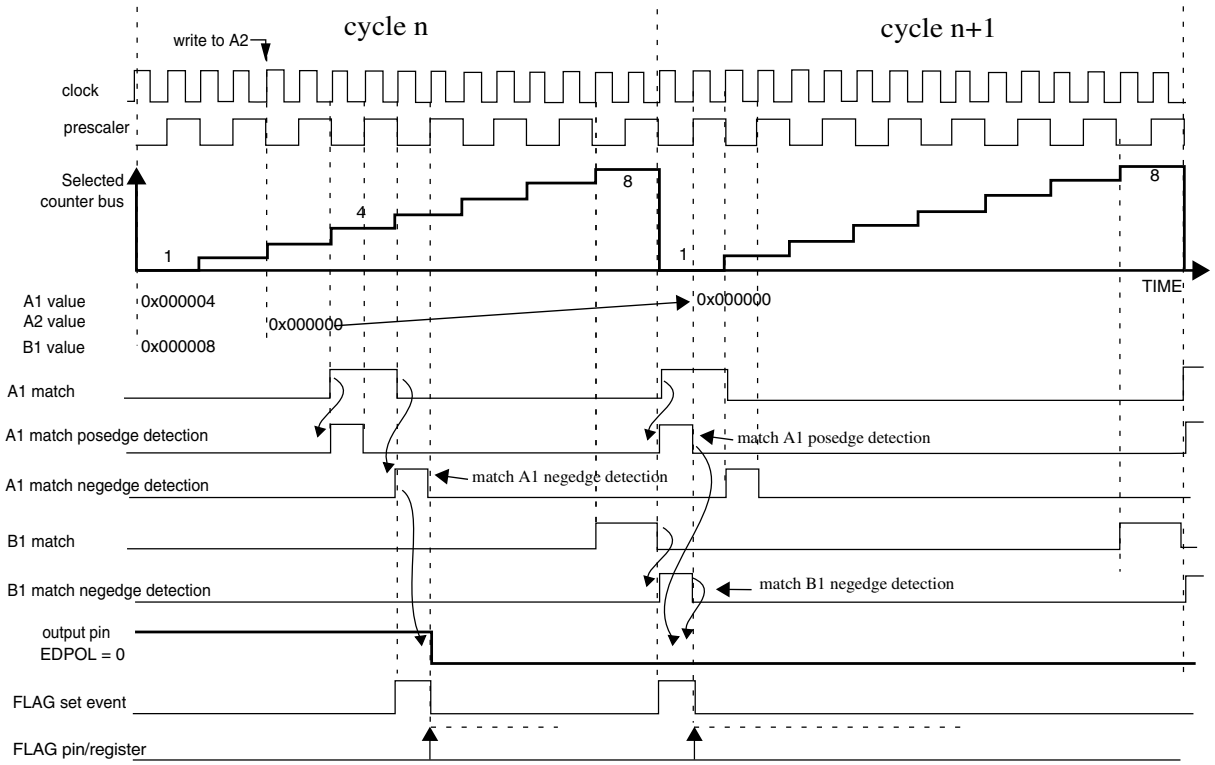


Figure 42-152. OPWMB Mode with 0% Duty Cycle

Figure 42-153 describes the operation of the OPWMB mode with the Output Disable signal being asserted. The Output Disable forces a transition in the output pin to the EDPOL bit value. After deasserted, the output disable allows the output pin to transition at the following A1 or B1 match. Note that the Output Disable does not modify the Flag bit behavior. Note that there is one system clock delay between the assertion of the output disable signal and the transition of the output pin to EDPOL.

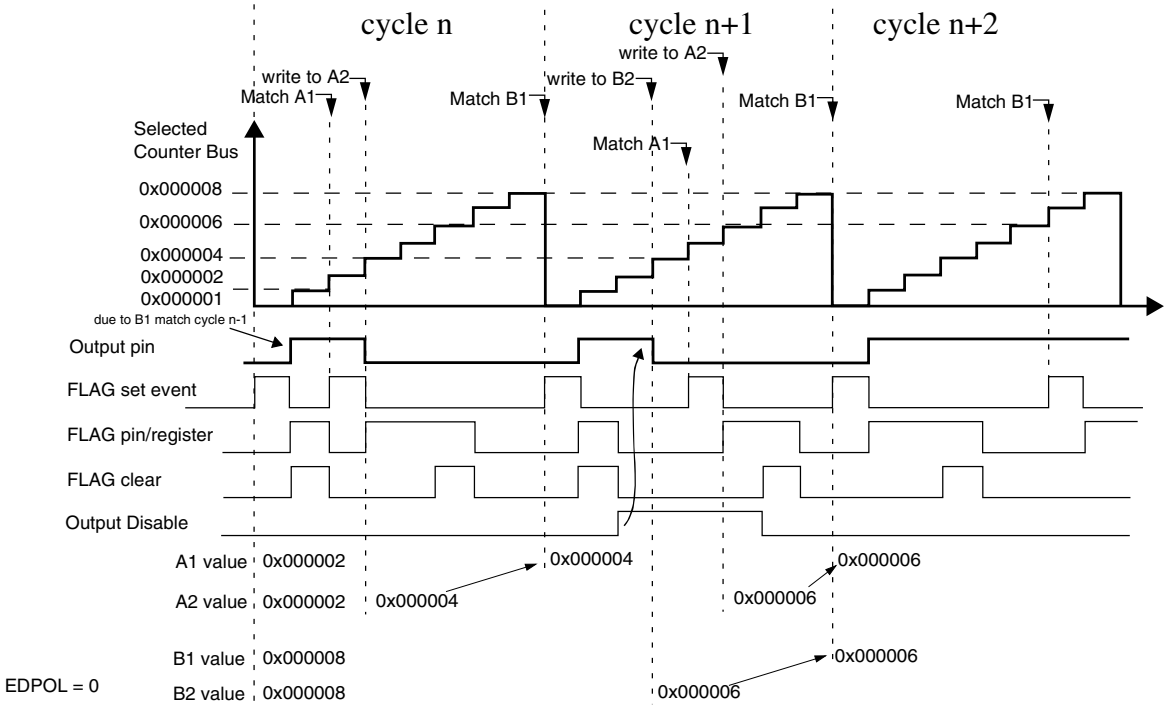


Figure 42-153. OPWMB Mode with Active Output Disable (MODE[6] bit is 1b)

Figure 42-154 shows a waveform changing from 100% to 0% duty cycle. EDPOL in this case is zero. In this example, B1 is programmed to the same value as the period of the external selected time base.

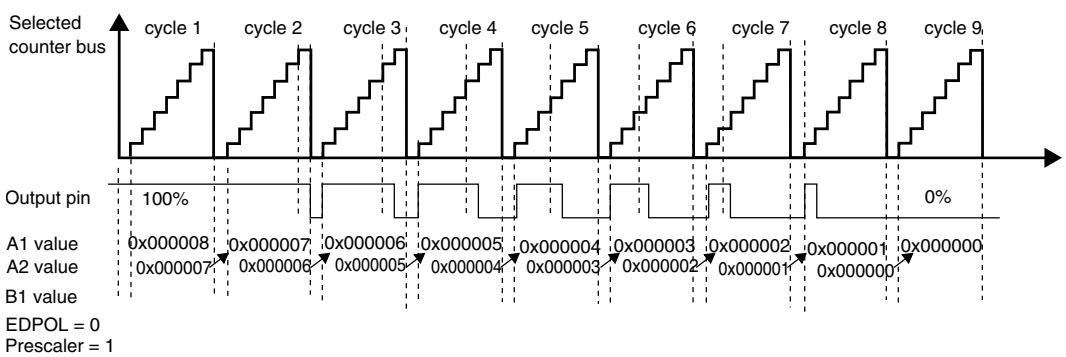


Figure 42-154. OPWMB Mode from 100% to 0% Duty Cycle

In Figure 42-154, if B1 is set to a value lower than 0x8, it is not possible to achieve 0% duty cycle by only changing A1 register value. Since B1 matches have precedence over A1 matches the output pin transitions to the opposite of EDPOL bit at B1 match. Note also that if B1 is set to 0x9, for instance, B1 match does not occur, thus a 0% duty cycle signal is generated.

42.6.1.1.19 Output Pulse Width Modulation with Trigger (OPWMT) Mode

OPWMT mode (MODE[0:6]=0100110) is intended to support the generation of Pulse Width Modulation signals where the period is not modified while the signal is being output, but where the duty cycle will be varied and must not create glitches. The mode is intended to be used in conjunction with other channels executing in the same mode and sharing a common timebase. It will support each channel with a fixed PWM leading edge position with respect to the other channels and the ability to generate a trigger signal at any point in the period that can be output from the module to initiate activity in other parts of the SoC such as starting ADC conversions.

An external counter driven in either MC Up or MCB Up mode must be selected from one of the counter buses.

Register A1 defines the leading edge of the PWM output pulse and as such the beginning of the PWM's period. This makes it possible to insure that the leading edge of multiple channels in OPWMT mode can occur at a specific time with respect to the other channels when using a shared timebase. This can allow the introduction of a fixed offset for each channel which can be particularly useful in the generation of lighting PWM control signals where it is desirable that edges are not coincident with each other to help eliminate noise generation. The value of register A1 represents the shift of the PWM channel with respect to the selected timebase. A1 can be configured with any value within the range of the selected time base. Note that registers loaded with 0x0 will not produce matches if the timebase is driven by a channel in MCB mode.

A1 is not buffered as the shift of a PWM channel must not be modified while the PWM signal is being generated. In case A1 is modified it is immediately updated and one PWM pulse could be lost.

EMIOSB[n] address gives access to B2 register for write and B1 register for read. Register B1 defines the trailing edge of the PWM output pulse and as such the duty cycle of the PWM signal. To synchronize B1 update with the PWM signal and so ensure a correct output pulse generation the transfer from B2 to B1 is done at every match of register A1. This behavior is the same as the OPWM mode with next period update.

EMIOSOUDIS register affects transfers between B2 and B1 only.

In order to account for the shift in the leading edge of the waveform defined by register A1 it will be necessary that the trailing edge, held in register B1, can roll over into the next period. This means that a match against the B1 register should not have to be qualified by a match in the A1 register. The impact of this would mean that incorrectly setting register B1 to a value less that register A1 will result in the output being held over a cycle boundary until the B1 value is encountered.

This mode provides a buffered update of the trailing edge by updating register B1 with register B2 contents only at a match of register A1.

The value loaded in register A1 is compared with the value on the selected time base. When a match on comparator A1 occurs, the output flip-flop is set to the value of the EDPOL bit. When a match occurs on comparator B, the output flip-flop is set to the complement of the EDPOL bit.

Note that the output pin and flag transitions are based on the posedges of the A1, B1 and A2 match signals. Please refer to Figure 42-151 in Output Pulse Width Modulation Buffered (OPWMB) Mode for details on match posedge.

Register A2 defines the generation of a trigger event within the PWM period and A2 should be configured with any value within the range of the selected time base, otherwise no trigger will be generated. A match on the comparator will generate the FLAG signal but it has no effect on the PWM output signal generation. The typical setup to obtain a trigger with FLAG is enabling DMA and driving the channel's ipd_done input high.

A2 is not buffered and therefore its update is immediate. If the channel is running when a change is made this could cause either the loss of one trigger event or the generation of two trigger events within the same period. Register A2 can be accessed by reading or writing the eMIOS200 UC Alternate A Register (eMIOS_ALTA) at UC[n] base address +0x14.

The FLAG signal is set only at match on the comparator with A2. A match on the comparator with A1 or B1 or B2 has no effect on FLAG.

At any time, the FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on A or B respectively. Any FORCMA and/or FORCMB has priority over any simultaneous match regarding to output pin transitions. Note that the load of B2 content on B1 register at an A match is not inhibited due to a simultaneous FORCMA/FORCMB assertion. If both FORCMA and FORCMB are asserted simultaneously the output pin goes to the opposite of EDPOL value such as if A1 and B1 registers had the same value. FORCMA assertion causes the transfer from register B2 to B1 such as a regular A match, regardless of FORCMB assertion.

If subsequent matches occur on comparators A1 and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

At OPWMT mode entry the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC[n] register.

In order to achieve 0% duty cycle both registers A1 and B must be set to the same value. When a simultaneous match on comparators A and B occur, the output flip-flop is set at every period to the complement value of EDPOL.

In order to achieve 100% duty cycle the register B1 must be set to a value greater than maximum value of the selected time base. As a consequence if 100% duty cycle must be implemented the maximum counter value for the time base is 0xFFFFFE for a 24-bit counter and 0xFFFE for a 16-bit counter, respectively. When a match on comparator A1 occurs the output flip-flop is set at every period to the value of EDPOL bit. The transfer from register B2 to B1 is still triggered by the match at comparator A.

As with other eMIOS200 mode, the OPWMT mode will implement the Output Disable function. Setting the ODIS bit in the eMIOS200 UC Control Register (EMIOSC[n]) will enable the Output Disable function. If the selected Output Disable input signal is asserted for the channel, the output pin will go to the inverse of the EDPOL. The channel will continue to operate normally, although the output will be fixed. When the Output Disable input signal is negated, the output pin will return to operate as normal.

Figure 42-155 shows the Unified Channel running in OPWMT mode with Trigger Event Generation and duty cycle update on next period update.

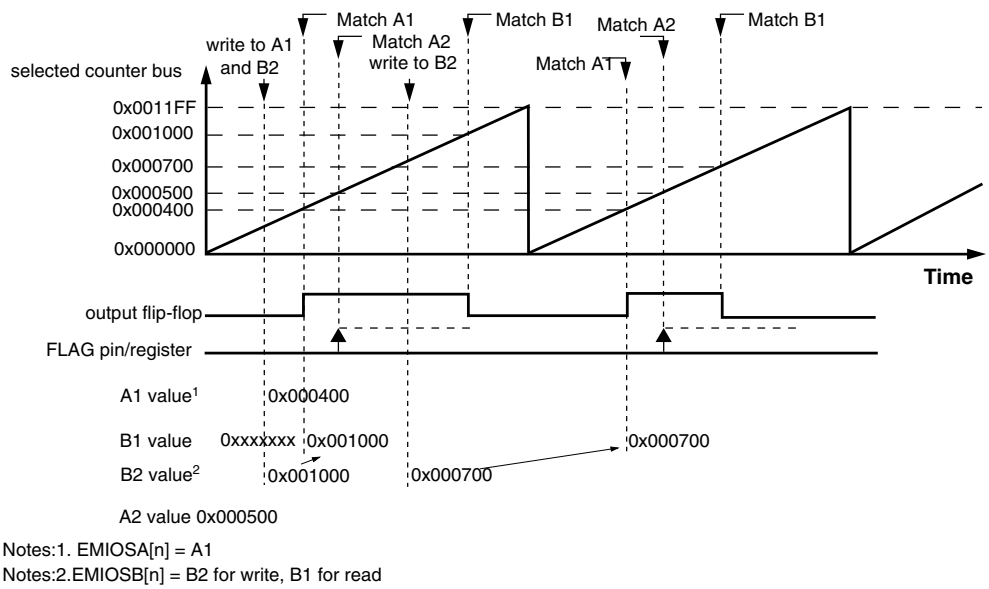


Figure 42-155. OPWMT example

Figure 42-156 shows the Unified Channel running in OPWMT mode with Trigger Event Generation and 0% duty.

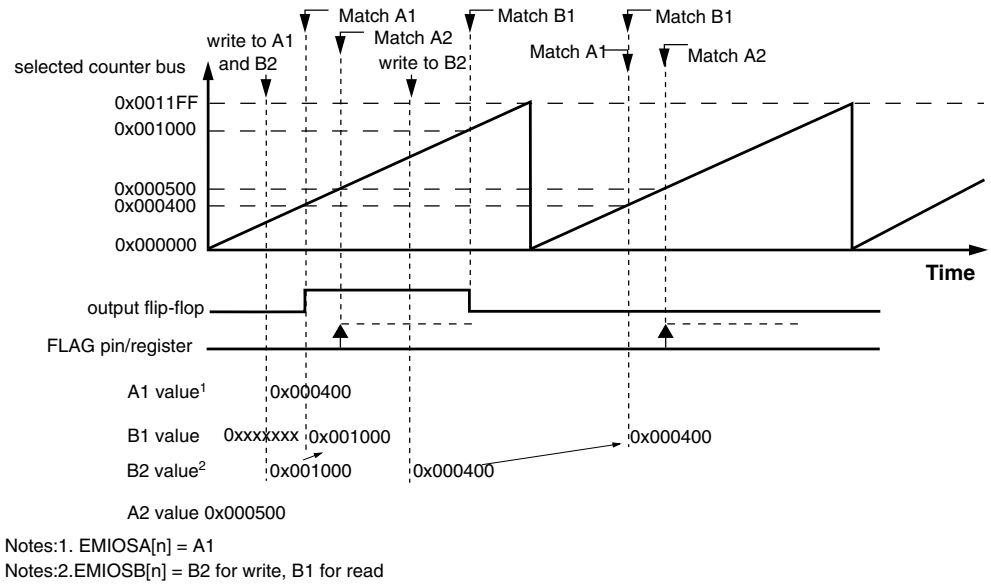


Figure 42-156. OPWMT with 0% Duty Cycle

Figure 42-157 shows the Unified Channel running in OPWMT mode with Trigger Event Generation and 100% duty cycle.

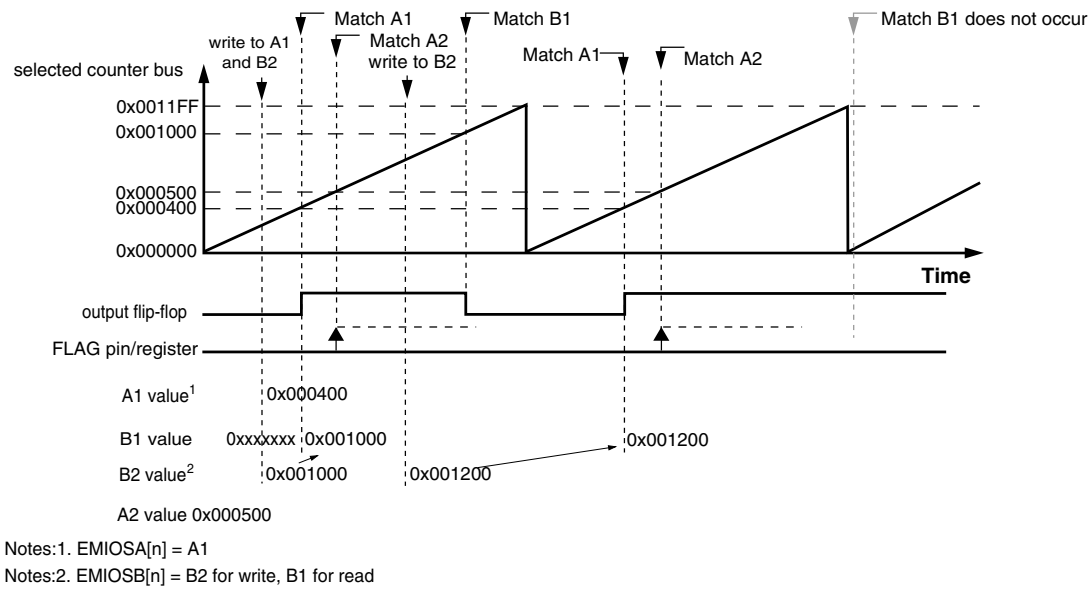


Figure 42-157. OPWMT with 100% Duty Cycle

42.6.1.2 Input Programmable Filter (IPF)

The IPF ensures that only valid input pin transitions are received by the Unified Channel edge detector. A block diagram of the IPF is shown in Figure 42-158.

The IPF is a 5-bit programmable up counter that is incremented by the selected clock source, according to bits IF[0:3] in EMIOSC[n] register.

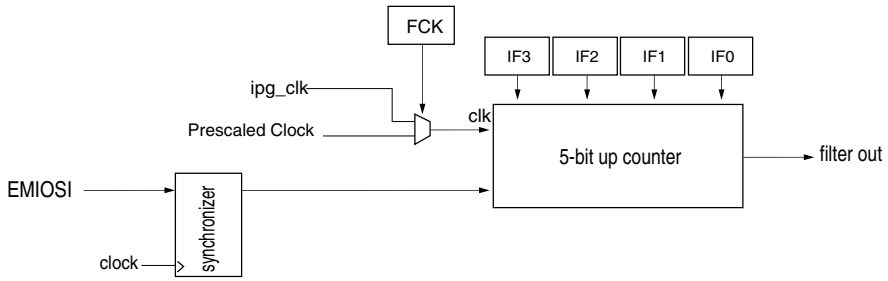


Figure 42-158. Input Programmable Filter Submodule Diagram

The input signal is synchronized by system clock . When a state change occurs in this signal, the 5-bit counter starts counting up. As long as the new state is stable on the pin, the counter remains incrementing. If a counter overflow occurs, the new pin value is validated. In this case, it is transmitted as a pulse edge to the edge detector. If the opposite edge appears on the pin before validation (overflow), the counter is reset. At the next pin transition, the counter starts counting again. Any pulse that is shorter than a full range of the masked counter is regarded as a glitch and it is not passed on to the edge detector. A timing diagram of the input filter is shown in Figure 42-159.

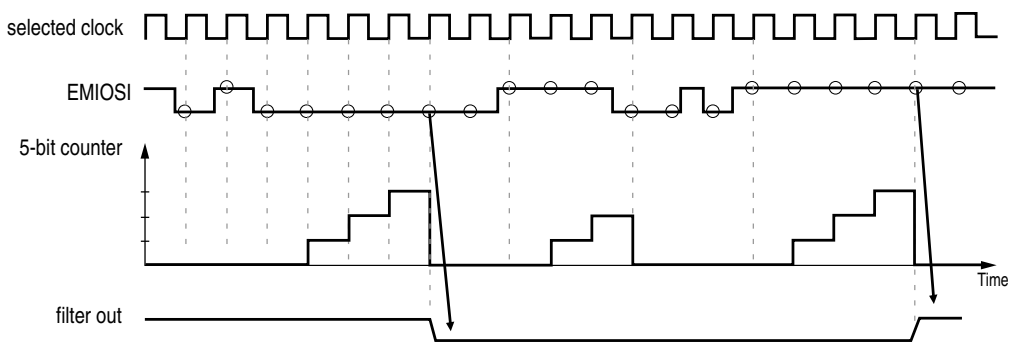


Figure 42-159. Input Programmable Filter Example (IF[0:3] is 0010b)

The filter is not disabled during either freeze state or negated GTBE input

42.6.1.3 Clock Prescaler (CP)

The CP divides the GCP output signal to generate a clock enable for the internal counter of the Unified Channels. The GCP output signal is prescaled by the value defined in the UCPRE[0:1] field description in the EMIOSC[n] register. The prescaler is enabled by setting the UCPREN bit in the EMIOSC[n] and can be stopped at any time by clearing this bit, thereby stopping the internal counter in the Unified Channel.

In order to ensure safe working and avoid glitches the following steps must be performed whenever any update in the prescaling rate is desired:

1. Write 0 at both GPREN bit in EMIOSMCR register and UCPREN bit in EMIOSC[n] register, thus disabling prescalers;
2. Write the desired value for prescaling rate at UCPRE[0:1] bits in EMIOSC[n] register;
3. Enable channel prescaler by writing 1 at UCPREN bit in EMIOSC[n] register;
4. Enable global prescaler by writing 1 at GPREN bit in EMIOSMCR register.

The prescaler is not disabled during either freeze state or negated GTBE input.

42.6.1.4 Effect of Freeze on the Unified Channel

When in debug mode, FRZ bit in the EMIOSMCR register and the FREN bit in the EMIOSC[n] are both set, the internal counter and Unified Channel capture and compare functions are halted. The UC is frozen in its current state.

During freeze, all registers are accessible. When the Unified Channel is operating in an output mode, the force match functions remain available, allowing the software to force the output to the desired level.

Note that for input modes, any input events that may occur while the channel is frozen are ignored.

When exiting debug mode or freeze enable bit is cleared (FRZ in the EMIOSMCR or FREN in the EMIOSC[n] register) the channel actions resume, but may be inconsistent until channel enters GPIO mode again.

42.6.2 IP Bus Interface Unit (BIU)

The BIU provides the interface between the Internal Interface Bus (IIB) and the Peripheral Bus, allowing communication among all submodules and this IP interface.

The BIU allows 8, 16 and 32 bits access. They are performed over a 32-bit data bus in a single cycle clock.

42.6.2.1 Effect of Freeze on the BIU

When the FRZ bit in the EMIOSMCR register is set and the module is in debug mode, the operation of BIU is not affected.

42.6.3 STAC Bus Client submodule

The STAC bus client provides one external time base, imported from the STAC bus, to the Unified Channels. Figure 42-160 provides a block diagram for the module.

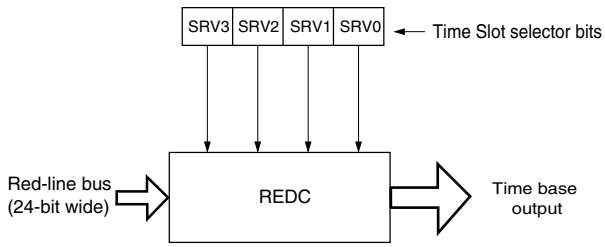


Figure 42-160. Block diagram

Bits SRV[0:3] in register EMIOSMCR selects the desired time slot of the STAC bus to be output. Figure 42-161 shows a timing diagram for the REDC.

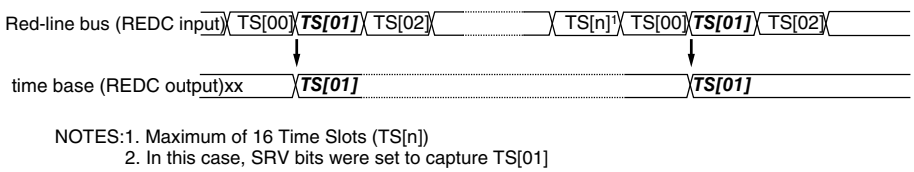


Figure 42-161. Timing diagram for the STAC bus and STAC output

Every time the selected time slot change, the STAC output is updated.

See the chip-specific eMIOS information for the STAC server IDs.

42.6.3.1 Effect of Freeze on the STAC bus

When the FRZ bit in the EMIOS_MCR register is set and the module is in debug mode, the operation of REDC submodule is not affected, i.e., there is no freeze function in this submodule.

42.6.4 Global Clock Prescaler Submodule (GCP)

The GCP divides the system clock to generate a clock for the CPs of the channels. The main clock signal is prescaled by the value defined in the Global Prescaler clock divider table within the eMIOS200 Module Configuration Register (EMIOSMCR) description

according to the GPRE[0:7] bits in EMIOSMCR register. The global prescaler is enabled by setting the GPREN bit in the EMIOSMCR register and can be stopped at any time by clearing this bit, thereby stopping the internal counters in all the channels.

In order to ensure safe working and avoid glitches the following steps must be performed whenever any update in the prescaling rate is desired:

1. Write 0 at GPREN bit in EMIOSMCR register, thus disabling global prescaler;
2. Write the desired value for prescaling rate at GPRE[0:7] bits in EMIOSMCR register;
3. Enable global prescaler by writing 1 at GPREN bit in EMIOSMCR register.

The prescaler is not disabled during either freeze state or negated GTBE input.

42.6.4.1 Effect of Freeze on the GCP

When the FRZ bit in the EMIOSMCR register is set and the module is in debug mode, the operation of GCP submodule is not affected, i.e., there is no freeze function in this submodule.

42.7 Initialization/Application Information

On resetting the eMIOS200, the Unified Channels enter GPIO input mode.

42.7.1 Considerations

Before changing an operating mode, the UC must be programmed to GPIO mode and EMIOSA[n] and EMIOSB[n] registers must be updated with the correct values for the next operating mode. Then the EMIOSC[n] register can be written with the new operating mode. If a UC is changed from one mode to another without performing this procedure, the first operation cycle of the selected time base can be random, i.e., matches can occur in random time if the contents of EMIOSA[n] or EMIOSB[n] were not updated with the correct value before the time base matches the previous contents of EMIOSA[n] or EMIOSB[n].

When interrupts are enabled, the software must clear the FLAG bits before exiting the interrupt service routine.

42.7.2 Application Information

Correlated output signals can be generated by all output operation modes. Bits OU[n] of eMIOS200_OUDIS register can be used to control the update of these output signals.

In order to guarantee that the internal counters of correlated channels are incremented in the same clock cycle, the internal prescalers must be set up before enabling the global prescaler. If the internal prescalers are set after enabling the global prescaler, the internal counters may increment in the same ratio, but at a different clock cycle.

It is recommended to drive Output Disable Input signals with the emios_flag_out signals of some UCs running in SAIC mode. When an output disable condition happens, the software interrupt routine must service the output channels before servicing the channels running SAIC. This procedure avoid glitches in the output pins.

42.7.2.1 Time Base Generation

For MC and OPWFM with internal clock source operation modes, the internal counter rate can be modified by configuring the clock prescaler ratio. Figure 42-162 shows an example of a time base with prescaler ratio equal to one.

Note

MCB and OPWFMB modes have a different behavior.

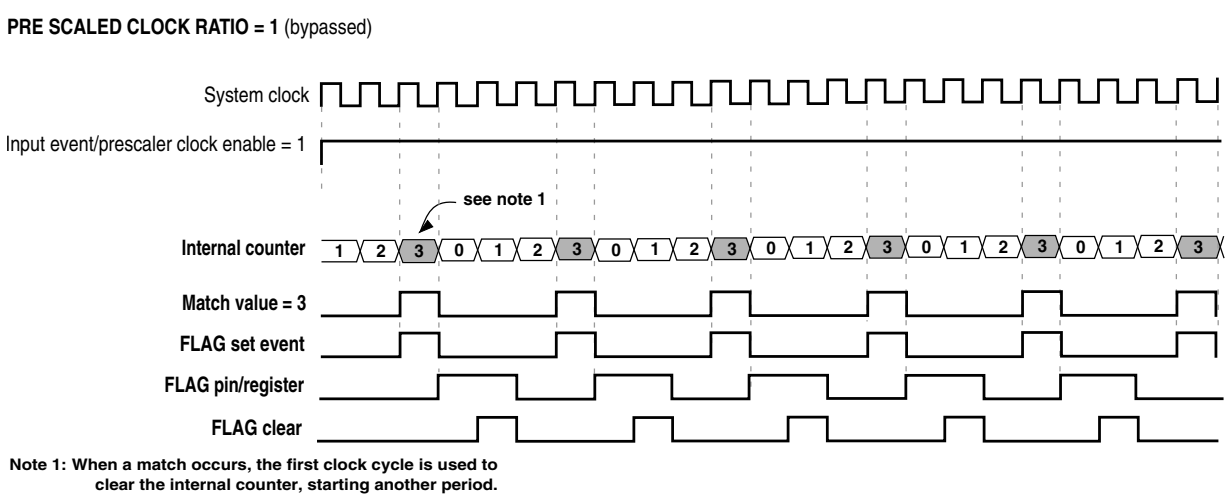


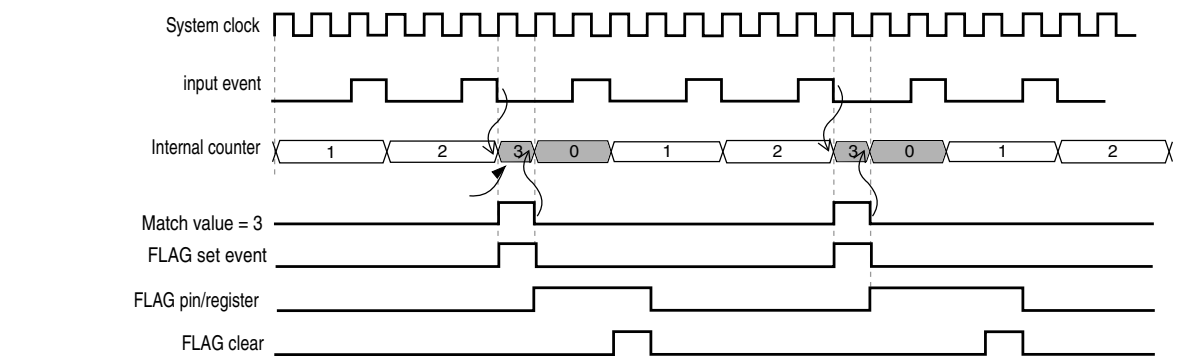
Figure 42-162. Time base period when running in the fastest prescaler ratio

If the prescaler ratio is greater than one or external clock is selected, the counter may behave in one of four different ways depending on the channel mode:

- If MC mode and Clear on Match Start and External Clock source are selected the internal counter behaves as described in Figure 42-163 .
- If MC mode and Clear on Match Start and Internal Clock source are selected the internal counter behaves as described in Figure 42-164.
- If MC mode and Clear on Match End are selected the internal counter behaves as described in Figure 42-165.
- If OPWFM mode is selected the internal counter behaves as described in Figure 42-164. The internal counter clears at the start of the match signal, skips the next prescaled clock edge and then increments in the subsequent prescaled clock edge.

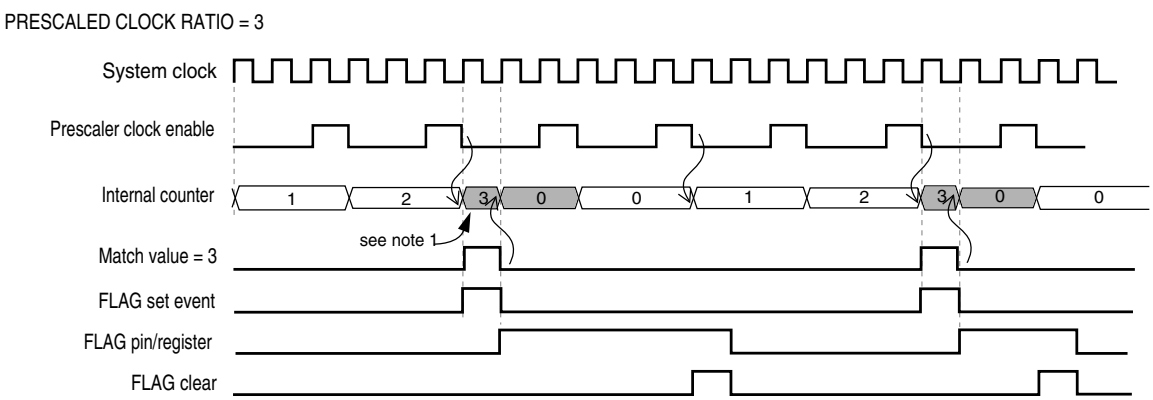
Note

MCB and OPWFMB modes have a different behavior.



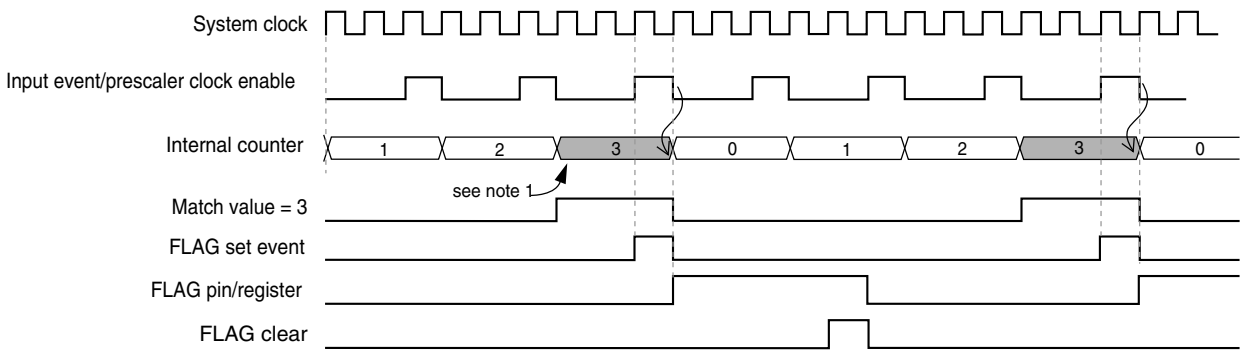
Note 1: When a match occurs, the first system clock cycle is used to clear the internal counter, and at the next edge of prescaler clock enable the counter will start counting.

Figure 42-163. Time base generation with external clock and clear on match start



Note 1: When a match occurs, the first clock cycle is used to clear the internal counter, and only after a second edge of pre scaled clock the counter will start counting.

Figure 42-164. Time base generation with internal clock and clear on match start



Note 1: The match occurs only when the input event/prescaler clock enable is active. Then, the internal counter is immediately cleared.

Figure 42-165. Time base generation with clear on match end

42.7.2.2 Coherent Accesses

Reading the EMIOSA[n] register again in the same period of the last read of EMIOSB[n] register may lead to incoherent results. This will occur if the last read of EMIOSB[n] register occurred after a disabled B2 to B1 transfer.

42.7.2.3 Channel/Modes Initialization

The following basic steps summarize basic output mode startup, assuming the channels are initially in GPIO mode:

1. *[global]* Disable Global Prescaler.
2. *[timebase channel]* Disable Channel Prescaler.
3. *[timebase channel]* Write initial value at internal counter.
4. *[timebase channel]* Set A/B register.
5. *[timebase channel]* Set channel to MC(B) Up mode.
6. *[timebase channel]* Set prescaler ratio.
7. *[timebase channel]* Enable Channel Prescaler.
8. *[output channel]* Disable Channel Prescaler.
9. *[output channel]* Set A/B register.
10. *[output channel]* Select timebase input through BSL[1:0] bits.
11. *[output channel]* Enter output mode.

12. *[output channel]* Set prescaler ratio (same ratio as timebase channel).
13. *[output channel]* Enable Channel Prescaler.
14. *[global]* Enable Global Prescaler.
15. *[global]* Enable Global Time Base.

The timebase channel and the output channel may be the same for some applications such as in OPWFM(B) mode or whenever the output channel is intended to run the timebase itself.

At any time the flags can be configured.

Chapter 43
Enhanced Time Processing Unit (eTPU)

43.1 Enhanced Time Processing Unit (eTPU) chip-specific module information

43.1.1 Configuration

eTPU is an intelligent, semi-autonomous co-processor designed for timing control. Operating in parallel with the Host CPU, the eTPU processes instructions, real-time input events, performs output waveform generation, and accesses shared data without Host intervention. Consequently, for each timer event, the Host CPU setup and service times are minimized or eliminated.

The eTPU2+ is an enhanced version of the eTPU2 targeted at C55 devices.

43.1.2 Device-specific information

- Dual (64 channels, dual core) Engine
- Shared Code Memory (SCM) error detection, error correction
- Shared Data Memory (SDM) error detection, error correction
- Class 3 Nexus support
- GTBE is a logical OR with the eMIOS GTBE
- Channels used to trigger the ADC subsystem
- Channels used to enable the Reaction Module modulation
- Channels with DMA request
- Channel output buffers are controlled through SIUL2, and cannot be controlled with microcode.
- Six latency monitors in each of 2 engines.

43.1.3 Register reset values

Some eTPU registers on this device have a different reset value, which overrides the corresponding reset value that is shown in the body of the eTPU documentation, as shown in the following table.

Table 43-1. Reset value overrides

Register	Reset value override
eTPU System Configuration Registers_MCR	000B_0000h
eTPU System Configuration Registers_ECR_ENG1	5000_0000h
eTPU System Configuration Registers_ECR_ENG2	5000_0000h
Time Base Registers_TBCR_ENG1	2000_0000h
Time Base Registers_TBCR_ENG2	2000_0000h
Time Base Registers_STAC_ENG1	0000_0200h
Time Base Registers_STAC_ENG2	0100_0300h

43.1.4 Memory

The eTPU is a true Harvard architecture machine that implements separate code and data memories. The eTPU engines share the same code and data memories. To meet the safety goals, the memories have ECC.

Memory characteristics:

- Code memory 24k with ECC. The code memory can only be accessed with 32-bit accesses so only 7 bits of ECC are implemented. The module size is 6144 locations x 39-bits wide.
- Data memory 6k with ECC. The data memory can be accessed as bytes, and cycles for updating the ECC are not allowed, so 20 bits of ECC are implemented. The required module size is 1536 locations x 52 bits wide.

43.1.5 Clocking

The eTPU is clocked by the non-FM clock used for the peripherals.

The eTPU BIU can optionally run at double the clock speed of the other peripheral on the AIPS. Double clock speed can be used in cases where higher eTPU processing throughput is needed.

43.1.6 Debug

The eTPU implements Nexus class 3 debug, accessed through the same Nexus debug port pins used for core debug.

43.1.7 STAC bus

The eTPU timebases and angle counter can be exported to eMIOS through the STAC bus.

Table 43-2. STAC server ID

Engine	Timebase	Server ID
1	TCR1	0
1	TCR2	2
2	TCR1	1
2	TCR2	3

NOTE

If the STAC-bus Export/Import feature is used, the clock ratio of eTPU : eMIOS should be 1:1 or 2:1, such as the following:

- eTPU 100MHz : eMIOS 100MHz
- eTPU 200MHz : eMIOS 100MHz

This applies whether the STAC-bus Export/Import is used from eTPU to eMIOS or used between eTPUs.

43.1.8 Input connections

The channel inputs are connected to:

- External pins,
- SD ADC trigger outputs,
- BCTU trigger outputs, and the
- Input glitch filter.

Channels with more than 1 input source are connected through the IOMUX (see SIUL2 chapter).

The TCRCLK input is connected to the same input as channel 0.

43.1.9 Output connections

The channel outputs are connected to:
- External pins
- ADC triggers
- Decimation filters

43.2 Introduction

eTPU is an intelligent, semi-autonomous co-processor designed for timing control. Operating in parallel with the Host CPU, the eTPU processes instructions and real-time input events, performs output waveform generation, and accesses shared data without Host intervention. Consequently, for each timer event, the Host CPU setup and service times are minimized or eliminated.

High-level assembler, compiler and documentation allows customers to develop their own functions on the eTPU.

eTPU architecture aims at high resolution timing capabilities. From a system perspective, high resolution timing is limited by Host CPU overhead required for servicing timing tasks such as period measurement, pulse measurement, pulse width modulated waveform generation, etc. On the eTPU, high resolution timing is achieved by three main capabilities:

- Reduced latency: pin actions are immediate.
- Reduce or eliminate host interrupt service time.
- Double action channel capability reducing the channel request rate.

eTPU provides higher resolution than the Host CPU can achieve and creates no Host overhead for servicing timing tasks.

Latency is the interval from occurrence of an event to the start of event servicing. eTPU can service its own events without interrupting the Host. There are two types of timing events:

- Input pin transition.
- Selected Time Base match, i.e., a selected Time Base counter reached or exceeded a pre-programmed value.

Service time is the time spent servicing an event. In general, in microcontrollers the service time is constrained because the instruction set is not optimized for time function synthesis. The eTPU instruction set is optimized, so that time functions can be implemented with much fewer instructions than the Host CPU. Instructions execute faster, service time is reduced and program memory compacted.

Instructions executed by the eTPU are connected directly to the eTPU timing hardware and allow parallelism of hardware related actions.

43.2.1 Overview

The figure below shows a top-level eTPU block diagram. It displays a dual eTPU Engine configuration.

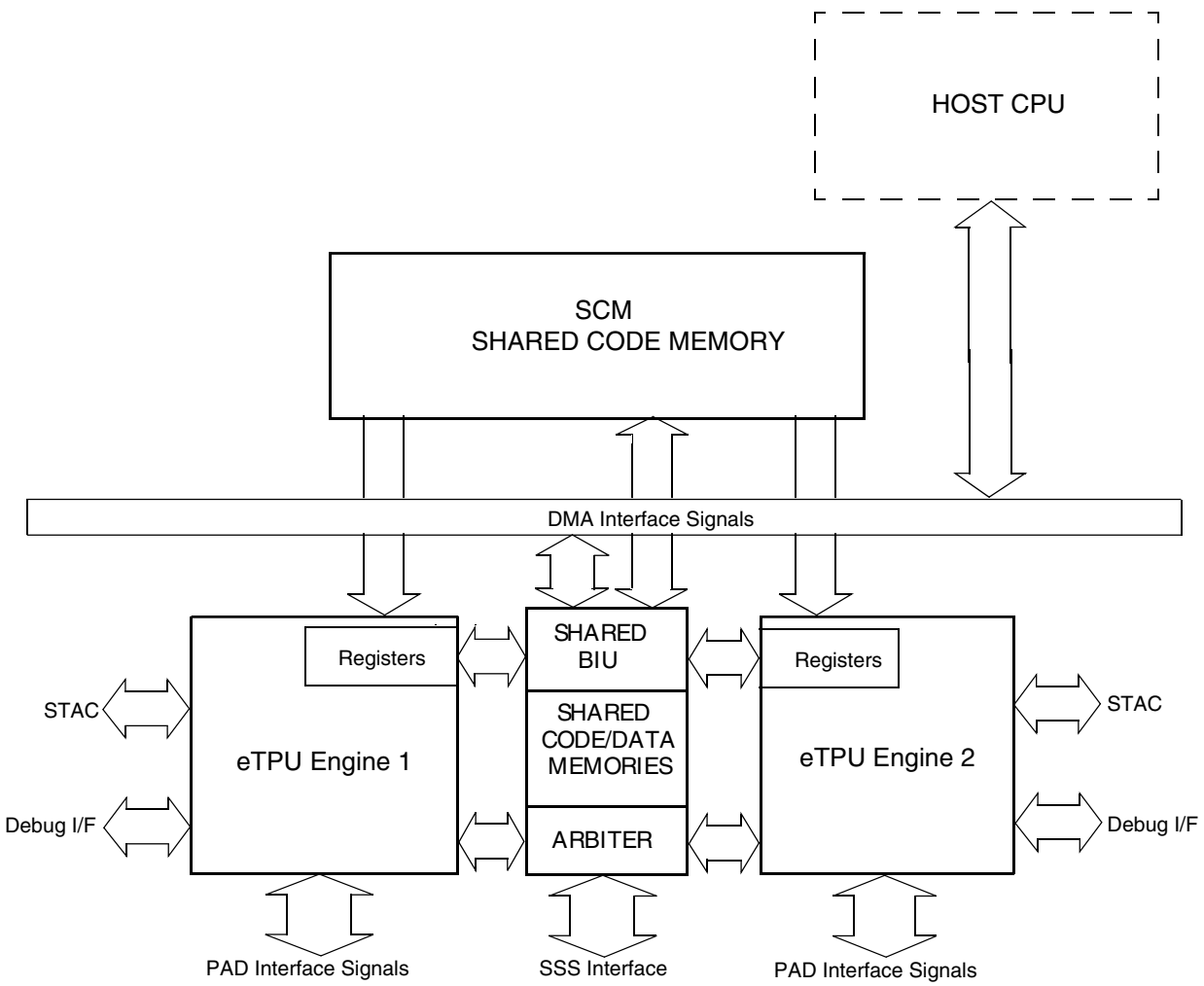


Figure 43-1. eTPU Block Diagram

eTPU Engine is responsible for processing input pin transitions and output pin waveform generation based on the **Time Bases**. Each eTPU Engine has its own microprocessor and dedicated hardware for processing signals on I/O pins and can also interface with external time bases through the STAC bus.

Both eTPU Engine CPUs, hereafter called **microengines**, fetch microinstructions from a **Shared Code Memory - SCM**.

Shared Data Memory - **SDM -** holds eTPU application parameters and work data. It is accessed by Host and both microengines.

Bus Interface Unit - BIU - allows Host to access eTPU registers, SCM and SDM.

Each I/O signal pair is associated with a dedicated **Channel**, which provides hardware for input signal processing and output signal generation, in relationship with selected Time Bases.

The eTPU, as a microprocessed subsystem, works much like a typical real-time system: it runs microengine code from instruction memory (SCM) to handle specific events, accessing data memory (SDM) for parameters, work data and application state info; events may originate from I/O Channels (due to pin transitions and/or time base matches), Host CPU requests or inter-channel requests; events that call for local eTPU processing activate the microengine by issuing a **Service Request**. The Service Request microcode may set an interrupt to the Host CPU. I/O channel events cannot directly interrupt the Host CPU.

Each Channel is associated with a **Function**, which defines its behavior: the Function is a software entity consisting, within the eTPU, of a set of microengine routines that attend to Service Requests. The Function routines are also responsible for Channel configuration. Function routines reside in SCM, which may contain several Functions. A Function may be assigned to several Channels, but a Channel can be associated with just one Function at a given moment. The association between Functions and Channels is defined by Host CPU, and is explained in detail in Functions and Threads.

eTPU hardware supplies resource sharing features that support concurrency:

- A hardware **Scheduler** dispatches the Service Request microengine routines based on a set of priorities defined by the Host CPU. Each Channel has its associated priority.
- A Service Request routine cannot be interrupted until it ends. This sequence of uninterrupted instruction execution is called a **Thread**.

- Channel-specific context (registers and flags) is automatically switched between the end of a Thread and the beginning of the next one.
- SDM arbitration, a dual-parameter coherency controller and semaphores can be used to ensure coherent access to eTPU data shared by both eTPU Engines and Host CPU.

43.2.1.1 eTPU Engine

Each eTPU Engine consists of all blocks that are duplicated in a dual eTPU configuration: two 24-bit time bases, 32 independent timer channels, a task scheduler, a microengine, and a Host interface. In addition, a 32-bit Shared Data Memory (SDM) is used for two eTPU Engines data storage and for passing information between the eTPU Engines and the Host CPU.

The following figure shows the block diagram for the eTPU Engine.

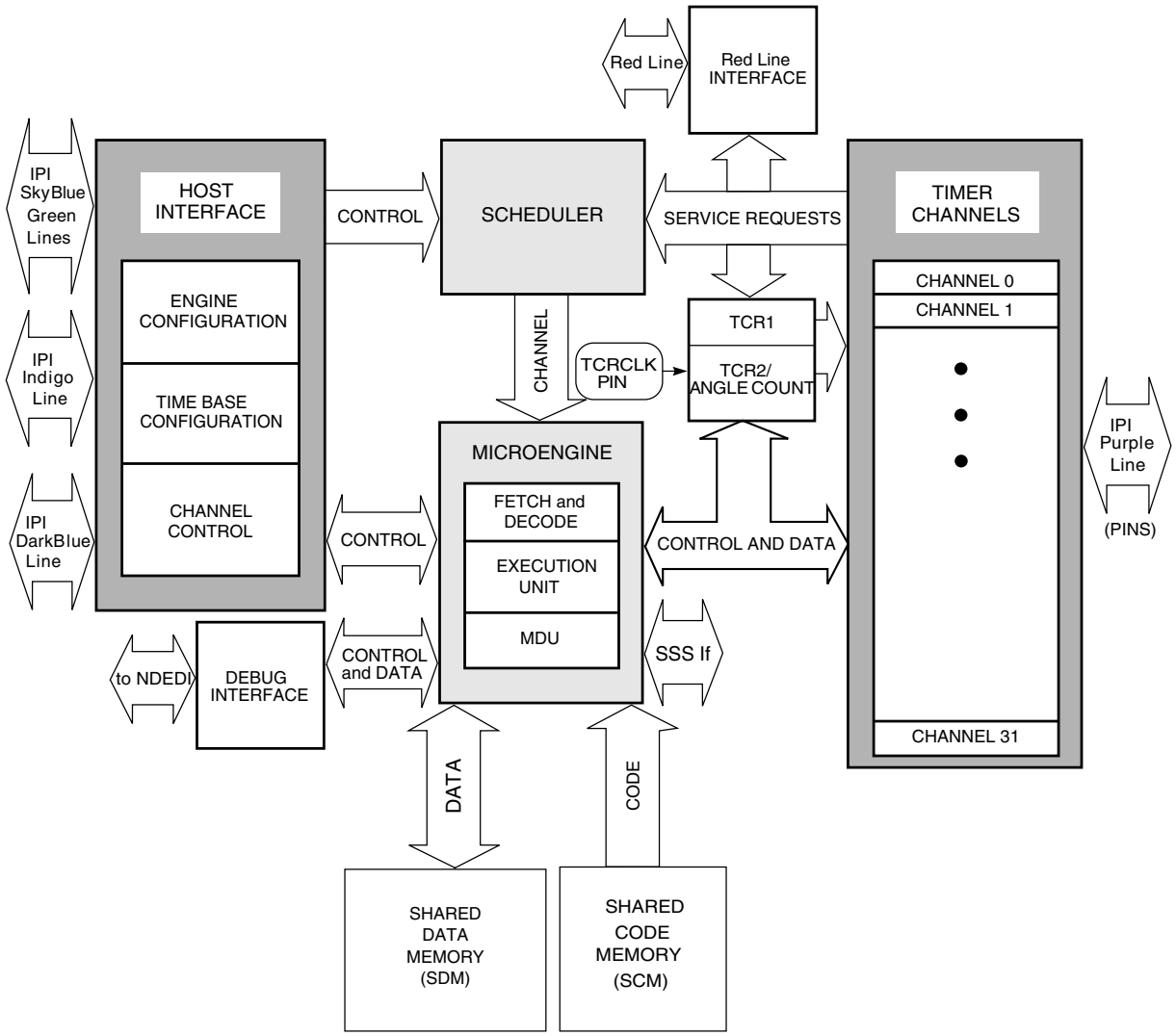


Figure 43-2. eTPU Engine Block Diagram

eTPU Engines 1 and 2 are sometimes called eTPU 1 and eTPU 2 for short throughout this chapter.

43.2.1.1.1 Time Bases

Two 24-bit counters TCR1 and TCR2 provide reference time bases for all match and input capture events. Prescalers for both time bases are controlled by the Host CPU through bit fields in the eTPU Engine configuration registers. The eTPU is able to export/import time to/from TCR1 or TCR2 in accordance to the STAC bus specification.

The clock for each of TCR1 and TCR2 clock can be independently derived from the eTPU clock or from an external input via the TCRCLK clock pin. In addition, the TCR2 timebase can be derived from special angle-clock hardware which enables implementing angle-based functions. This feature is added to support advanced angle based engine control applications.

For further details refer to Time Bases.

43.2.1.1.2 eTPU Timer Channels

Each eTPU Engine has 32 independent channels, each corresponding to an Input/Output signal pair. The channel time resolution is 24 bits, and are all identical.

Each channel consists of logic which supports two events and output controls. The event logic contains two 24-bit capture registers, two 24-bit match registers, greater-equal and equal-only comparators. Supporting two events enables many combinations of double action functions (for example the channel can handle two events with a single microcode service).

The channel configuration can be changed immediately by the microengine. Each channel can perform double capture, double match and other capture-match combinations. Channel modes available can do ordered or unordered match. Some modes are also provided that can block one match by the occurrence of the other. Service request can be generated on one or both of the match events.

Input signal can be separated from output signal in each channel. They can, optionally, be combined in a single I/O pin driver. An output buffer enable is controlled by the SIU (or SIUL). Digital filters are provided for the input signals, with distinct filtering modes available.

Each channel can use any time base or angle counter for either match or capture operation. For example, a match on TCR1 can capture the value of TCR2. The channels can request service from the microengine due to recognized pin transitions (input events) or timebase matches.

The eTPU channels also support the basic single-action operations found on TPU3 functionality with the exception that time resolution is 24-bits.

Channel configuration combinations:

- Single input capture, no match (TPU3 functionality)
- Single input capture with single match timeout (TPU3 functionality)
- Single input capture with double match timeout with several double match sub-modes
- Double input capture with single or double match timeout with several double match sub-modes
- Single output match (TPU3 functionality)

- Double output match with several double match sub-modes
- Input-dependent output generation

The double match functionality has various combinations for generation of service request and determining pin actions. For more details refer to Enhanced Channels.

In addition to the predefined channel configurations above, the user can also program its own channel configuration, defining how input captures, matches and service-requests are related.

43.2.1.1.3 Host Interface

The Host interface allows the Host CPU to control the operation of the eTPU. The Host CPU must initialize the eTPU by writing to the appropriate Host interface registers to assign a Function and priority to each channel. In addition, the Host writes to the Host Service Request and channel configuration registers to further define Function operation for each initialized channel. Refer to Host Interface for a detailed description.

When the SCM is implemented by RAM, the Host must first initialize it with the proper microcode program prior to enabling any eTPU Function, and then enable eTPU access (which also disables Host access).

43.2.1.1.4 Shared Data Memory - SDM

The SDM works as data RAM which can be accessed by the Host CPU and up to two eTPU Engines. This memory is used for information transfer between the Host CPU and the eTPU, as data storage for the eTPU microcode program or for communication between the two eTPU Engines. SDM width is 32 bits, and is accessible by the Host as byte, 16-bit or 32-bit wide. eTPU can access it as full 32-bits, lower 24-bits or upper byte (8-bit).

The host can also access the SDM space mirrored in other area with Parameter Sign Extension (PSE). Parameter Sign Extension accesses differ from the usual host accesses to the original SDM area as follows:

- Writes are effective only to the lower 3 bytes of a word: the word's most significant byte is kept unaltered in SDM.
- Reads return the lower 3 bytes of a word sign-extended to 32 bits, i.e.: the most significant bit of the word's 2nd most significant byte is copied in all 8 bits of the most significant read byte.

Each eTPU channel can be associated with a variable number of parameters located in the SDM, according to its selected Function. In addition, the SDM can be fully shared between two eTPU Engines, enabling direct communication between them.

High flexibility of the SDM utilization is achieved as follows:

- Each channel has a programmable base address pointing to the address of its first parameter with two parameter granularity. This way the SDM can be partitioned according to the actual function needs.
- The microcode can access the first 128 parameters of the selected channel in channel relative access mode.
- Each Engine can access all the SDM address space in indirect addressing mode. Blocks of data are easily transferred using stack operation.
- Absolute addressing mode can access the first 256 parameters (TPU3 functionality), implementing a shared pool of parameters holding global variables.

In the Host address space each parameter occupies four bytes. eTPU usage of the upper byte is achieved by having a 32-bit P register which can access the upper byte, the lower 24 bits or all the 32 bits. The microcode can switch between access sizes at any time.

Each Function may require a different number of parameters. During the eTPU initialization the Host has to program channel base addresses, allocating proper parameters for each channel according to its selected Function.

43.2.1.1.5 Shared Subsystem Area

The Shared Subsystem (SSS) area is a memory map region used by both the host and the eTPU to access registers of the reaction channel, giving the user the option to control them by the host, the eTPU code, or a combination of both.

43.2.1.1.6 Scheduler

Out of reset, all channels are disabled. The Host CPU makes a channel active by assigning it one of three priorities: high, middle, or low. The Scheduler determines the order in which channels are serviced based on channel number and assigned priority. The priority mechanism, implemented in hardware, ensures that all requesting channels are serviced. For additional details refer to Scheduler.

43.2.1.1.7 Microengine

eTPU microengine is a simple RISC implementation which performs each instruction in a microcycle of two eTPU clocks, while prefetching the next instruction through an instruction pipeline. Instruction execution time is constant unless it gets wait states from the SDM arbitration. Two eTPU Engines share code memory without having any performance degradation by interleaving their accesses (the Shared Code Memory has one-clock access time).

Instruction width is 32 bits. The microengine instruction set provides basic arithmetic and logic operations, flow control (jumps and subroutine calls), SDM access, and Channel configuration and control. The instruction formats are defined in such a way that allow particular combinations of two or three of these operations with unconflicting resources to be executed in parallel in the same microcycle.

Microengine has also an independent Multiply/Divide/MAC unit that performs these complex operations in parallel with other microengine instructions.

Channel functionality is tightly integrated to the instruction set through Channel Control operations and conditional Branch operations, which support jumps/calls on Channel-specific conditions. This allows quick and terse Channel configuration and control code, contributing to reduced service time.

Detailed description can be found in Microengine.

43.2.1.1.8 Dual eTPU Engine System

A typical eTPU implementation includes two eTPU Engines sharing SDM and the same code in SCM.

The two eTPU Engines share the Bus Interface Unit (BIU) and the data memory (SDM) which enable Host-eTPU and eTPU Engine-Engine communication. The shared BIU includes coherency logic which supports dual parameter (8 bytes) coherency in transfers between Host and eTPU, using a temporary parameter area within the SDM. More details on this can be found on Parameter Sharing and Coherency.

43.2.2 Features

The following sections show eTPU features and enhancements.

43.2.2.1 eTPU Feature Summary

The eTPU includes these distinctive features:

- up to 32 channels for each eTPU Engine, each channel is associated with an Input/ Output signal pair.
  - enhanced input digital filters on the input pins for improved noise immunity. The eTPU digital filter can use 2 samples, 3 samples or work in continuous mode.
  - identical, orthogonal channels, except for channel 0: each channel can perform any time function. Each time function can be assigned to more than one channel at a given time, so each signal can have any functionality. Channel 0 has the same capabilities of the others, but can also work with special Angle Counter logic (see below).
  - Link Service Request allows activation of a Channel function by request of another channel, even between eTPU Engines.
  - Host Service Request allows activation of a Channel function by Host CPU request
  - each channel has an event mechanism which supports single and double action functionality in various combinations. It includes two 24-bit capture registers, two 24-bit match registers, 24-bit greater-equal and equal-only comparators.
- Two independent 24-bit time bases for channel synchronization:
  - first time base clocked by eTPU clock with programmable prescaler division from 1 to 512 (in steps of 2), or by output of second time base prescaler.
  - first time base can also be clocked by external signal with programmable prescaler division of 1 to 256.
  - second time base clocked by external signal with programmable prescaler division from 1 to 64.
  - second time base external clock source can be replaced by eTPU clock divided by 8.
  - both time bases can be exported or imported via Shared Time and Counter bus.
  - second time base counter can work as an Angle counter, enabling angle based applications to match angle instead of time.
  - second time base can also be used as a pulse accumulator gated by external signal.
- Event-Triggered RISC processor (microengine):
  - 3 stage pipeline implementation (fetch, decoding and execution), with separate instruction memory - SCM - and data memory - SDM (Harvard architecture)

- fixed-length instruction execution in one microcycle of two eTPU clocks
- interleaved SCM access in dual eTPU Engine avoids contention in time for instruction memory
- SCM address space of up to 16K positions (64 Kbytes) (see chip-specific eTPU information)
- SDM with interleaved access in dual eTPU Engine avoids contention for data memory
- SDM address space (both Engines) of up to 8 Kbytes (see chip-specific eTPU information)
- instruction set with embedded Channel support, including specialized Channel control subinstructions and conditional branching on Channel-specific flags.
- channel-oriented addressing: channel-bound address mode with Host configured Channel Base Address allows channel data isolation, independent of microengine application code.
- channel-bound data address space of up to 128 32-bit parameters (512 bytes)
- global parameter address mode allows access to common Channel data of up to 256 32-bit parameters (1024 bytes)
- support for indirect and stacked data access schemes.
- parallel execution of: data access, ALU, Channel control and flow control subinstructions in selected combinations.
- 32-bit microengine registers and 24-bit resolution ALU, with 1 microcycle addition and subtraction, absolute value, bitwise logical operations on 24-bit, 16-bit, or byte operands; single bit manipulation, shift and rotate operations, sign extension and conditional execution.
- additional 24-bit Multiply/MAC/Divide unit which supports all signed/unsigned Multiply/MAC combinations, and unsigned 24-bit Divide. The MAC/Divide unit works in parallel with the regular microcode commands.

- Resource sharing features support channel sharing of channel registers, memory and microengine time:
  - hardware Scheduler works as a "task management" unit, dispatching event service routines by predefined, Host-configured priority.

- automatic Channel context switch when a "task switch" occurs, i.e., one Function Thread ends and another begins to service a request from other Channel: Channel-specific registers, flags and parameter base address are automatically loaded for the next serviced channel.
- individual channel priority setting in 3 levels: high, middle and low.
- Scheduler priority scheme allows calculation of worst case latency for event servicing and ensures servicing all channels by preventing permanent blockage.
- SDM shared between Host CPU and both eTPU Engines, supporting communication either between Channels and Host or inter-channel.
- hardware implementation of 4 Semaphores supports resource sharing between both eTPU Engines.
- Hardware semaphores directly supported by the microengine instruction set.
- dual parameter coherency hardware support allows atomic (to host) access to 2 parameters by microengine(s) in back-to-back accesses.
- coherent dual-parameter controller allows atomic (to microengines) accesses to 2 parameters by the host.

- Development support features:
  - Nexus class 3 debug support (optional, associated with the eTPU-Nexus Block NDEDI).
  - Software breakpoints.
  - Debug interface supporting single-step execution, forced microinstruction execution, Hardware breakpoints and watchpoints on several conditions.
- Safety support features:
  - SCM (code memory) continuous signature-check built-in self test (MISC - Multiple Input Signature Calculator), runs concurrently with eTPU normal operation.
  - See eTPU Plus Enhancements over eTPU) for additional safety features.

43.2.2.2 eTPU Enhancements over TPU3

- 32 orthogonal channels with enhanced functionality. Full support for double action with double match and double transition sub-mode combinations.

- Input and Output features separated in channel logic and microinstructions, allowing input and output signals to be processed separately or combined.
- Increased time resolution and execution unit to 24 bits.
- Increased linear code memory, shared by two eTPU Engines, configurable up to 16K positions (64 KB).
- Increased SDM address range (8 KB each Engine) and width (32 bits per parameter). The SDM can be dynamically allocated to support variable number of parameters for each channel. Each channel can have access to at least 256 parameters.
- The SDM is fully shared by two eTPU Engines (SDM), supporting direct inter-engine communication with the help of hardware semaphores.
- Enhanced arithmetic operations, including add/subtract with carry, absolute value, multiple shift and rotate, conditional execution with variable operand widths.
- Enhanced logic operations, including bitwise operations (and, or, xor) and bit manipulation, with conditional execution. Support for read-modify-write of any bit in the SDM.
- Hardware for Multiply/MAC/Divide, running in parallel to execution of other operations. The 24-bit divide result is available after 13 other unrelated instructions. Multiplication supports any data width of both operands (8, 16 or 24 bits), signed or unsigned. A 24x24 Multiply/MAC result is available after four other unrelated instructions. A 24x8 Multiply/MAC result is available after one other unrelated instruction.
- Supports export/import of time bases from other sources through the real time bus (STAC - Shared Time and Counter bus). This internal bus is used for sharing real time data between multiple peripherals.
- Contains angle clock hardware, supported by microcode, which can provide a 24-bit angle bus instead of time bus. This feature enables the eTPU to run angle based engine control applications.
- More interrupt types. Each eTPU channel can generate a data transfer request interrupt, in addition to regular interrupts, and one global exception interrupt. Data Transfer requests can be used either as interrupt sources or DMA requests. This feature takes advantage of DMA peripherals which off-load the Host. Interrupt Overflow status is also provided.
- Improved visibility to the Host (pin states, time bases, serviced channel).

- An edge case of priority inversion on TPU3 Scheduler was resolved.
- Supports channel link requests between eTPU Engines.

43.2.2.3 eTPU Plus Enhancements over eTPU

eTPU Plus features include the enhancements below in addition to all eTPU features

- TCR1, channel logic and digital filters (both channel and TCRCLK) now have an option to run at divisions of full eTPU clock speed, besides eTPU clock / 2 (see T2/T4 Channel Timing).
- Channels support unordered transitions: transition B can now be detected before transition A. Related to this enhancement, TDLA and TDLB can now be independently negated by microcode (see Channel Mode Logic and Event Flags)
- Channel logic can now work on full eTPU clock, allowing faster response with slow clocks (see T2/T4 Channel Timing).
- Added a new User Programmable Channel Mode: the blocking, enabling, service request and capture characteristics of this channel mode can be programmed via microcode (see General Channel Registers).
- Microinstructions now provide an option to issue Interrupt and Data Transfer requests selected by CHAN. They can also be requested simultaneously at the same instruction (see Channel Interrupt and Data Transfer Requests).
- Channel Flags 0 and 1 can now be tested for branching, besides selecting the entry point (see Branch Operations).
- Channel digital filters can be bypassed, and can be clocked by full eTPU clock (see Bypass Mode).

Backward compatibility is maintained so that all legacy object code (both Host and microcode) runs on eTPU Plus without modification.

43.2.2.4 eTPU2 enhancements over eTPU Plus

eTPU2 features include the enhancements below in addition to all eTPU Plus enhancements and eTPU features.

- Scheduler priority-passing mechanism can now be disabled (see Priority Passing Disabling).
- New Watchdog mechanism kills threads over a programmable timeout (see Watchdog).

- New counter allows microengine load information collection for performance analysis (see Idle Counter).
- Channels 1 and 2 (besides channel 0) can now be selected to control the EAC (see the TBCR_ENG1 and TBCR_ENG2 eTPU Time Base Configuration Registers).
- Timebase prescalers are now reset when the GTBE input is negated, guaranteeing synchronization with eMIOS in all cases (see GTBE - Global Time Base Enable).
- New MISC flag indicates when an SCM signature calculation round is completed. This allows measuring of the average MISC scan period in a real application situation (see the eTPU Module Configuration Register - MCR).
- New channel TCCEA flag allows continuous capture even after TDLA is set, making it fully compatible with TPU behavior (see TCCEA - Transition Continuous Capture Enable).
- New branch condition PRSS tells the pin state at the time when a channel (match or transition) service request occurred (see Pin Control Registers).
- MRLEA/B can now be negated independently by microcode (see Disable Matches).
- New Engine Relative address mode allows a function to access SDM address space common to one engine, but distinct between engines (see Engine Relative Addressing Mode under SDM Addressing Modes).
- Memory error support for Code (SCM) and Data (SDM) memories: error detection, correction, and hardware-assisted soft-error fixing (see Memory Error Support).
- Sharing of microengine and host accesses to subsystem modules (for instance, the Reaction Module), allowing their control by the eTPU, the host, or both (see Shared Subsystems Support).

Backward compatibility is maintained so legacy object code (both Host and microcode) runs on eTPU2 without modification.

43.2.2.5 eTPU2 Plus enhancements over eTPU2

eTPU2 Plus features include the enhancements listed below in addition to all eTPU features, eTPU Plus enhancements and eTPU2 enhancements.

- New Latency Monitor allows latency measurements, either for individual channels or channels combined in a single priority (High, Medium or Low).
- Watchdog can now be configured to disable thread forced ending and channel blocking, allowing it to monitor thread lengths with no interference on the application.

- EAC Angle Tick Generator can now be clocked by the TCR2 prescaler output, as an alternative to the TCR1 clock.
- EAC now supports up to 7 missing teeth on hardware.

Backward compatibility is maintained so legacy object code (both Host and microcode) runs on eTPU2 Plus without modification.

43.2.3 Modes of Operation

eTPU can be seen as capable of working in the following modes:

- **User Configuration Mode**

  User has the ability to program the eTPU Cores with User Time Functions, having access to the Shared Code Memory (SCM).

- **User Mode**

  User does not access the eTPU Shared Code Memory:

  - Use of predefined eTPU Functions
  - No need for eTPU Core programming ability

- **Debug Mode**

  User debugs eTPU code, accessing special Trace/Debug features via Nexus interface:

  - hardware breakpoint/watchpoint setting
  - access to internal registers
  - single-step execution
  - forced instruction execution
  - software breakpoint insertion and removal.

- **Module Disable Mode**

  eTPU Engine clocks are stopped through a register write to ETPUECR bit MDIS, saving power. Input sampling stops. eTPU Engines can be in Module Disable Mode independently. Module Disable Mode stops only the Engine clock, so that the Shared BIU, and Global Channel registers can be accessed, and interrupts and DMAs can be cleared and enabled/disabled. An Engine only enters Module Disable Mode when any currently running thread is finished (see Functions and Threads).

- **Stop Mode**

Stop Mode is entered when eTPU answers device stop request assertion with stop acknowledge. The definition of which clocks are stopped is made at the MCU level, which defines whether or not registers can be accessed, interrupts and DMA requests cleared.

These modes are loosely selected: there is no unique register field or signals to choose between them. Some features of one mode can be used with features of other mode(s). More on this subject can be found on eTPU Mode Selection below.

Note

Throughout this chapter, an engine is said to be "stopped" if it is either in Module Disable mode or Stop mode.

43.2.3.1 eTPU Mode Selection

User and User Configuration are the production operating modes, and differ from each other only in access to SCM. User programmability is only possible with a RAM SCM.

On chips where the SCM is implemented as a RAM, it can either be accessed directly from IP-Bus for code loading, or for software breakpoint setting . On chips with a ROM SCM, an internal SCM Emulation RAM may be used, depending on the specific MCU implementation, to replace ROM SCM for test or debug purposes. SCM Emulation RAM is selected in a MCU-specific way. For more details, see .

Debug Mode is characterized by the use of the debug interface features. Specifically, Nexus blocks provide Nexus class 3 debug features. For more information on debug features, refer to Development Support Features.

Module Disable Mode is entered by setting ETPUECR register bit MDIS. eTPU Engines can be individually stopped going into Module Disable Mode (there is one ETPUECR register for each Engine). Each engine can leave Module Disable Mode by writing MDIS=0 (which can only be done if VIS=0).

Stop Mode is activated by IP-Bus (device stop request). In this case, the eTPU waits for both eTPU Engines to enter in stop mode, and then asserts the stop acknowledge line. eTPU leaves Stop Mode when device stop request is negated, but only if VIS=0. If device stop request is negated and VIS=1, eTPU will leave Stop Mode as soon as VIS=0.

Note

An Engine can stay in Module Disable mode when it leaves Stop Mode if its bit MDIS=1, even if the other leaves it.

43.3 Detailed Signal Descriptions

Detailed signal descriptions are presented in the following sections.

43.3.1 eTPU Channel Output Signals

Each channel output signal is associated with a channel. The microcode may affect the logic level of an output signal[1] by implementing one of two actions:

- Specify the logic level output to the signal when there is a match or a transition.
- Immediately force a logic level.

The output signal may also be forced to a logic level, independently of the output value from the channel logic, by one of the four (each Engine) output disable input signals (see eTPU Channel Output Disable Signals).

The output buffer enable signals are not used.

43.3.2 eTPU Channel Input Signals

Each channel input signal is associated with a channel. The microcode can directly control the effect of the transition edge. Each channel can be programmed to sense a transition when a rising and/or falling edge is detected. The channel logic can also process two transition events, and relate these events to each other and to other programmed timer events. The edge sensitivities of the two transition events are configured independently by microcode. For further information refer to Enhanced Channels and Transition Detection and Pin Action Control.

Each channel input signal has an associated synchronizer made of two flip-flops sampling the signal on every other eTPU clock[2], followed by a digital filter. This digital filter can work in three sub-modes, whose purpose is to filter out noise pulses that have width less then a programmed value of eTPU clocks, preventing these transitions from being input to the transition detect logic. The synchronizer and digital filter are guaranteed to pass pulses that are greater than a programmed value. All channel input filters in one Engine work on the same mode and sampling clock. For more information on channel input

1. Note that the minimum pulse width is one microcycle (two eTPU clocks), and slow pads may not be able to transfer it on time. For generation of very short pulses the eTPU pads have to be programmed by the system integration for fast operation mode with the voltage levels defined for fast pad operation in the MCU technology.
2. Sampled on the T4 microcycle phase, see Microcycle and I/O Timing.

filters, refer to Enhanced Digital Filter - EDF . In one of the Angle Modes, the output of the digital filter of channel 0 is replaced by the output of TCRCLK signal digital filter (see EAC - eTPU Angle Counter).

43.3.3 Time Base Clock Signal — TCRCLK

TCRCLK is an input signal used to control the Time Bases TCR1 and TCR2. There is one independent TCRCLK input for each Engine. For pulse accumulator operations TCRCLK can be used as a gate for a counter based on the eTPU clock divided by eight. For Angle operations TCRCLK can be used to get the tooth transition indications in Angle Mode. Further details can be found on Time Bases and EAC - eTPU Angle Counter.

Like the channel input signals, the TCRCLK signal has an associated synchronizer followed by a digital filter. This digital filter can work in two sub-modes, whose purpose is to filter out noise pulses that have width less then a programmed value of eTPU clocks, preventing these transitions from being input to the transition detect logic. The synchronizer and digital filter are guaranteed to pass pulses that are greater than a programmed value. The clock and operation sub-mode of the TCRCLK filter is configured independently of the other channel input filters, through the field TCRCF in register ETPUTBCR. For more information on filter sub-modes, refer to TCRCLK Digital Filter. In one of the Angle Modes, the output of the digital filter of channel 0 is replaced with the output of TCRCLK signal digital filter (see EAC - eTPU Angle Counter).

43.3.4 eTPU Channel Output Disable Signals

Each of these 4 input signals are used to force the outputs of a group of 8 channels to an inactive level. When an ODIS input is active, all the channels in its group of 8 that have their bits ODIS=1 in ETPUCxCR register have their outputs forced to the opposite of the value specified in bit OPOL of the same register. Therefore, channels can be individually selected to be affected by the output disable signals, as well as their disabling forced polarity (see 'Pin State Input/Output Logic' figure in Pin Control Registers).

See the chip-specific eTPU information for details on the output channel disable group definition.

In a Dual-engine eTPU there are 8 output disable signals for the 64 channels.

43.4 Memory Map/Register Definition

The guideline for the description of all bits and fields throughout the“Memory Map/ Register Definition” section is to provide only a brief explanation (without examples or method of use) of the features, since it will be used mainly as a reference for the reader that is studying the “Functional Description" section, where those features are explained in detail.

43.5 Memory Map

The eTPU System simplified memory map is shown in Table 43-3. Each of the register areas shown may have their own reserved address areas.

Table 43-4 shows a detailed memory map. Offsets are relative to the eTPU Base address, which is MCU-dependent.

See the chip-specific eTPU information about accessing eTPU registers in chip-level Stop mode.

SCM unused area is decoded and returns a fixed opcode, determined by the register ETPUSCMOFFDATAR.

Table 43-3. High Level Memory Map

Offset	Use
0x00 – 0x1F	System Configuration Registers
0x20 – 0x2F	eTPU 1 Time Base Registers
0x30 – 0x3F	RESERVED[1, 1]
0x40 – 0x4F	eTPU 2 Time Base Registers
0x50 – 0x5F	RESERVED[1]
0x60 – 0x6F	eTPU 1 Extra Engine Registers
0x70 – 0x7F	eTPU 2 Extra Engine Registers
0x80 – 0xFF	Latency Monitor Registers
0x100 – 0x13F	Memory Error Support Registers
0x140 – 0x1FF	RESERVED[1]
0x200 – 0x2FF	eTPU 1/2 Global Channel Registers
0x300 – 0x3FF	RESERVED[1]
0x400 – 0x7FF	eTPU 1 Channel Registers
0x800 – 0xBFF	eTPU 2 Channel Registers
0xC00 – 0xFFF	Shared Subsystems Integration Registers
0x1000 – 0x3FFF	RESERVED[1]

Table continues on the next page...

Table 43-3. High Level Memory Map (continued)

Offset	Use
0x4000 – 0x7FFF	SSS - Shared Subsystems area
0x8000 – 0xBFFF[2, 2]	SDM - Shared Data Memory
0xC000 – 0xFFFF[2]	SDM PSE mirror[3]
0x10000 – 0x1FFFF[2]	SCM - Shared Code Memory[4]

1. Reserved addresses must not be used. Access to these memory positions complete with 0-wait-states, but the result is unspecified.
2. Actual sizes of SCM and SDM are MCU-dependent (see Configuration section).
3. Parameter Sign Extension access area, see the “Parameter Access” section.
4. SCM access is available only when bit VIS=1 on register ETPUMCR, under certain conditions (see the “ETPUMCR - eTPU Module Configuration Register” section).

Table 43-4. Detailed Memory Map

Offset	Use
System Configuration Registers	
0x00	ETPUMCR - eTPU Module Configuration Register
0x04	ETPUCDCR - eTPU Coherent Dual-Parameter Controller Register
0x08	RESERVED
0x0C	ETPUMISCCMPR - eTPU MISC Compare Register
0x10	ETPUSCMOFFDATAR - eTPU SCM Off-range Data Register[1]
0x14	ETPUECR_1 - eTPU 1 Engine Configuration Register
0x18	ETPUECR_2 - eTPU 2 Engine Configuration Register
0x1C	RESERVED
Time Base Registers	
0x20	ETPUTBCR_1 - eTPU 1 Time Base Configuration Register
0x24	ETPUTB1R_1 - eTPU 1 Time Base 1
0x28	ETPUTB2R_1 - eTPU 1 Time Base 2
0x2C	ETPUREDCR_1 - eTPU 1 STAC Configuration Register
0x30	RESERVED
0x34	RESERVED
0x38	RESERVED
0x3C	RESERVED
0x40	ETPUTBCR_2 - eTPU 2 Time Base Configuration Register
0x44	ETPUTB1R_2 - eTPU 2 Time Base 1
0x48	ETPUTB2R_2 - eTPU 2 Time Base 2
0x4C	ETPUREDCR_2 - eTPU 2 STAC Configuration Register
0x50	RESERVED
0x54	RESERVED
0x58	RESERVED
0x5C	RESERVED
Engine Related Registers	

Table continues on the next page...

Table 43-4. Detailed Memory Map (continued)

Offset	Use
0x60	ETPUWDTR_1 - eTPU 1 Watchdog Timer Register
0x64	RESERVED
0x68	ETPUIDLER_1 - eTPU 1 Idle Counter Register
0x6C	RESERVED
0x70	ETPUWDTR_2 - eTPU 2 Watchdog Timer Register
0x74	RESERVED
0x78	ETPUIDLER_2 - eTPU 2 Idle Counter Register
0x7C	RESERVED
Latency Monitor Registers	
0x80	ETPULM0CR_1 - eTPU 1 Latency Monitor 0 Configuration Register
0x84	ETPULM0MR_1 - eTPU 1 Latency Monitor 0 Measure Register
0x88	ETPULM1CR_1 - eTPU 1 Latency Monitor 1 Configuration Register
0x8C	ETPULM1MR_1 - eTPU 1 Latency Monitor 1 Measure Register
0x90	ETPULM1CR_1 - eTPU 1 Latency Monitor 2 Configuration Register
0x94	ETPULM1MR_1 - eTPU 1 Latency Monitor 2 Measure Register
0x98	ETPULM1CR_1 - eTPU 1 Latency Monitor 3 Configuration Register
0x9C	ETPULM1MR_1 - eTPU 1 Latency Monitor 3 Measure Register
0xA0	ETPULM1CR_1 - eTPU 1 Latency Monitor 4 Configuration Register
0xA4	ETPULM1MR_1 - eTPU 1 Latency Monitor 4 Measure Register
0xA8	ETPULM1CR_1 - eTPU 1 Latency Monitor 5 Configuration Register
0xAC	ETPULM1MR_1 - eTPU 1 Latency Monitor 5 Measure Register
0xC0	ETPULM0CR_2 - eTPU 2 Latency Monitor 0 Configuration Register
0xC4	ETPULM0MR_2 - eTPU 2 Latency Monitor 0 Measure Register
0xC8	ETPULM1CR_2 - eTPU 2 Latency Monitor 1 Configuration Register
0xCC	ETPULM1MR_2 - eTPU 2 Latency Monitor 1 Measure Register
0xD0	ETPULM1CR_2 - eTPU 2 Latency Monitor 2 Configuration Register
0xD4	ETPULM1MR_2 - eTPU 2 Latency Monitor 2 Measure Register
0xD8	ETPULM1CR_2 - eTPU 2 Latency Monitor 3 Configuration Register
0xDC	ETPULM1MR_2 - eTPU 2 Latency Monitor 3 Measure Register
0xE0	ETPULM1CR_2 - eTPU 2 Latency Monitor 4 Configuration Register
0xE4	ETPULM1MR_2 - eTPU 2 Latency Monitor 4 Measure Register
0xE8	ETPULM1CR_2 - eTPU 2 Latency Monitor 5 Configuration Register
0xEC	ETPULM1MR_2 - eTPU 2 Latency Monitor 5 Measure Register
Memory Error Support Registers[2]	
0x100	ETPUMECR - eTPU Memory Error Control Register
0x104	ETPUDEIAR - eTPU Data Error Injection Address Register
0x108	ETPUDEIDPR - eTPU Data Error Injection Data Pattern Register
0x10C	ETPUDEIPPR - eTPU Data Error Injection Parity Pattern Register
0x110	ETPUDERAR - eTPU Data Error Report Address Register

Table continues on the next page...

Table 43-4. Detailed Memory Map (continued)

Offset	Use
0x114	ETPUDERDR - eTPU Data Error Report Data Register
0x118	ETPUDERSR - eTPU Data Error Report Syndrome Register
0x11C	RESERVED
0x120	ETPUMESR - eTPU Memory Error Status Register
0x124	ETPUCEIAR - eTPU Code Error Injection Address Register
0x128	ETPUCEIDPR - eTPU Code Error Injection Data Pattern Register
0x12C	ETPUCEIPPR - eTPU Code Error Injection Parity Pattern Register
0x130	ETPUCERAR - eTPU Code Error Report Address Register
0x134	ETPUCERDR - eTPU Code Error Report Data Register
0x138	ETPUCERSR - eTPU Code Error Report Syndrome Register
0x13C	ETPUCEFR - eTPU Code Error Fix Register
0x140 - 0x1FF	RESERVED
Global Channel Registers	
0x200	ETPUCISR_1 - eTPU 1 Channel Interrupt Status Register
0x204	ETPUCISR_2 - eTPU 2 Channel Interrupt Status Register
0x208	RESERVED
0x20C	RESERVED
0x210	ETPUCDTRSR_1 - eTPU 1 Channel Data Transfer Request Status Register
0x214	ETPUCDTRSR_2 - eTPU 2 Channel Data Transfer Request Status Register
0x218	RESERVED
0x21C	RESERVED
0x220	ETPUCIOSR_1 - eTPU 1 Channel Interrupt Overflow Status Register
0x224	ETPUCIOSR_2 - eTPU 2 Channel Interrupt Overflow Status Register
0x228	RESERVED
0x22C	RESERVED
0x230	ETPUCDTROSR_1 - eTPU 1 Channel Data Transfer Request Overflow Status Register
0x234	ETPUCDTROSR_2 - eTPU 2 Channel Data Transfer Request Overflow Status Register
0x238	RESERVED
0x23C	RESERVED
0x240	ETPUCIER_1 - eTPU 1 Channel Interrupt Enable Register
0x244	ETPUCIER_2 - eTPU 2 Channel Interrupt Enable Register
0x248	RESERVED
0x24C	RESERVED
0x250	ETPUCDTRER_1 - eTPU 1 Channel Data Transfer Request Enable Register
0x254	ETPUCDTRER_2 - eTPU 2 Channel Data Transfer Request Enable Register
0x258	RESERVED
0x25C	RESERVED
0x260	ETPUWDSR_1 - eTPU 1 Watchdog Status Register
0x264	ETPUWDSR_2 - eTPU 2 Watchdog Status Register

Table continues on the next page...

Table 43-4. Detailed Memory Map (continued)

Offset	Use
0x268-0x27F	RESERVED
0x280	ETPUCPSSR_1 - eTPU 1 Channel Pending Service Status Register
0x284	ETPUCPSSR_2 - eTPU 2 Channel Pending Service Status Register
0x288	RESERVED
0x28C	RESERVED
0x290	ETPUCSSR_1 - eTPU 1 Channel Service Status Register
0x294	ETPUCSSR_2 - eTPU 2 Channel Service Status Register
0x298	RESERVED
0x29C	RESERVED
0x300 - 0x3FF	RESERVED
Channel Configuration and Control Registers	
0x400	ETPUC0CR_1 - eTPU 1 Channel 0 Configuration Register
0x404	ETPUC0SCR_1 - eTPU 1 Channel 0 Status and Control Register
0x408	ETPUC0HSRR_1 - eTPU 1 Channel 0 Host Service Request Register
0x40C	RESERVED
0x410	ETPUC1CR_1 - eTPU 1 Channel 1 Configuration Register
0x414	ETPUC1SCR_1 - eTPU 1 Channel 1 Status and Control Register
0x418	ETPUC1HSRR_1 - eTPU 1 Channel 1 Host Service Request Register
0x41C	RESERVED
-----	-----
0x5F0	ETPUC31CR_1 - eTPU 1 Channel 31 Configuration Register
0x5F4	ETPUC31SCR_1 - eTPU 1 Channel 31 Status and Control Register
0x5F8	ETPUC31HSRR_1 - eTPU 1 Channel 31 Host Service Request Register
0x5FC - 0x7FF	RESERVED
0x800	ETPUC0CR_2 - eTPU 2 Channel 0 Configuration Register
0x804	ETPUC0SCR_2 - eTPU 2 Channel 0 Status and Control Register
0x808	ETPUC0HSRR_2 - eTPU 2 Channel 0 Host Service Request Register
0x80C	RESERVED
0x810	ETPUC1CR_2 - eTPU 2 Channel 1 Configuration Register
0x814	ETPUC1SCR_2 - eTPU 2 Channel 1 Status and Control Register
0x818	ETPUC1HSRR_2 - eTPU 2 Channel 1 Host Service Request Register
0x81C	RESERVED
-----	-----
0x9F0	ETPUC31CR_2 - eTPU 2 Channel 31 Configuration Register
0x9F4	ETPUC31SCR_2 - eTPU 2 Channel 31 Status and Control Register
0x9F8	ETPUC31HSRR_2 - eTPU 2 Channel 31 Host Service Request Register
0x9FC - 0xBFF	RESERVED
Shared Subsystem Support Registers	
0xC00	ETPUSSS0IR - eTPU SSS 0 Integration Register

Table continues on the next page...

Table 43-4. Detailed Memory Map (continued)

Offset	Use
0xC04	ETPUSSS1IR - eTPU SSS 1 Integration Register
0xC08	ETPUSSS2IR - eTPU SSS 2 Integration Register
0xC0A	ETPUSSS3IR - eTPU SSS 3 Integration Register
0xC0C	ETPUSSS4IR - eTPU SSS 4 Integration Register
0xC10	ETPUSSS5IR - eTPU SSS 5 Integration Register
0xC14	ETPUSSS6IR - eTPU SSS 6 Integration Register
0xC18	ETPUSSS7IR - eTPU SSS 7 Integration Register
0xC1A	ETPUSSS8IR - eTPU SSS 8 Integration Register
0xC1C	ETPUSSS9IR - eTPU SSS 9 Integration Register
0x1000-0x3FFF	RESERVED
0x4000 - 0x7FFF	Shared Subsystems area - SSS
0x8000 - 0xBFFF[3, 3]	Shared Data Memory - SDM
0xC000 - 0xFFFF[3]	Shared Data Memory - SDM - PSE mirror [4]
0x10000-1FFFF[5]	Shared Code Memory - SCM[6]

1. This register is not implemented in some MCUs; see the “ETPUSCMOFFDATAR - eTPU SCM Off-range Data Register" section.
2. These registers are not implemented in some MCUs; see the“Memory Error Support” section.
3. The actual SDM size is MCU-dependent (see Configuration section).
4. Parameter Sign Extension access area, see the “Parameter Access” section.
5. The actual SCM size is MCU-dependent (see Configuration section). When the size is not the maximum, the unused SCM address range returns the value of the register ETPUSCMOFFDATAR.
6. SCM access is available only when bit VIS=1 on register ETPUMCR, under certain conditions (see the “ETPUMCR - eTPU Module Configuration Register” section).

43.6 System Configuration memory map and registers

eTPU_SCR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	eTPU Module Configuration Register (eTPU_SCR_MCR)	32	R/W	0000_0000h	43.6.1/1347
4	eTPU Coherent Dual-Parameter Controller Register (eTPU_SCR_CDCR)	32	R/W	0000_0000h	43.6.2/1351
C	eTPU MISC Compare Register (eTPU_SCR_MISCCMPR)	32	R/W	0000_0000h	43.6.3/1353
10	eTPU SCM Off-range Data Register (eTPU_SCR_SCMOFFDATAR)	32	R/W	F377_5FFBh	43.6.4/1353
14	eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1)	32	R/W	See section	43.6.5/1354
18	eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2)	32	R/W	See section	43.6.6/1358

43.6.1 eTPU Module Configuration Register (eTPU_SCR_MCR)

eTPU_MCR is global to both eTPU engines, and resides in the shared BIU. eTPU_MCR gathers global configuration and status in the eTPU system, including global exception. It is also used for configuring the SCM (shared code memory) operation and test.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	SDMERR	WDTO1	WDTO2	MGE1	MGE2	ILF1	ILF2	Reserved	0	0	SCMSIZE				
W	GEC															
Reset	0	0	0	0	0	0	0	0	0	0	0					

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LTO1	LTO2	0		SCMMISC	SCMMISF	SCMMISEN	SCMERR	0	VIS	0					GTBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_SCR_MCR field descriptions

Field	Description
0 GEC	Global Exception Clear. This write-only bit negates global exception request and clears global exception status bits SDMERR, SSAERR, SCMERR, WDTO1/2, MGE1/2, ILF1/2, LTO1/2, and SCMMISF. GEC works the same way with either one or both Engines in Module Disable Mode. 0 Keep Global Exception request and status bits SDMERR, SCMERR, WDTO1/2, ILF1/2, MGE1/2, LTO1/2, and SCMMISF as is. 1 Negate Global Exception, clear status bits SDMERR, SCMERR, WDTO1/2, ILF1/2, MGE1/2, LTO1/2, and SCMMISF.
1 SDMERR	SDM Read Error This flag indicates that an SDM non-correctable read error occurred on a microengine or host (including CDC) read, generating a Global Exception. This bit is cleared by writing 1 to GEC. 0 No Global Exception pending because of SDM read error. 1 Global Exception requested by SDM read error is pending.
2 WDTO1	Watchdog 1 Timeout. Flag WDTO1 indicates that a Watchdog Timeout occurred in engine 1, generating a Global Exception. This bit is cleared by writing 1 to GEC. 0 No Global Exception pending because of Watchdog timeout. 1 Global Exception requested by Watchdog timeout.
3 WDTO2	Watchdog 2 Timeout. Flag WDTO2 indicates that a Watchdog Timeout occurred in engine 2, generating a Global Exception. This bit is cleared by writing 1 to GEC. 0 No Global Exception pending because of Watchdog timeout. 1 Global Exception requested by Watchdog timeout.
4 MGE1	Microcode Global Exception Engine 1. This bit indicates that a global exception was asserted by microcode executed on engine 1. The determination of the reason why the global exception was asserted is application dependent: it can be coded in an SDM status parameter, for instance. This bit is cleared by writing 1 to GEC. 0 No microcode-requested global exception pending. 1 Global exception requested by microcode is pending.
5 MGE2	Microcode Global Exception Engine 2. This bit indicates that a global exception was asserted by microcode executed on engine 2. The determination of the reason why the global exception was asserted is application dependent: it can be coded in an SDM status parameter, for instance. This bit is cleared by writing 1 to GEC. 0 No microcode-requested global exception pending. 1 Global exception requested by microcode is pending.
6 ILF1	Illegal Instruction Flag for Engine 1. The ILFA1 flag is set by the microengine to indicate that an illegal instruction was decoded in Engine 1. This bit is cleared by host writing 1 to GEC. See Illegal Instructions for more details.

Table continues on the next page...

eTPU_SCR_MCR field descriptions (continued)

Field	Description
	0 Illegal Instruction not detected. 1 Illegal Instruction detected by Engine 1.
7 ILF2	Illegal Instruction Flag for Engine 2. The ILFA2 flag is set by the microengine to indicate that an illegal instruction was decoded in Engine 2. This bit is cleared by host writing 1 to GEC. See Illegal Instructions for more details. 0 Illegal Instruction not detected. 1 Illegal Instruction detected by Engine 2.
8 Reserved	Reserved. This field is reserved.
9 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
10 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
11–15 SCMSIZE	SCM Size = 64 Kbytes. This read-only field holds the number of 2 KB SCM Blocks minus 1. See Register Reset Values in Configuration information.
16 LTO1	Latency Timeout for Engine 1. The LTO1 flag indicates the latency measured by the Latency Monitor of Engine 1 surpassed the specified maximum. See Latency Monitor for details. This bit is cleared by writing 1 to GEC. 0 No latency timeout detected by Engine 1’s Latency Monitor. 1 Latency timeout detected by Engine 1's Latency Monitors.
17 LTO2	Latency Timeout for Engine 2. The LTO2 flag indicates the latency measured by the Latency Monitor of Engine 2 surpassed the specified maximum. See Latency Monitor for details. This bit is cleared by writing 1 to GEC. 0 No latency timeout detected by Engine 2’s Latency Monitor. 1 Latency timeout detected by Engine 2's Latency Monitors.
18–19 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
20 SCMMISC	SCM MISC Complete flag. Flag SCMMISC indicates that MISC has completed the evaluation of the SCM signature since reset or since the last time it was cleared. SCMMISC is cleared by writing 1 to it, and is not cleared when MISC is disabled (SCMMISEN=0). SCMMISC asserts at the end of the SCM memory scan, either if the signature matches or not.

Table continues on the next page...

eTPU_SCR_MCR field descriptions (continued)

Field	Description
	0 MISC has not yet completed an SCM signature calculation and compare since the last time SCMMISC was cleared. 1 MISC completed at least one SCM signature calculation and compare since the last time SCMMISC was cleared.
21 SCMMISF	SCM MISC Flag. The SCMMISF bit is set by the SCM MISC (Multiple Input Signature Calculator) logic to indicate that the calculated signature does not match the expected value, at the end of a MISC iteration. See SCM Test - Multiple Input Signature Calculator for more details. This bit is cleared when Global Exception is cleared by writing 1 to GEC. 0 Signature mismatch not detected. 1 MISC has read entire SCM array and the expected signature in ETPUMISCCMPR does not match the value calculated.
22 SCMMISEN	SCM MISC Enable. The SCMMISEN bit is used for enabling/disabling the operation of the MISC logic. SCMMISEN is readable and writable at any time. The MISC logic will only operate when this bit is set to 1. When the bit is reset, the MISC address counter is set to the initial SCM address. When enabled, the MISC will continuously cycle through the SCM addresses, reading each and calculating a CRC. In order to save power, the MISC can be disabled by clearing the SCMMISEN bit. See SCM Test - Multiple Input Signature Calculator for more details. SCMMISEN resets automatically when MISC logic detects an error, i.e., when SCMMISF transitions from 0 to 1, disabling the MISC operation. 0 MISC operation disabled. The MISC logic is reset to its initial state. 1 MISC operation enabled.
23 SCMERR	SCM Read Error. This flag indicates that an SCM non-correctable read error occurred on a host or microengine (including MISC) read, generating a Global Exception. This bit is cleared by writing 1 to GEC. 0 No Global Exception pending because of SCM read error. 1 Global Exception requested by SCM read error is pending.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 VIS	SCM Visibility Bit. VIS bit turns SCM visible to the IP-Bus and resets MISC state (but SCMMISEN keeps its value). This bit is write protected when any of the Engines is not halted or stopped. When VIS=1, the MDIS bits in the eTPU Memory Error Support registers are write protected. 0 SCM is not visible to the slave bus. Accessing SCM address space issues a bus error, writes are protected and reads are meaningless. 1 SCM is visible to the slave bus. MISC state is reset.
26–30 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

eTPU_SCR_MCR field descriptions (continued)

Field	Description
31 GTBE	Global Time Base Enable. GTBE enables time bases in both Engines, allowing them to be started synchronously. **NOTE:** Global Time Base Enable action may also depend on other blocks, as explained in GTBE - Global Time Base Enable. **NOTE:** When GTBE is turned off with Angle Mode enabled, the EAC must be reinitialized before GTBE is turned on again. The EAC reinitialization procedure is described in Restarting Angle Logic. 0 Time bases in both Engines are disabled to run. 1 Time bases in both Engines are enabled to run.

43.6.2 eTPU Coherent Dual-Parameter Controller Register (eTPU_SCR_CDCR)

ETPUCDCR configures and controls dual-parameter coherent transfers. For more information, see Coherent Dual-parameter Controller - CDC.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	STS	CTBASE							PBBASE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PWIDTH	PARM0							WR	PARM1						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_SCR_CDCR field descriptions

Field	Description
0 STS	Start Bit. This bit is set by the host in order to start the data transfer between the parameter buffer pointed by PBBASE and the target addresses selected by the concatenation of fields CTBASE and PARM0/1. The host receives wait-states until the data transfer is complete, when this bit is reset by the coherency logic (see Coherent Dual-parameter Controller - CDC). Therefore, host always reads STS as 0.

Table continues on the next page...

eTPU_SCR_CDCR field descriptions (continued)

Field	Description
	0 Write does not start a coherent transfer. 1 Write starts a coherent transfer.
1–5 CTBASE	Channel Transfer Base. This field concatenates with fields PARM0 and PARM1 to determine the absolute word offset (from the SDM base) of the parameters to be transferred: Parameter 0 word address = {CTBASE, PARM0} + SDM base word address. Parameter 1 word address = {CTBASE, PARM1} + SDM base word address.
6–15 PBBASE	Parameter Buffer Base Address. This field points to the base address of the parameter buffer location, with granularity of 2 parameters (8 bytes). The host (byte) address of the first parameter in the buffer is PBBASE * 8 + SDM Base Byte Address. The microengine absolute (word) address of the first parameter in the buffer is PBBASE * 2.
16 PWIDTH	Parameter Width Selection. This bit selects the width of the parameters to be transferred between the PB and the target address. 0 Transfer 24-bit parameters. The upper byte remains unchanged in the destination address. 1 Transfer 32-bit parameters. All 32 bits of the parameters are written in the destination address.
17–23 PARM0	Channel Parameter number 0. The PARM0 and PARM1 fields in concatenation with CTBASE determine the word address offset (from the SDM base) of the parameters that are destination or source (defined by WR) of the coherent transfer. The word SDM address offset of the parameters are {CTBASE, PARM0/1}. **NOTE:** PARM0 and PARM1 allow non-contiguous parameters to be transferred coherently. The parameter pointed by {CTBASE, PARM0} is the first transferred.
24 WR	Read/Write selection. This bit selects the direction of the coherent data transfer. 0 Read operation. Data transfer is from the selected SDM address to the PB. 1 Write operation. Data transfer is from the PB to the selected SDM address.
25–31 PARM1	Channel Parameter number 1. The PARM0 and PARM1 fields in concatenation with CTBASE determine the word address offset (from the SDM base) of the parameters that are destination or source (defined by WR) of the coherent transfer. The word SDM address offset of the parameters are {CTBASE, PARM0/1}. **NOTE:** PARM0 and PARM1 allow non-contiguous parameters to be transferred coherently. The parameter pointed by {CTBASE, PARM0} is the first transferred.

43.6.3 eTPU MISC Compare Register (eTPU_SCR_MISCCMPR)

ETPUMISCCMPR holds the 32-bit signature expected from the whole SCM array. This register must be written by the host with the 32-bit word to be compared against the calculated signature at the end of the MISC cycle. This register is global to both eTPU Engines. For more detail, see SCM Test - Multiple Input Signature Calculator.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ETPUMISCCMP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_SCR_MISCCMPR field descriptions

Field	Description
0–31 ETPUMISCCMP	Expected Multiple Input Signature Register value. For details, see SCM Test - Multiple Input Signature Calculator.

43.6.4 eTPU SCM Off-range Data Register (eTPU_SCR_SCMOFFDATAR)

ETPUSCMOFFDATAR holds the 32-bit value returned when SCM array is accessed at non-implemented addresses, either by the Host or by the microengine. This register can be written by the host with the 32-bit instruction to be executed by the microengine to recover from runaway code. This register is global to both eTPU Engines. For more detail see SCM Off-range Data.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ETPUSCMOFFDATAR																															
Reset	1	1	1	1	0	0	1	1	0	1	1	1	0	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1

eTPU_SCR_SCMOFFDATAR field descriptions

Field	Description
0–31 ETPUSCMOFFDATAR	SCM Off-range read data value. See SCM Off-range Data.

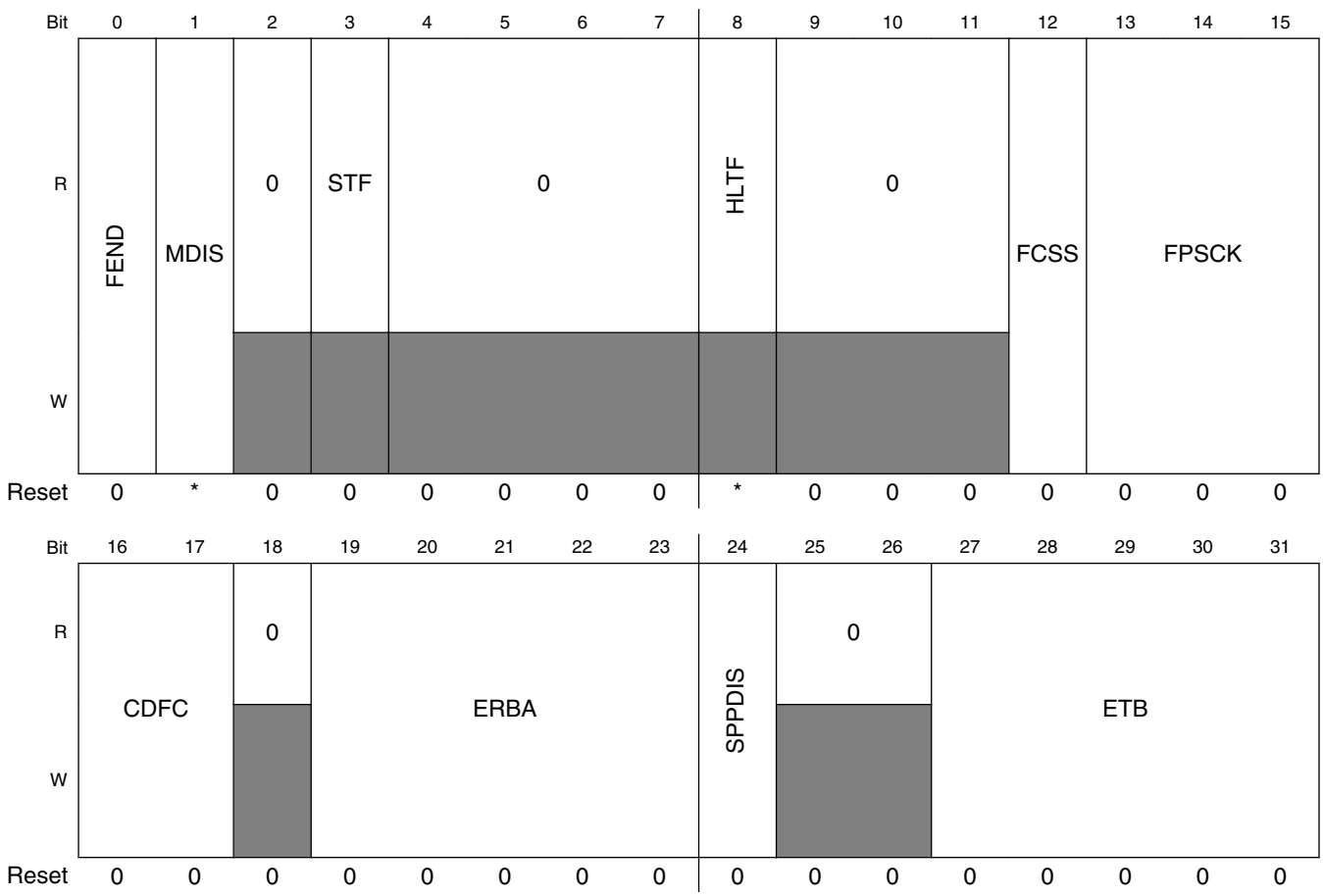
43.6.5 eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1)

Each engine has its own ETPUECR register. ETPUECR holds configuration and status fields that are programmed independently in each engine.

NOTE

See Register Reset Values in Configuration information.

Address: 0h base + 14h offset = 14h



* Notes:

- HLTF field: HLTF field reset value is 1 if, after a reset, the eTPU Nexus debug support is on and a breakpoint request is asserted. Otherwise the reset value is 0.
- MDIS field: Reset value is 1.

eTPU_SCR_ECR_ENG1 field descriptions

Field	Description
0 FEND	Force End. FEND assertion terminates any current running thread as if an END instruction have been executed (see Ending Current Thread - END).

Table continues on the next page...

eTPU_SCR_ECR_ENG1 field descriptions (continued)

Field	Description
	This bit is self-negating when the thread ends. **NOTE:** Only on rare occasions (e.g., during a long stall, see Microengine Stall) FEND can be read as 1, because it negates as soon as the end begins execution. Writing FEND = 1 is ignored and FEND stays 0 when the microengine is in TST, halted, stopped, or idle (no thread executing). 0 Normal operation. 1 Ends any ongoing thread.
1 MDIS	Module Disable Bit. When MDIS is set, the Engine shuts down its internal clocks, going into Module Disable Mode. TCR1 and TCR2 cease to increment, and input sampling stops. The Engine asserts the stop flag (STF) bit to indicate that it has stopped. However, the BIU continues to run, and the Host can access all registers except for the channel registers (see channel registers starting with eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1)) **NOTE:** The Timebase registers can still be read with MDIS=1, but writes are ineffective and a Bus Error is issued. Global Channel Registers and SDM can be accessed normally. After MDIS is set, even before STF asserts, data read from the channel registers is not meaningful and writes are ineffective, issuing a Bus Error. When the MDIS bit is asserted while the microcode is executing, the eTPU will stop when the thread is complete. Stop completes on the next eTPU clock after the stop condition is valid. **NOTE:** Once MDIS is switched from 1 to 0 or vice versa, it must not write a different value until STF changes accordingly. The MDIS bit is write-protected when VIS=1. See Register Reset Values in Configuration information. 0 eTPU Engine runs. 1 Commands Engine to stop its clocks.
2 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
3 STF	Stop Flag Bit. Each Engine asserts its stop flag (STF) to indicate that it has stopped. Only then the host can assume that the Engine has actually stopped. The eTPU system is fully stopped when the STF bits of both eTPU Engines are asserted. In case of a stop request, the eTPU responds by acknowledging stop only after both eTPU 1 and eTPU 2 have been stopped. Engine only stops when any ongoing thread is complete also in this case. Summarizing Engine stop conditions, which STF reflects: STF_1 := (after stop completed) MDIS_1 | device stop request. STF_2 := (after stop completed) MDIS_2 | device stop request. STF_1 and STF_2 mean STF bit from engine 1 and STF bit from engine 2, respectively. See Register Reset Values in Configuration information.

Table continues on the next page...

eTPU_SCR_ECR_ENG1 field descriptions (continued)

Field	Description
	0 Engine is operating. 1 Engine has stopped (after the local MDIS bit has been asserted, or after the IP-Bus stop line has been asserted).
4–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8 HLTF	Halt Mode Flag. If eTPU Engine entered halt state, this flag is asserted. The flag remains asserted while the microengine is in halt state, even during a single-step or forced instruction execution. See Development Support Features for further details about entering Halt Mode. 0 eTPU Engine is not halted. 1 eTPU Engine is halted.
9–11 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
12 FCSS	Filter Clock Source Selection Speeds up the filter clock source before the prescaler, allowing more input capture resolution at minimum prescaling. **NOTE:** FCSS=1 also makes the channel work on T2/T4 timing mode (see T2/T4 Channel Timing). 0 Use eTPU clock / 2 as EDF clock source before prescaler. 1 Use eTPU clock as EDF clock source before prescaler
13–15 FPSCK	Filter Prescaler Clock Control. FPSCK controls the prescaling of the clocks used in digital filters for the channel input signals and TCRCLK input, as shown in the table below. Filtering can be controlled independently by each engine, but all input digital filters in the same engine have same clock prescaling. For more details, see Filter Clock Prescaler. A new value written to FPSCK only becomes effective when the filter prescaler finishes the current count.

Table 43-13. Filter prescaler clock control

Filter control	Sample on system clock divided by:	
	FCSS = 0	FCSS = 1
000	2	1
001	4	2
010	8	4
011	16	8
100	32	16
101	64	32
110	128	64
111	256	128

Table continues on the next page...

eTPU_SCR_ECR_ENG1 field descriptions (continued)

Field	Description
16–17 CDFC	Channel Digital Filter Control. These bits select a digital filtering mode for the channels when configured as inputs for improved noise immunity (refer to the table below). The eTPU has three digital filtering modes for the channels which provide programmable trade-off between signal latency and noise immunity (see Enhanced Digital Filter - EDF). Changing CDFC during eTPU normal input channel operation is not recommended since it changes the behavior of the transition detection logic while executing its operation.
18 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
19–23 ERBA	Engine Relative Base Address. This field value is concatenated with the AID instruction field in engine relative address mode to form the SDM address (see the “Engine Relative Addressing Mode" section under SDM Addressing Modes).
24 SPPDIS	Schedule Priority Passing Disable. SPPDIS is used to disable the priority passing mechanism of the microengine scheduler (see the “Primary Scheme - Priority Among Channels on Different Levels" section under Channel Priority Schemes). **NOTE:** SPPDIS bit must not be changed while any channel is enabled. 0 Scheduler priority passing mechanism enabled. 1 Scheduler priority passing mechanism disabled.
25–26 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–31 ETB	Entry Table Base.

Table 43-12. Channel digital filter control

CDFC	Selected digital filter
00	TPU2/3 Two Sample Mode: Using the filter clock which is the system clock divided by (2, 4, 8,..., 256) as a sampling clock (selected by FPSCK field in ETPUECR), comparing two consecutive samples which agree with each other sets the input signal state. This is the default reset state.
01	eTPU bypass mode: the input signal is taken unfiltered, also making the channels work on T2/T4 timing mode.[1]
10	eTPU Three Sample Mode: Similar to the TPU2/3 two sample mode, but comparing three consecutive samples which agree with each other sets the input signal state.
11	eTPU Continuous Mode: Signal need to be stable for the whole filter clock period. This mode compares all the values at the rate of eTPU clock (FCSS=1) or eTPU clock divided by two (FCSS=0), between two consecutive filter clock pulses. Signal needs to be continuously stable for the entire period. If all the values agree with each other, input signal state is updated.

1. See T2/T4 Channel Timing.

Table continues on the next page...

eTPU_SCR_ECR_ENG1 field descriptions (continued)

Field	Description
	The field determines the location of the microcode entry table for the eTPU functions in SCM (see Entry Points). The following table shows the entry table base address options.

Table 43-11. Entry table base address options

ETB	Entry Table Base Address for CPU Host Address (byte format)	Entry Table Base Address for Microcode Address (word format)
00000	0x000	0x000
00001	0x800	0x200
00010	0x1000	0x400
--------	----------	---------
11110	0xF000	0x3C00
11111	0xF800	0x3E00

3. See T2/T4 Channel Timing.

43.6.6 eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2)

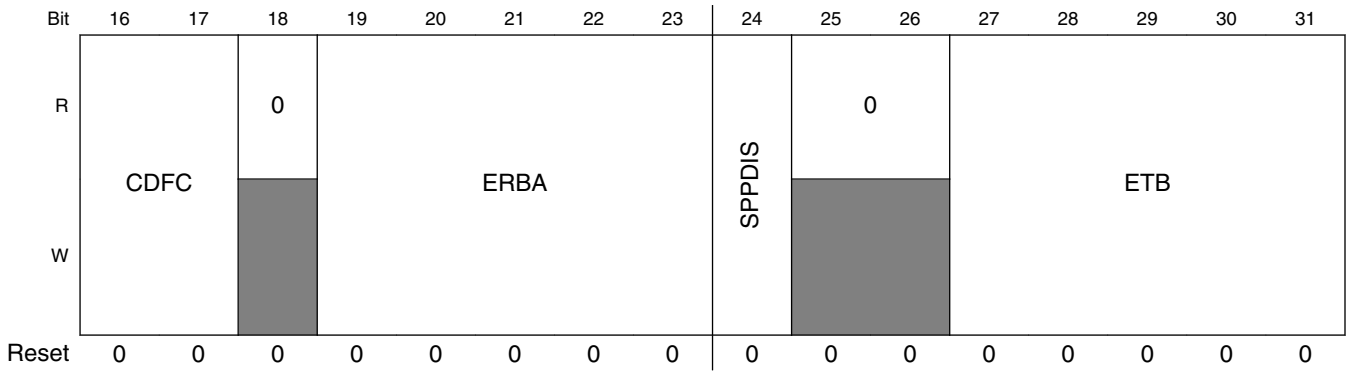
Each engine has its own ETPUECR register. ETPUECR holds configuration and status fields that are programmed independently in each engine.

NOTE

See Register Reset Values in Configuration information.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FEND	MDIS	0	STF	0				HLTF	0			FCSS	FPSCK		
W																
Reset	0	*	0	0	0	0	0	0	*	0	0	0	0	0	0	0



* Notes:
- HLTF field: HLTF field reset value is 1 if, after a reset, the eTPU Nexus debug support is on and a breakpoint request is asserted. Otherwise the reset value is 0.
- MDIS field: Reset value is 1.

eTPU_SCR_ECR_ENG2 field descriptions

Field	Description
0 FEND	Force End. FEND assertion terminates any current running thread as if an END instruction have been executed (see Ending Current Thread - END). This bit is self-negating when the thread ends. **NOTE:** Only on rare occasions (e.g., during a long stall, see Microengine Stall) FEND can be read as 1, because it negates as soon as the end begins execution. Writing FEND = 1 is ignored and FEND stays 0 when the microengine is in TST, halted, stopped, or idle (no thread executing). 0 Normal operation. 1 Ends any ongoing thread.
1 MDIS	Module Disable Bit. When MDIS is set, the Engine shuts down its internal clocks, going into Module Disable Mode. TCR1 and TCR2 cease to increment, and input sampling stops. The Engine asserts the stop flag (STF) bit to indicate that it has stopped. However, the BIU continues to run, and the Host can access all registers except for the channel registers (see channel registers starting with eTPU Channel Configuration-Control registers) **NOTE:** The Timebase registers can still be read with MDIS=1, but writes are ineffective and a Bus Error is issued. Global Channel Registers and SDM can be accessed normally. After MDIS is set, even before STF asserts, data read from the channel registers is not meaningful and writes are ineffective, issuing a Bus Error. When the MDIS bit is asserted while the microcode is executing, the eTPU will stop when the thread is complete. Stop completes on the next eTPU clock after the stop condition is valid. **NOTE:** Once MDIS is switched from 1 to 0 or vice versa, it must not write a different value until STF changes accordingly. The MDIS bit is write-protected when VIS=1. See Register Reset Values in Configuration information.

Table continues on the next page...

eTPU_SCR_ECR_ENG2 field descriptions (continued)

Field	Description
	0 eTPU Engine runs. 1 Commands Engine to stop its clocks.
2 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
3 STF	Stop Flag Bit. Each Engine asserts its stop flag (STF) to indicate that it has stopped. Only then the host can assume that the Engine has actually stopped. The eTPU system is fully stopped when the STF bits of both eTPU Engines are asserted. In case of IP-Bus stop, the eTPU system responds by acknowledging stop only after both eTPU 1 and eTPU 2 have been stopped. Engine only stops when any ongoing thread is complete also in this case. Summarizing Engine stop conditions, which STF reflects: STF_1 := (after stop completed) MDIS_1 | device stop request. STF_2 := (after stop completed) MDIS_2 | device stop request. STF_1 and STF_2 mean STF bit from engine 1 and STF bit from engine 2, respectively. See Register Reset Values in Configuration information. 0 Engine is operating. 1 Engine has stopped (after the local MDIS bit has been asserted, or after the IP-Bus stop line has been asserted).
4–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8 HLTF	Halt Mode Flag. If eTPU Engine entered halt state, this flag is asserted. The flag remains asserted while the microengine is in halt state, even during a single-step or forced instruction execution. See Development Support Features for further details about entering Halt Mode. 0 eTPU Engine is not halted. 1 eTPU Engine is halted.
9–11 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
12 FCSS	Filter Clock Source Selection Speeds up the filter clock source before the prescaler, allowing more input capture resolution at minimum prescaling. **NOTE:** FCSS=1 also makes the channel work on T2/T4 timing mode (see T2/T4 Channel Timing). 0 Use eTPU clock / 2 as EDF clock source before prescaler. 1 Use eTPU clock as EDF clock source before prescaler
13–15 FPSCK	Filter Prescaler Clock Control.

Table continues on the next page...

eTPU_SCR_ECR_ENG2 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>FPSCK controls the prescaling of the clocks used in digital filters for the channel input signals and TCRCLK input, as shown in the table below. Filtering can be controlled independently by each engine, but all input digital filters in the same engine have same clock prescaling. For more details, see Filter Clock Prescaler.

A new value written to FPSCK only becomes effective when the filter prescaler finishes the current count.

Table 43-17. Filter prescaler clock control
<table><tr><th>Filter control</th><th>Sample on system clock divided by:</th></tr><tr><td>000</td><td>2</td></tr><tr><td>001</td><td>4</td></tr><tr><td>010</td><td>8</td></tr><tr><td>011</td><td>16</td></tr><tr><td>100</td><td>32</td></tr><tr><td>101</td><td>64</td></tr><tr><td>110</td><td>128</td></tr><tr><td>111</td><td>256</td></tr></table></td></tr>
<tr><td>16–17
CDFC</td><td>Channel Digital Filter Control.

These bits select a digital filtering mode for the channels when configured as inputs for improved noise immunity (refer to the table below). The eTPU has three digital filtering modes for the channels which provide programmable trade-off between signal latency and noise immunity (see Enhanced Digital Filter - EDF). Changing CDFC during eTPU normal input channel operation is not recommended since it changes the behavior of the transition detection logic while executing its operation.
Table 43-16. Channel digital filter control
<table><tr><th>CDFC</th><th>Selected digital filter</th></tr><tr><td>00</td><td>TPU2/3 Two Sample Mode: Using the filter clock which is the system clock divided by (2, 4, 8,..., 256) as a sampling clock (selected by FPSCK field in ETPUECR), comparing two consecutive samples which agree with each other sets the input signal state. This is the default reset state.</td></tr><tr><td>01</td><td>eTPU bypass mode: the input signal is taken unfiltered, also making the channels work on T2/T4 timing mode.[1]</td></tr><tr><td>10</td><td>eTPU Three Sample Mode: Similar to the TPU2/3 two sample mode, but comparing three consecutive samples which agree with each other sets the input signal state.</td></tr><tr><td>11</td><td>eTPU Continuous Mode: Signal need to be stable for the whole filter clock period. This mode compares all the values at the rate of eTPU clock (FCSS=1) or eTPU clock divided by two (FCSS=0), between two consecutive filter clock pulses. Signal needs to be continuously stable for the entire period. If all the values agree with each other, input signal state is updated.</td></tr></table>
1. See T2/T4 Channel Timing.</td></tr>
<tr><td>18
Reserved</td><td>Reserved.

This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
</table>

Table continues on the next page...

eTPU_SCR_ECR_ENG2 field descriptions (continued)

Field	Description
19–23 ERBA	Engine Relative Base Address. This field value is concatenated with the AID instruction field in engine relative address mode to form the SDM address (see the “Engine Relative Addressing Mode" section under SDM Addressing Modes).
24 SPPDIS	Schedule Priority Passing Disable. SPPDIS is used to disable the priority passing mechanism of the microengine scheduler (see the “Primary Scheme - Priority Among Channels on Different Levels" section under Channel Priority Schemes). **NOTE:** SPPDIS bit must not be changed while any channel is enabled. 0 Scheduler priority passing mechanism enabled. 1 Scheduler priority passing mechanism disabled.
25–26 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–31 ETB	Entry Table Base. The field determines the location of the microcode entry table for the eTPU functions in SCM (see Entry Points). The following table shows the entry table base address options.

Table 43-15. Entry table base address options

ETB	Entry Table Base Address for CPU Host Address (byte format)	Entry Table Base Address for Microcode Address (word format)
00000	0x000	0x000
00001	0x800	0x200
00010	0x1000	0x400
--------	----------	---------
11110	0xF000	0x3C00
11111	0xF800	0x3E00

3. See T2/T4 Channel Timing.

43.7 Timebase memory map and registers

Time Base registers allows configuration and visibility of internally-generated time bases TCR1 and TCR2. There is one of each of these registers for each eTPU Engine.

NOTE

Writes to this register issue bus error and are ineffective when MDIS=1. Reads are always allowed.

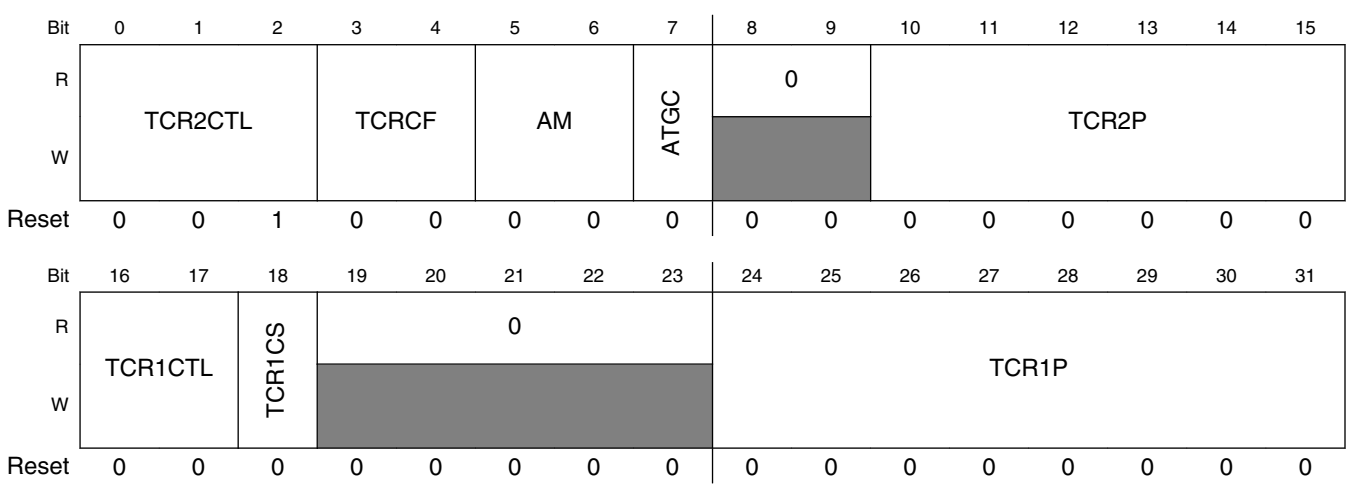
eTPU_TBR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20	eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1)	32	R/W	2000_0000h	43.7.1/1363
24	eTPU Engine 1 Time Base 1 (TCR1) Visibility Register (eTPU_TBR_TB1R_ENG1)	32	R/W	0000_0000h	43.7.2/1367
28	eTPU Engine 1 Time Base 2 (TCR2) Visibility Register (eTPU_TBR_TB2R_ENG1)	32	R/W	0000_0000h	43.7.3/1367
2C	eTPU Engine 1 STAC Configuration Register (eTPU_TBR_STAC_ENG1)	32	R/W	See section	43.7.4/1368
40	eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2)	32	R/W	2000_0000h	43.7.5/1370
44	eTPU Engine 2 Time Base 1 (TCR1) Visibility Register (eTPU_TBR_TB1R_ENG2)	32	R/W	0000_0000h	43.7.6/1374
48	eTPU Engine 2 Time Base 2 (TCR2) Visibility Register (eTPU_TBR_TB2R_ENG2)	32	R/W	0000_0000h	43.7.7/1374
4C	eTPU Engine 2 STAC Configuration Register (eTPU_TBR_STAC_ENG2)	32	R/W	See section	43.7.8/1375

43.7.1 eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1)

This register configures several timebase options.

Address: 0h base + 20h offset = 20h



eTPU_TBR_TBCR_ENG1 field descriptions

Field	Description
0–2 TCR2CTL	TCR2 Clock/Gate Control.

Table continues on the next page...

eTPU_TBR_TBCR_ENG1 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>These bits are part of the TCR2 clocking system (see Time Bases). They determine the clock source for TCR2 before the prescaler. TCR2 can count on any detected edge of the TCRCLK signal or use it for gating eTPU clock divided by 8. After reset - TCRCLK signal rising edge is selected. TCR2 can also be clocked by the eTPU clock divided by 8. TCR2CTL also determines the TCRCLK edge selected for angle tooth detection in angle mode. See the following table.
Table 43-23. TCR2 Clock Source
<table>
<tr><th>TCR2CTL</th><th>AM = 0
TCR2 Clock before prescaler</th><th>AM > 0
Angle Tooth detection</th></tr>
<tr><td>000</td><td>Gated DIV8 clock (eTPU clock / 8). In this case, when the external TCRCLK signal is low, the DIV8 clock is blocked, preventing it from incrementing the TCR2 prescaler. When the external TCRCLK signal is high, TCR2 prescaler is incremented at the frequency of the eTPU clock divided by 8.</td><td>Do not use with AM > 0</td></tr>
<tr><td>001</td><td>Rise transition on TCRCLK signal increments the TCR2 prescaler.</td><td>Rising edge</td></tr>
<tr><td>010</td><td>Fall transition on TCRCLK signal increments the TCR2 prescaler.</td><td>Falling edge</td></tr>
<tr><td>011</td><td>Rise or Fall transition on TCRCLK signal increments the TCR2 prescaler.</td><td>Both edges</td></tr>
<tr><td>100</td><td>DIV8 clock (eTPU clock / 8)</td><td>Do not use with AM > 0</td></tr>
<tr><td>101</td><td>Do not use with AM=0</td><td>Do not use with AM > 0</td></tr>
<tr><td>110</td><td>Do not use with AM=0</td><td>No edge[1]</td></tr>
<tr><td>111</td><td>TCR2 frozen, except as STAC client</td><td>Do not use with AM > 0</td></tr>
</table>
1. TCRCLK edges are not detected by the EAC logic, but they can still be detected by the channel 0 logic if AM=01.</td></tr>
<tr><td>3–4
TCRCF</td><td>TCRCLK Signal Filter Control.

This field controls the TCRCLK digital filter (see TCRCLK Digital Filter), determining whether the TCRCLK signal input (after a synchronizer) is filtered with the same filter clock as the channel input signals (see Enhanced Digital Filter - EDF) or uses the eTPU clock divided by 2, and also whether the TCRCLK digital filter works in integrator mode or two sample mode (see the following table).
Table 43-22. TCRCLK Filter Clock/Mode
<table>
<tr><th>TCRCF</th><th>Filter Clock</th><th>Filter Mode</th></tr>
<tr><td>00</td><td>eTPU clock divided by 2</td><td>Two sample</td></tr>
<tr><td>01</td><td>Filter clock of the channels</td><td>Two sample</td></tr>
<tr><td>10</td><td>eTPU clock divided by 2</td><td>Integrator</td></tr>
<tr><td>11</td><td>Filter clock of the channels</td><td>Integrator</td></tr>
</table></td></tr>
<tr><td>5–6
AM</td><td>Angle Mode Selection.

This field enables the Enhanced Angle Counter logic to generate angle information (see EAC - eTPU Angle Counter), and also select the tooth signal input and the channel used to process it, as shown in the following table. When EAC is not disabled by AM and neither TCR1 nor TCR2 are STAC Clients, the EAC</td></tr>
</table>

Table continues on the next page...

eTPU_TBR_TBCR_ENG1 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>(eTPU Angle Clock) hardware provides angle information to the channels using the TCR2 bus. When AM is reset (non-angle mode), the EAC operation is disabled, and its internal registers can be used as general purpose. For more information, see EAC - eTPU Angle Counter.
Table 43-21. AM - Angle Mode Selection
<table>
<tr><th>Value</th><th>TCR2 Value</th><th>Tooth signal</th><th>Tooth processing channel</th></tr>
<tr><td>00</td><td>Timebase (EAC operation disabled)</td><td colspan="2">Not applicable</td></tr>
<tr><td>01</td><td rowspan="3">Angle Ticks</td><td>TCRCLK input</td><td>0</td></tr>
<tr><td>10</td><td>Channel 1 input</td><td>1</td></tr>
<tr><td>11</td><td>Channel 2 input</td><td>2</td></tr>
</table>
If TCR1 or TCR2 is a STAC Bus Client (see STAC Interface), the EAC operation is not allowed, and AM must be set to 0.

NOTE: Changing AM may cause unexpected transition detections on the channel selected by AM, depending on the channel mode and state (see Transition Detection and Time Base Capture). If AM must be changed with GTBE=1, the recommended procedure is described in Restarting Angle Logic.</td></tr>
<tr><td>7
ATGC</td><td>Angle Tick Generator Clock.

This bit selects the clock that provides the time interval unit for Angle Tick Generation. For details, see Generating the Angle Ticks.

0 TCR1 prescaler output provides time intervals for tick generation.
1 TCR2 prescaler output provides time intervals for tick generation.</td></tr>
<tr><td>8–9
Reserved</td><td>Reserved.

This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>10–15
TCR2P</td><td>Timer Count Register 2 Prescaler Control.

These bits are part of the TCR2 clocking system (see Time Bases). TCR2 is clocked from the output of a prescaler. The prescaler divides its input by (TCR2P + 1) allowing frequency divisions from 1 to 64. The prescaler input is the eTPU clock divided by 8 (in gated or non-gated clock mode), or Internal Timebase input, or TCRCLK filtered input. This field has no effect on TCR2 in Angle Mode.</td></tr>
<tr><td>16–17
TCR1CTL</td><td>TCR1 Clock/Gate Control.

TCR1CTL is part of the TCR1 clocking system (see Time Bases). It determines, together with TCR1CS, the clock source for TCR1. TCR1 can count on detected rising edge of the TCRCLK signal, the eTPU clock, or the eTPU clock divided by 2 (see the table in the TCR1CS bit description). After reset TCRCLK signal is selected.</td></tr>
<tr><td>18
TCR1CS</td><td>TCR1 Clock Source

TCR1CS provides the option to double the TCR1 incrementing speed, using eTPU clock as its clock source instead of eTPU clock / 2.</td></tr>
</table>

Table continues on the next page...

eTPU_TBR_TBCR_ENG1 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
NOTE: If ATGC=0, the clock source of the EAC angle tick generator will still be an even division of eTPU clock if TCR1CS=1, obeying to the field TCR1P as if TCR1CS=0 (see Angle Tick Generator).
Table 43-20. TCR1 Clock Source
<table>
<tr><th>TCR1CTL</th><th>TCR1CS</th><th>TCR1 Clock before prescaler</th></tr>
<tr><td>00</td><td>0</td><td>Selects TCRCLK as clock source for the TCR1 prescaler.[1]</td></tr>
<tr><td>01</td><td>0</td><td>Reserved</td></tr>
<tr><td>01</td><td>1</td><td>Reserved</td></tr>
<tr><td>10</td><td>0</td><td>Selects eTPU clock divided by 2 as clock source for the TCR1 prescaler.</td></tr>
<tr><td>10</td><td>1</td><td>Selects eTPU clock as clock source for the TCR1 prescaler.</td></tr>
<tr><td>11</td><td>0</td><td>TCR1 frozen, except as a STAC client</td></tr>
</table>
1. This selection must not be used in Angle Mode.

0 Use eTPU clock / 2 as TCR1 clock source before the prescaler, if that clock source is selected by TCR1CTL.

1 Use eTPU clock as TCR1 clock source before the prescaler; can only be set in specific combinations with TCR1CTL, as shown in the table.

NOTE: TCR1CS=1 also makes the channel work on T2/T4 timing mode (see T2/T4 Channel Timing).
</td></tr>
<tr><td>19–23
Reserved</td><td>Reserved.

This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>24–31
TCR1P</td><td>Timer Count Register 1 Prescaler Control.

TCR1 is clocked from the output of a prescaler. The input to the prescaler is the internal eTPU clock divided by 2, eTPU clock, or the output of TCRCLK filter. The prescaler divides this input by (TCR1P + 1) allowing frequency divisions from 1 up to 256.</td></tr>
</table>

1. TCRCLK edges are not detected by the EAC logic, but they can still be detected by the channel 0 logic if AM=01.
2. This selection must not be used in Angle Mode.

43.7.2 eTPU Engine 1 Time Base 1 (TCR1) Visibility Register (eTPU_TBR_TB1R_ENG1)

This register provides visibility of the TCR1 time base for host read access (see Time Bases). This register is read-only. The value of the TCR1 time base shown can be driven by the TCR1 counter or imported from STAC bus, depending on the configuration set in REDCR_ENG1.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								TCR1																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_TBR_TB1R_ENG1 field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 TCR1	TCR1 value. TCR1 value used on matches and captures. See Time Bases.

43.7.3 eTPU Engine 1 Time Base 2 (TCR2) Visibility Register (eTPU_TBR_TB2R_ENG1)

This register provides visibility of the TCR2 time base for host read access (see Time Bases). This register is read-only. The value of the TCR2 time base shown can be driven by the TCR2 counter or imported from the STAC bus, depending on Angle Mode and STAC configurations set in ETPUREDCR_ENG1.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								TCR2																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_TBR_TB2R_ENG1 field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 TCR2	TCR2 value. TCR2 value used on matches and captures. See Time Bases.

43.7.4 eTPU Engine 1 STAC Configuration Register (eTPU_TBR_STAC_ENG1)

This register configures the eTPU STAC bus operation as a STAC Server/Client module (see STAC Interface).

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	REN1	RSC1	0		SERVER_ID1				0				SRV1			
W																
Reset	0	0	0	0	*	*	*	*	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	REN2	RSC2	0		SERVER_ID2				0				SRV2			
W																
Reset	0	0	0	0	*	*	*	*	0	0	0	0	0	0	0	0

* Notes:
- SERVER_ID2 field: See the chip-specific eTPU information for the reset value.
- SERVER_ID1 field: See the chip-specific eTPU information for the reset value.

eTPU_TBR_STAC_ENG1 field descriptions

Field	Description
0 REN1	TCR1 Resource Client/Server Operation Enable Bit. This bit enables Client/Server operation to eTPU STAC resources. REN1 enables TCR1 STAC bus operations. 0 Server/Client Operation for TCR1 is disabled. 1 Server/Client Operation for TCR1 is enabled.
1 RSC1	TCR1 Resource Server/Client Assignment Bits. This bit selects the eTPU data resource assignment to be used as Servers or Clients. For Server mode, MCU-dependent value determines the unique server address assigned to each TCR. See the chip-specific

Table continues on the next page...

eTPU_TBR_STAC_ENG1 field descriptions (continued)

Field	Description
	eTPU information for the STAC server IDs. For a Client mode, the SRV1 field determines the Server address to which the Client listens. **NOTE:** When TCR1 or TCR2 is configured as a STAC Bus Client (REN1/2 = 1, RSC1/2 = 0) the eTPU Angle Clock hardware cannot be used. **NOTE:** RSC1 must not be changed when the REN1 bit is asserted. 0 Resource Client operation. 1 Resource Server operation.
2–3 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SERVER_ID1	STAC ID 1. STAC Server ID= 0 (read-only MCU-dependent values; see the chip-specific eTPU information for the STAC server IDs) used for TCR1 when STAC servers are used.
8–11 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
12–15 SRV1	TCR1 Resource Server. This bit selects the address of the specific STAC Server to which the local TCR1 listens when configured as a STAC Client. SRV1 selects the STAC Server of TCR1.
16 REN2	TCR2 Resource Client/Server Operation Enable Bit. This bit enables Client/Server operation to eTPU STAC resources. REN2 enables TCR2 STAC bus operations. 0 Server/Client Operation for TCR2 is disabled. 1 Server/Client Operation for TCR2 is enabled.
17 RSC2	TCR2 Resource Server/Client Assignment Bits. **NOTE:** Either time or angle TCR2 values. 0 Resource Client operation. 1 Resource Server operation.
18–19 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
20–23 SERVER_ID2	STAC ID 2. STAC Server ID= 2 (read-only MCU-dependent values; see the chip-specific eTPU information for the STAC server IDs) used for TCR2 when when STAC servers are used.
24–27 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

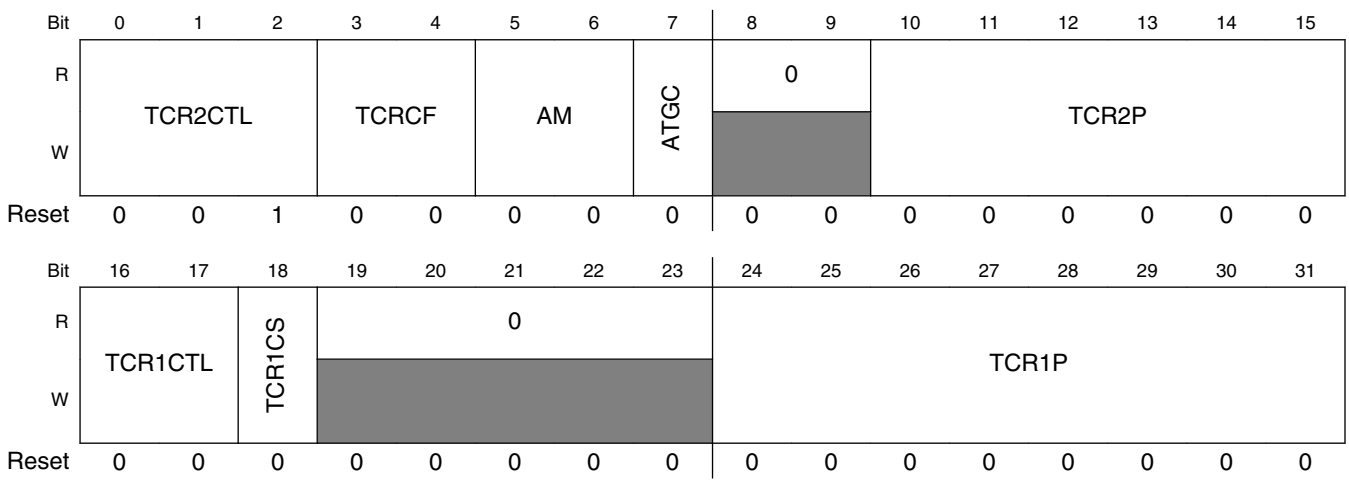
eTPU_TBR_STAC_ENG1 field descriptions (continued)

Field	Description
28–31 SRV2	TCR2 Resource Server. This bit selects the address of the specific STAC Server to which the local TCR2 listens when configured as a STAC Client. SRV2 selects the Red Line STAC Server of TCR2.

43.7.5 eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2)

This register configures several timebase options.

Address: 0h base + 40h offset = 40h



eTPU_TBR_TBCR_ENG2 field descriptions

Field	Description
0–2 TCR2CTL	TCR2 Clock/Gate Control. These bits are part of the TCR2 clocking system (see Time Bases). They determine the clock source for TCR2 before the prescaler. TCR2 can count on any detected edge of the TCRCLK signal or use it for gating eTPU clock divided by 8. After reset - TCRCLK signal rising edge is selected. TCR2 can also be

Table continues on the next page...

eTPU_TBR_TBCR_ENG2 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>clocked by the eTPU clock divided by 8. TCR2CTL also determines the TCRCLK edge selected for angle tooth detection in angle mode. See the following table.
Table 43-31. TCR2 Clock Source
<table>
<tr><th>TCR2CTL</th><th>AM = 0
TCR2 Clock before prescaler</th><th>AM > 0
Angle Tooth detection</th></tr>
<tr><td>000</td><td>Gated DIV8 clock (eTPU clock / 8). In this case, when the external TCRCLK signal is low, the DIV8 clock is blocked, preventing it from incrementing the TCR2 prescaler. When the external TCRCLK signal is high, TCR2 prescaler is incremented at the frequency of the eTPU clock divided by 8.</td><td>Do not use with AM > 0</td></tr>
<tr><td>001</td><td>Rise transition on TCRCLK signal increments the TCR2 prescaler.</td><td>Rising edge</td></tr>
<tr><td>010</td><td>Fall transition on TCRCLK signal increments the TCR2 prescaler.</td><td>Falling edge</td></tr>
<tr><td>011</td><td>Rise or Fall transition on TCRCLK signal increments the TCR2 prescaler.</td><td>Both edges</td></tr>
<tr><td>100</td><td>DIV8 clock (eTPU clock / 8)</td><td>Do not use with AM > 0</td></tr>
<tr><td>101</td><td>Do not use with AM=0</td><td>Do not use with AM > 0</td></tr>
<tr><td>110</td><td>Do not use with AM=0</td><td>No edge[1]</td></tr>
<tr><td>111</td><td>TCR2 frozen, except as STAC client</td><td>Do not use with AM > 0</td></tr>
</table>
1. TCRCLK edges are not detected by the EAC logic, but they can still be detected by the channel 0 logic if AM=01.</td></tr>
<tr><td>3–4
TCRCF</td><td>TCRCLK Signal Filter Control.

This field controls the TCRCLK digital filter (see TCRCLK Digital Filter, determining whether the TCRCLK signal input (after a synchronizer) is filtered with the same filter clock as the channel input signals (see Enhanced Digital Filter - EDF) or uses the eTPU clock divided by 2, and also whether the TCRCLK digital filter works in integrator mode or two sample mode (see the following table).
Table 43-30. TCRCLK Filter Clock/Mode
<table>
<tr><th>TCRCF</th><th>Filter Clock</th><th>Filter Mode</th></tr>
<tr><td>00</td><td>eTPU clock divided by 2</td><td>Two sample</td></tr>
<tr><td>01</td><td>Filter clock of the channels</td><td>Two sample</td></tr>
<tr><td>10</td><td>eTPU clock divided by 2</td><td>Integrator</td></tr>
<tr><td>11</td><td>Filter clock of the channels</td><td>Integrator</td></tr>
</table></td></tr>
<tr><td>5–6
AM</td><td>Angle Mode Selection.

This field enables the Enhanced Angle Counter logic to generate angle information (see EAC - eTPU Angle Counter), and also select the tooth signal input and the channel used to process it, as shown in the following table. When EAC is not disabled by AM and neither TCR1 nor TCR2 are STAC Clients, the EAC (eTPU Angle Clock) hardware provides angle information to the channels using the TCR2 bus. When AM</td></tr>
</table>

Table continues on the next page...

eTPU_TBR_TBCR_ENG2 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>is reset (non-angle mode), the EAC operation is disabled, and its internal registers can be used as general purpose. For more information, see EAC - eTPU Angle Counter.
Table 43-29. AM - Angle Mode Selection
<table>
<tr><th>Value</th><th>TCR2 Value</th><th>Tooth signal</th><th>Tooth processing channel</th></tr>
<tr><td>00</td><td>Timebase (EAC operation disabled)</td><td colspan="2">Not applicable</td></tr>
<tr><td>01</td><td rowspan="3">Angle Ticks</td><td>TCRCLK input</td><td>0</td></tr>
<tr><td>10</td><td>Channel 1 input</td><td>1</td></tr>
<tr><td>11</td><td>Channel 2 input</td><td>2</td></tr>
</table>
If TCR1 or TCR2 is a STAC Bus Client (see STAC Interface), the EAC operation is not allowed, and AM must be set to 0.

NOTE: Changing AM may cause unexpected transition detections on the channel selected by AM, depending on the channel mode and state (see Transition Detection and Time Base Capture). If AM must be changed with GTBE=1, the recommended procedure is described in Restarting Angle Logic.</td></tr>
<tr><td>7
ATGC</td><td>Angle Tick Generator Clock.

This bit selects the clock that provides the time interval unit for Angle Tick Generation. For details, see Generating the Angle Ticks.

0 TCR1 prescaler output provides time intervals for tick generation.
1 TCR2 prescaler output provides time intervals for tick generation.</td></tr>
<tr><td>8–9
Reserved</td><td>Reserved.

This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>10–15
TCR2P</td><td>Timer Count Register 2 Prescaler Control.

These bits are part of the TCR2 clocking system (see Time Bases). TCR2 is clocked from the output of a prescaler. The prescaler divides its input by (TCR2P + 1) allowing frequency divisions from 1 to 64. The prescaler input is the eTPU clock divided by 8 (in gated or non-gated clock mode), or Internal Timebase input, or TCRCLK filtered input. This field has no effect on TCR2 in Angle Mode.</td></tr>
<tr><td>16–17
TCR1CTL</td><td>TCR1 Clock/Gate Control.

TCR1CTL is part of the TCR1 clocking system (see Time Bases). It determines, together with TCR1CS, the clock source for TCR1. TCR1 can count on detected rising edge of the TCRCLK signal, the eTPU clock, or the eTPU clock divided by 2 (see the table in the TCR1CS bit description). After reset TCRCLK signal is selected.</td></tr>
<tr><td>18
TCR1CS</td><td>TCR1 Clock Source

TCR1CS provides the option to double the TCR1 incrementing speed, using eTPU clock as its clock source instead of eTPU clock / 2.</td></tr>
</table>

Table continues on the next page...

eTPU_TBR_TBCR_ENG2 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
NOTE: If ATGC=0, the clock source of the EAC angle tick generator will still be an even division of eTPU clock if TCR1CS=1, obeying to the field TCR1P as if TCR1CS=0 (see Angle Tick Generator).
Table 43-28. TCR1 Clock Source
<table>
<tr><th>TCR1CTL</th><th>TCR1CS</th><th>TCR1 Clock before prescaler</th></tr>
<tr><td>00</td><td>0</td><td>Selects TCRCLK as clock source for the TCR1 prescaler.[1]</td></tr>
<tr><td>01</td><td>0</td><td>Reserved</td></tr>
<tr><td>01</td><td>1</td><td>Reserved</td></tr>
<tr><td>10</td><td>0</td><td>Selects eTPU clock divided by 2 as clock source for the TCR1 prescaler.</td></tr>
<tr><td>10</td><td>1</td><td>Selects eTPU clock as clock source for the TCR1 prescaler.</td></tr>
<tr><td>11</td><td>0</td><td>TCR1 frozen, except as a STAC client</td></tr>
</table>
1. This selection must not be used in Angle Mode.

0 Use eTPU clock / 2 as TCR1 clock source before the prescaler, if that clock source is selected by TCR1CTL.

1 Use eTPU clock as TCR1 clock source before the prescaler; can only be set in specific combinations with TCR1CTL, as shown in the table.

NOTE: TCR1CS=1 also makes the channel work on T2/T4 timing mode (see T2/T4 Channel Timing).
</td></tr>
<tr><td>19–23
Reserved</td><td>Reserved.

This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>24–31
TCR1P</td><td>Timer Count Register 1 Prescaler Control.

TCR1 is clocked from the output of a prescaler. The input to the prescaler is the internal eTPU clock divided by 2, eTPU clock, or the output of TCRCLK filter. The prescaler divides this input by (TCR1P + 1) allowing frequency divisions from 1 up to 256.</td></tr>
</table>

1. TCRCLK edges are not detected by the EAC logic, but they can still be detected by the channel 0 logic if AM=01.
2. This selection must not be used in Angle Mode.

43.7.6 eTPU Engine 2 Time Base 1 (TCR1) Visibility Register (eTPU_TBR_TB1R_ENG2)

This register provides visibility of the TCR1 time base for host read access (see Time Bases). This register is read-only. The value of the TCR1 time base shown can be driven by the TCR1 counter or imported from STAC bus, depending on the configuration set in REDCR_ENG2.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								TCR1																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_TBR_TB1R_ENG2 field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 TCR1	TCR1 value. TCR1 value used on matches and captures. See Time Bases.

43.7.7 eTPU Engine 2 Time Base 2 (TCR2) Visibility Register (eTPU_TBR_TB2R_ENG2)

This register provides visibility of the TCR2 time base for host read access (see Time Bases). This register is read-only. The value of the TCR2 time base shown can be driven by the TCR2 counter or imported from the STAC bus, depending on Angle Mode and STAC configurations set in ETPUREDCR_ENG2.

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								TCR2																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_TBR_TB2R_ENG2 field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 TCR2	TCR2 value. TCR2 value used on matches and captures. See Time Bases.

43.7.8 eTPU Engine 2 STAC Configuration Register (eTPU_TBR_STAC_ENG2)

This register configures the eTPU STAC bus operation as a STAC Server/Client module (see STAC Interface).

Address: 0h base + 4Ch offset = 4Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	REN1	RSC1	0		SERVER_ID1				0				SRV1			
W																
Reset	0	0	0	0	*	*	*	*	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	REN2	RSC2	0		SERVER_ID2				0				SRV2			
W																
Reset	0	0	0	0	*	*	*	*	0	0	0	0	0	0	0	0

* Notes:
- SERVER_ID2 field: See the chip-specific eTPU information for the reset value.
- SERVER_ID1 field: See the chip-specific eTPU information for the reset value.

eTPU_TBR_STAC_ENG2 field descriptions

Field	Description
0 REN1	TCR1 Resource Client/Server Operation Enable Bit. This bit enables Client/Server operation to eTPU STAC resources. REN1 enables TCR1 STAC bus operations. 0 Server/Client Operation for TCR1 is disabled. 1 Server/Client Operation for TCR1 is enabled.
1 RSC1	TCR1 Resource Server/Client Assignment Bits. This bit selects the eTPU data resource assignment to be used as Servers or Clients. For Server mode, MCU-dependent value determines the unique server address assigned to each TCR. See the chip-specific

Table continues on the next page...

eTPU_TBR_STAC_ENG2 field descriptions (continued)

Field	Description
	eTPU information for the STAC server IDs. For a Client mode, the SRV1 field determines the Server address to which the Client listens. **NOTE:** Resource identifies any parameter that changes along the time and can be exported / imported from other device. In eTPU context, a resource can be TCR1 or TCR2 (either Time or Angle values). **NOTE:** When TCR1 or TCR2 is configured as a STAC Bus Client (REN2 = 1, RSC2 = 0) the eTPU Angle Clock hardware cannot be used. **NOTE:** RSC1 must not be changed when the REN1 bit is asserted. 0 TCR1 Client operation. 1 TCR1 Server operation.
2–3 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SERVER_ID1	STAC ID 1. STAC Server ID= 1 (read-only MCU-dependent values; see the chip-specific eTPU information for the STAC server IDs) used for TCR1 when STAC servers are used.
8–11 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
12–15 SRV1	TCR1 Resource Server. This bit selects the address of the specific STAC Server to which the local TCR1 listens when configured as a STAC Client. SRV1 selects the STAC Server of TCR1.
16 REN2	TCR2 Resource Client/Server Operation Enable Bit. This bit enables Client/Server operation to eTPU STAC resources. REN2 enables TCR2 STAC bus operations. 0 Server/Client Operation for resource 2 is disabled. 1 Server/Client Operation for resource 2 is enabled.
17 RSC2	TCR2 Resource Server/Client Assignment Bits. **NOTE:** Resource identifies any parameter that changes along the time and can be exported / imported from other device. In eTPU context, a resource can be TCR1 or TCR2 (either Time or Angle values). 0 Resource Client operation. 1 Resource Server operation.
18–19 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
20–23 SERVER_ID2	STAC ID 2.

Table continues on the next page...

eTPU_TBR_STAC_ENG2 field descriptions (continued)

Field	Description
	STAC Server ID= 3 (read-only MCU-dependent values; see the chip-specific eTPU information for the STAC server IDs) used for TCR2 when when STAC servers are used.
24–27 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
28–31 SRV2	TCR2 Resource Server. This bit selects the address of the specific STAC Server to which the local TCR2 listens when configured as a STAC Client. SRV2 selects the Red Line STAC Server of TCR2.

43.8 Engine memory map and registers

This section gathers registers that are engine-related, other than ETPUECR (see ECR_ENG1 and ECR_ENG2 registers).

eTPU_ER memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
60	eTPU Watchdog Timer Register for Engine 1 (eTPU_ER_WDTR_ENG1)	32	R/W	0000_0000h	43.8.1/1377
68	eTPU Idle Register for Engine 1 (eTPU_ER_IDLER_ENG1)	32	R/W	Undefined	43.8.2/1379
70	eTPU Watchdog Timer Register for Engine 2 (eTPU_ER_WDTR_ENG2)	32	R/W	0000_0000h	43.8.3/1380
78	eTPU Idle Register for Engine 2 (eTPU_ER_IDLER_ENG2)	32	R/W	Undefined	43.8.4/1381

43.8.1 eTPU Watchdog Timer Register for Engine 1 (eTPU_ER_WDTR_ENG1)

This register configures the watchdog timer for the engine.

Address: 0h base + 60h offset = 60h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WDM		0	WMO	0											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

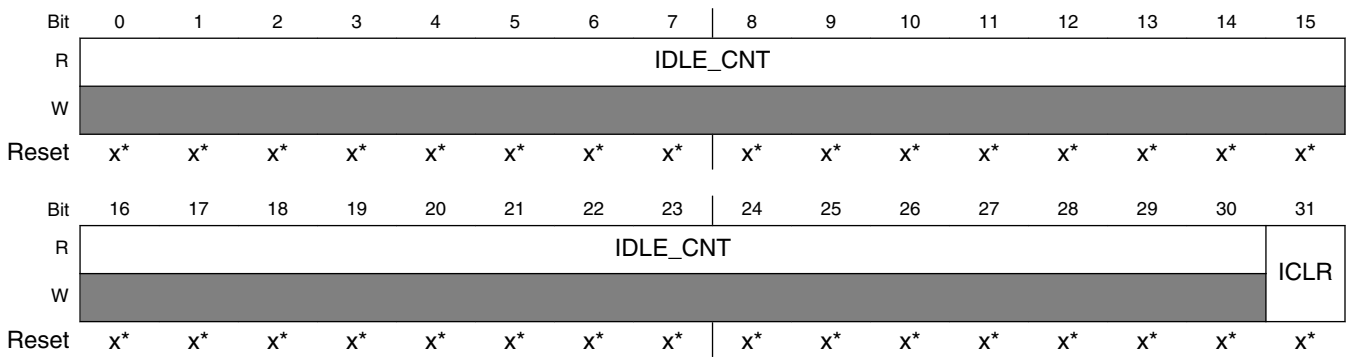
eTPU_ER_WDTR_ENG1 field descriptions

Field	Description
0–1 WDM	Watchdog Mode. WDM selects the Watchdog operation mode. For more information on the Watchdog operation, see Watchdog). **NOTE:** Before a new mode is configured, all conditions below must apply: 1. All channels must be disabled (ETPUCnCR field CPR=00. 2. No thread must be executing (register ETPUCSSR reads 0). 3. The watchdog must be disabled (WDM=00). 00 Disabled 01 Reserved 10 Thread length 11 Busy length
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 WMO	Watchdog Monitor Operation. This bit enables the monitor operation of the watchdog, inhibiting the thread ending and channel disabling in case of a timeout. The Global Exception occurs on a watchdog timeout, independently of WMO. **NOTE:** The ETPUWDSR bits WDSn, the ETPUMCR bits WDTO1/2, and the Global Exception always asserts in case of a watchdog timeout, independently of WMO. 0 Force the end of the running thread and disable the service channel in case of a timeout. 1 Do not force the end of the running thread, nor disable the service channel in case of a timeout.
4–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 WDCNT	Watchdog Count. This field indicates the maximum number of microcycles allowed for a thread (in thread length mode) or a sequence of threads (in busy length mode) before the current running thread is forced to end. For more information on Watchdog operation, see Watchdog. **NOTE:** The TST microcycles are also counted by the watchdog. **NOTE:** Fields WDCNT and WMO must not be changed when the watchdog is enabled.

43.8.2 eTPU Idle Register for Engine 1 (eTPU_ER_IDLER_ENG1)

This register counts the microcycles in which the microengine is idle (see Idle Counter

Address: 0h base + 68h offset = 68h



* Notes:
- x = Undefined at reset.

eTPU_ER_IDLER_ENG1 field descriptions

Field	Description
0–30 IDLE_CNT	Idle Count. This is a freerunning count of the number of idle microcycles in the microengine. For more information on idle counter operation, see Idle Counter. **NOTE:** When the register is read, the ICLR bit is concatenated with the IDLE_CNT field (as the LSB) to form a 32-bit value.
31 ICLR	Idle Clear. Writing a 1 to this bit clears the IDLE_CNT field. **NOTE:** When the register is read, the ICLR bit is concatenated with the IDLE_CNT field (as the LSB) to form a 32-bit value. 0 Do not clear the IDLE_CNT field. 1 Clear the IDLE_CNT field.

43.8.3 eTPU Watchdog Timer Register for Engine 2 (eTPU_ER_WDTR_ENG2)

This register configures the watchdog timer for the engine.

Address: 0h base + 70h offset = 70h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WDM		0	WMO	0											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	WDCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_ER_WDTR_ENG2 field descriptions

Field	Description
0–1 WDM	Watchdog Mode. WDM selects the Watchdog operation mode. For more information on the Watchdog operation, see Watchdog. **NOTE:** Before a new mode is configured, all conditions below must apply: 1. All channels must be disabled (ETPUCnCR field CPR=00. 2. No thread must be executing (register ETPUCSSR reads 0). 3. The watchdog must be disabled (WDM=00). 00 Disabled 01 Reserved 10 Thread length 11 Busy length
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 WMO	Watchdog Monitor Operation. This bit enables the monitor operation of the watchdog, inhibiting the thread ending and channel disabling in case of a timeout. The Global Exception occurs on a watchdog timeout, independently of WMO. **NOTE:** The ETPUWDSR bits WDSn, the ETPUMCR bits WDTO1/2, and the Global Exception always assert in case of a watchdog timeout, independently of WMO. 0 Force the end of the running thread and disable the service channel in case of a timeout. 1 Do not force the end of the running thread, nor disable the service channel in case of a timeout.
4–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 WDCNT	Watchdog Count.

Table continues on the next page...

eTPU_ER_WDTR_ENG2 field descriptions (continued)

Field	Description
	This field indicates the maximum number of microcycles allowed for a thread (in thread length mode) or a sequence of threads (in busy length mode) before the current running thread is forced to end.For more information on Watchdog operation, see Watchdog. **NOTE:** The TST microcycles are also counted by the watchdog. **NOTE:** Fields WDCNT and WMO must not be changed when the watchdog is enabled.

43.8.4 eTPU Idle Register for Engine 2 (eTPU_ER_IDLER_ENG2)

This register counts the microcycles in which the microengine is idle (see Idle Counter).

Address: 0h base + 78h offset = 78h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IDLE_CNT															
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IDLE_CNT															ICLR
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

eTPU_ER_IDLER_ENG2 field descriptions

Field	Description
0–30 IDLE_CNT	Idle Count. This is a freerunning count of the number of idle microcycles in the microengine. For more information on idle counter operation, see Idle Counter. **NOTE:** When the register is read, the ICLR bit is concatenated with the IDLE_CNT field (as the LSB) to form a 32-bit value.
31 ICLR	Idle Clear. Writing a 1 to this bit clears the IDLE_CNT field. **NOTE:** When the register is read, the ICLR bit is concatenated with the IDLE_CNT field (as the LSB) to form a 32-bit value. 0 Do not clear the IDLE_CNT field. 1 Clear the IDLE_CNT field.

43.9 Latency Monitor memory map and registers

eTPU_LMR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
80	eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM0CR_ENG1)	32	R/W	0000_0000h	43.9.1/1383
84	eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM0MR_ENG1)	32	R/W	0000_0000h	43.9.2/1385
88	eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM1CR_ENG1)	32	R/W	0000_0000h	43.9.1/1383
8C	eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM1MR_ENG1)	32	R/W	0000_0000h	43.9.2/1385
90	eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM2CR_ENG1)	32	R/W	0000_0000h	43.9.1/1383
94	eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM2MR_ENG1)	32	R/W	0000_0000h	43.9.2/1385
98	eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM3CR_ENG1)	32	R/W	0000_0000h	43.9.1/1383
9C	eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM3MR_ENG1)	32	R/W	0000_0000h	43.9.2/1385
A0	eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM4CR_ENG1)	32	R/W	0000_0000h	43.9.1/1383
A4	eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM4MR_ENG1)	32	R/W	0000_0000h	43.9.2/1385
A8	eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM5CR_ENG1)	32	R/W	0000_0000h	43.9.1/1383
AC	eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM5MR_ENG1)	32	R/W	0000_0000h	43.9.2/1385
C0	eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM0CR_ENG2)	32	R/W	0000_0000h	43.9.3/1386
C4	eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM0MR_ENG2)	32	R/W	0000_0000h	43.9.4/1387
C8	eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM1CR_ENG2)	32	R/W	0000_0000h	43.9.3/1386
CC	eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM1MR_ENG2)	32	R/W	0000_0000h	43.9.4/1387
D0	eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM2CR_ENG2)	32	R/W	0000_0000h	43.9.3/1386
D4	eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM2MR_ENG2)	32	R/W	0000_0000h	43.9.4/1387
D8	eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM3CR_ENG2)	32	R/W	0000_0000h	43.9.3/1386
DC	eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM3MR_ENG2)	32	R/W	0000_0000h	43.9.4/1387

Table continues on the next page...

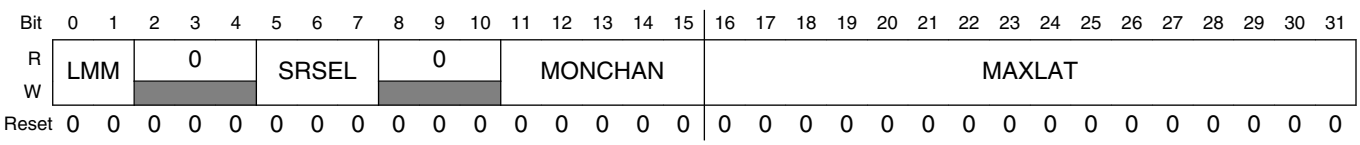
eTPU_LMR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E0	eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM4CR_ENG2)	32	R/W	0000_0000h	43.9.3/1386
E4	eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM4MR_ENG2)	32	R/W	0000_0000h	43.9.4/1387
E8	eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM5CR_ENG2)	32	R/W	0000_0000h	43.9.3/1386
EC	eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM5MR_ENG2)	32	R/W	0000_0000h	43.9.4/1387

43.9.1 eTPU Latency Monitor Configuration Register for Engine 1 (eTPU_LMR_LM*n*CR_ENG1)

This register configures the operation of individual latency monitors.

Address: 0h base + 80h offset + (8d × i), where i=0d to 5d



eTPU_LMR_LM*n*CR_ENG1 field descriptions

Field	Description
0–1 LMM	Latency Monitor Mode. This field selects the operation mode of the latency monitor. See the Latency Monitor section for detailed mode descriptions. **NOTE:** LMM must be in disabled mode to be changed to channel or priority mode. Channel prioritiy fields CPR in ETPUCnCR registers must not be changed while LMM of any Latency Monitor is in priority mode. 00 Disabled mode 01 Channel mode 10 Priority mode 11 Reserved
2–4 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SRSEL	Service Request Selection

Table continues on the next page...

eTPU_LMR_LM*n*CR_ENG1 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>This field selects which service requests are monitored. The SRSEL value has different meanings in channel and priority modes, according to the following table.
Table 43-42. SRSEL - Service Request Selection
<table>
<tr><th>SRSEL</th><th>Channel Mode</th><th>Priority Mode</th></tr>
<tr><td>001</td><td>Link SRs monitored only</td><td>Low priority monitored</td></tr>
<tr><td>010</td><td>Channel SRs monitored only</td><td>Medium priority monitored</td></tr>
<tr><td>100</td><td>HSRs monitored only</td><td>High priority monitored</td></tr>
<tr><td>000</td><td>No SRs monitored</td><td rowspan="5">Reserved</td></tr>
<tr><td>011</td><td>Channel and Link SRs monitored</td></tr>
<tr><td>101</td><td>HSRs and Link SRs monitored</td></tr>
<tr><td>110</td><td>HSRs and Channel SRs monitored</td></tr>
<tr><td>111</td><td>All SRs monitored</td></tr>
</table></td></tr>
<tr><td>8–10
Reserved</td><td>Reserved.

This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>11–15
MONCHAN</td><td>Monitored Channel.

When the latency monitor is configured for channel mode, this field selects which channel has the latency monitored. MONCHAN has no meaning in priority mode.

NOTE: MONCHAN must only be changed when the Latency Monitor is in disabled mode.</td></tr>
<tr><td>16–31
MAXLAT</td><td>Maximum Latency.

This field specifies the maximum latency monitored, in microcycles, before a latency timeout. If MAXLAT=0, the latency keeps on being counted and measured, but the comparison with MAXLAT asserts neither ETPUMCR bit LTO1/2 nor issues a Global Exception. See the Latency Monitor section for the detailed latency timeout description.

NOTE: MAXLAT must only be changed when the Latency Monitor is in disabled mode.</td></tr>
</table>

43.9.2 eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM*n*MR_ENG1)

This register holds the latency measured value.

Address: 0h base + 84h offset + (8d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LATENCY															LTC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_LMR_LM*n*MR_ENG1 field descriptions

Field	Description
0–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16–30 LATENCY	Latency measurement. This read-only field holds the value of the greatest latency measured, in microcycles, since reset or the latest clear. When the value is greater than the maximum specified in the ETPULMnCR field MAXLAT and MAXLAT value is not zero, a Global Exception is issued and the ETPUMCR bit LTO1 (engine 1) or LTO2 (engine 2) is set. See the Latency Monitor section for the detailed latency measurement description. **NOTE:** When read, the LATENCY field is concatenated with the LTC bit (LSB) to form a 16-bit value. **NOTE:** Any latency measured above 0xFFFF is captured as 0xFFFF into the LATENCY+LTC field.
31 LTC	Latency measurement clear Writing a 1 to the LTC bit clears the LATENCY field. **NOTE:** When read, the LATENCY field is concatenated with the LTC bit (LSB) to form a 16-bit value. **NOTE:** The LATENCY value is not automatically cleared when the Latency Monitor is disabled. 0 Keep the LATENCY field value. 1 Clear the LATENCY field value.

43.9.3 eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM*n*CR_ENG2)

This register configures the operation of individual latency monitors.

Address: 0h base + C0h offset + (8d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LMM		0			SRSEL			0			MONCHAN					MAXLAT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_LMR_LM*n*CR_ENG2 field descriptions

Field	Description
0–1 LMM	Latency Monitor Mode. This field selects the operation mode of the latency monitor. See the Latency Monitor section for detailed mode descriptions. **NOTE:** LMM must be in disabled mode to be changed to channel or priority mode. Channel prioritiy fields CPR in ETPUCnCR registers must not be changed while LMM of any Latency Monitor is in priority mode. 00 Disabled mode 01 Channel mode 10 Priority mode 11 Reserved
2–4 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
5–7 SRSEL	Service Request Selection This field selects which service requests are monitored. The SRSEL value has different meanings in channel and priority modes, according to the following table.

Table 43-63. SRSEL - Service Request Selection

SRSEL	Channel Mode	Channel Mode
001	Link SRs monitored only	Low priority monitored
010	Channel SRs monitored only	Medium priority monitored
100	HSRs monitored only	High priority monitored
000	No SRs monitored	Reserved
011	Channel and Link SRs monitored	
101	HSRs and Link SRs monitored	
110	HSRs and Channel SRs monitored	
111	All SRs monitored	

Table continues on the next page...

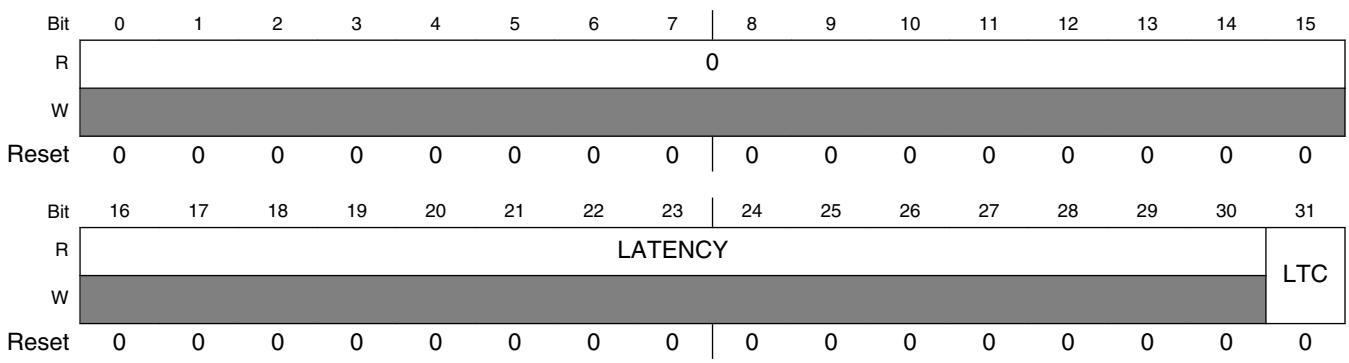
eTPU_LMR_LM*n*CR_ENG2 field descriptions (continued)

Field	Description
8–10 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
11–15 MONCHAN	Monitored Channel. When the latency monitor is configured for channel mode, this field selects which channel has the latency monitored. MONCHAN has no meaning in priority mode. **NOTE:** MONCHAN must only be changed when the Latency Monitor is in disabled mode.
16–31 MAXLAT	Maximum Latency. This field specifies the maximum latency monitored, in microcycles, before a latency timeout. If MAXLAT=0, the latency keeps on being counted and measured, but the comparison with MAXLAT asserts neither ETPUMCR bit LTO1/2 nor issues a Global Exception. See the Latency Monitor section for the detailed latency timeout description. **NOTE:** MAXLAT must only be changed when the Latency Monitor is in disabled mode.

43.9.4 eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM*n*MR_ENG2)

This register holds the latency measured value.

Address: 0h base + C4h offset + (8d × i), where i=0d to 5d



eTPU_LMR_LM*n*MR_ENG2 field descriptions

Field	Description
0–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16–30 LATENCY	Latency measurement.

Table continues on the next page...

eTPU_LMR_LM*n*MR_ENG2 field descriptions (continued)

Field	Description
	This read-only field holds the value of the greatest latency measured, in microcycles, since reset or the latest clear. When the value is greater than the maximum specified in the ETPULMnCR field MAXLAT and MAXLAT value is not zero, a Global Exception is issued and the ETPUMCR bit LTO1 (engine 1) or LTO2 (engine 2) is set. See the Latency Monitor section for the detailed latency measurement description. **NOTE:** When read, the LATENCY field is concatenated with the LTC bit (LSB) to form a 16-bit value. **NOTE:** Any latency measured above 0xFFFF is captured as 0xFFFF into the LATENCY+LTC field.
31 LTC	Latency measurement clear. Writing a 1 to the LTC bit clears the LATENCY field. **NOTE:** When read, the LATENCY field is concatenated with the LTC bit (LSB) to form a 16-bit value. **NOTE:** The LATENCY value is not automatically cleared when the Latency Monitor is disabled. 0 Keep the LATENCY field value. 1 Clear the LATENCY field value.

43.10 Memory Error Support memory map and registers

eTPU_MESR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
100	eTPU Memory Error Control Register (eTPU_MESR_MECR)	32	R/W	0000_0000h	43.10.1/ 1389
104	eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR)	32	R/W	0000_0000h	43.10.2/ 1392
108	eTPU Data Error Injection Data Pattern Register (eTPU_MESR_DEIDPR)	32	R/W	0000_0000h	43.10.3/ 1393
10C	eTPU Data Error Injection Parity Pattern Register (eTPU_MESR_DEIPPR)	32	R/W	0000_0000h	43.10.4/ 1393
110	eTPU Data Error Report Address Register (eTPU_MESR_DERAR)	32	R	0000_0000h	43.10.5/ 1395
114	eTPU Data Error Report Data Register (eTPU_MESR_DERDR)	32	R	0000_0000h	43.10.6/ 1396
118	eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR)	32	R	0000_0000h	43.10.7/ 1397
120	eTPU Memory Error Status Register (eTPU_MESR_MESR)	32	R/W	0000_0000h	43.10.8/ 1399
124	eTPU Code Error Injection Address Register (eTPU_MESR_CEIAR)	32	R/W	0000_0000h	43.10.9/ 1402
128	eTPU Code Error Injection Data Pattern Register (eTPU_MESR_CEIDPR)	32	R/W	0000_0000h	43.10.10/ 1403
12C	eTPU Code Error Injection Parity Pattern Register (eTPU_MESR_CEIPPR)	32	R/W	0000_0000h	43.10.11/ 1403

Table continues on the next page...

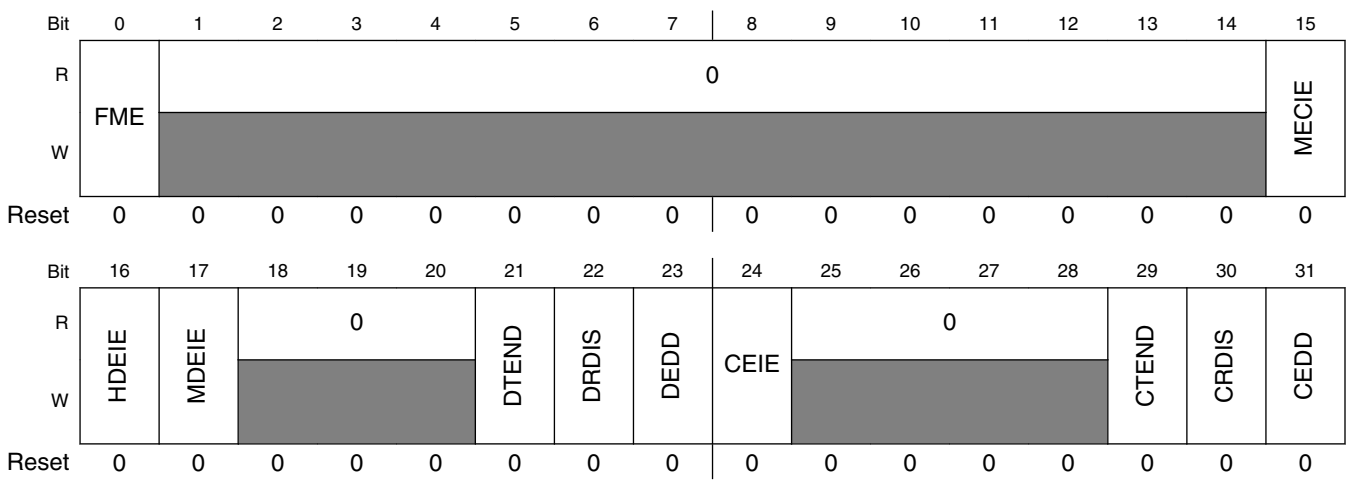
eTPU_MESR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
130	eTPU Code Error Report Address Register (eTPU_MESR_CERAR)	32	R	0000_0000h	43.10.12/ 1404
134	eTPU Code Error Report Data Register (eTPU_MESR_CERDR)	32	R	0000_0000h	43.10.13/ 1405
138	eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR)	32	R	0000_0000h	43.10.14/ 1406
13C	eTPU Code Error Fix Register (eTPU_MESR_CEFR)	32	R/W	0000_0000h	43.10.15/ 1407

43.10.1 eTPU Memory Error Control Register (eTPU_MESR_MECR)

This register gathers control bits of both SCM and SDM error detection and correction support.

Address: 0h base + 100h offset = 100h



eTPU_MESR_MECR field descriptions

Field	Description
0 FME	Field Modification Enable. Negation of this bit disables the modification of all fields in the register, except MECIE, DRDIS, CRDIS and FME itself: Write accesses do not cause a bus error, regardless of FME state. 0 Fields cannot be modified during a write access, except MECIE, DRDIS, CRDIS and FME. 1 All fields can be modified during a write access.
1–14 Reserved	Reserved.

Table continues on the next page...

eTPU_MESR_MECR field descriptions (continued)

Field	Description
	This field is reserved. This read-only field is reserved and always has the value 0.
15 MECIE	Memory Error Correction Interrupt Enable. This bit enables the Memory Error Correction Interrupt (see Interrupt Types and Sources). 0 Memory Error Correction Interrupt disabled. 1 Memory Error Correction Interrupt enabled.
16 HDEIE	Host Data Error Injection Enable This bit enables error injection into the SDM host read path, including CDC transfers. The eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR) register specifies the address where the error is injected, while registers eTPU Data Error Injection Data Pattern Register (eTPU_MESR_DEIDPR) and eTPU Data Error Injection Parity Pattern Register (eTPU_MESR_DEIPPR) define the bit flips on data and parity, respectively. **NOTE:** Data error injection is also performed when data error detection is disabled (DEDD=1). 0 SDM error injection disabled for host reads. 1 SDM error injection enabled for host reads.
17 MDEIE	Microengine(s) Data Error Injection Enable. This bit enables error injection into the SDM microengine read path. The eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR) register specifies the address where the error is injected, while registers eTPU Data Error Injection Data Pattern Register (eTPU_MESR_DEIDPR) and eTPU Data Error Injection Parity Pattern Register (eTPU_MESR_DEIPPR) define the bit flips on data and parity, respectively. **NOTE:** Data error injection is also performed when data error detection is disabled (DEDD=1). 0 SDM error injection disabled for microengine reads. 1 SDM error injection enabled for microengine reads.
18–20 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
21 DTEND	Thread End on Data Non-correctable Error. This bit controls the behavior of the microengine when a non-correctable data error occurs. For more details, see “Memory Error Support” Memory Error Support. **NOTE:** DTEND must only be changed when the engine is stopped or halted, or when all channels are disabled. 0 Thread continues on non-correctable data read access by the microengine. 1 Thread ends on non-correctable data read access by the microengine.
22 DRDIS	Data Error Report Disable. This bit disables the sampling of SDM error address, data and syndrome into the registers eTPU Memory Error Support Registers, and eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR). It does not, however, disable the setting of eTPU Memory Error Status Register (eTPU_MESR_MESR) bits DCERR and DNCERR.

Table continues on the next page...

eTPU_MESR_MECR field descriptions (continued)

Field	Description
	NOTE: To guarantee coherent reading of registers ETPUDERAR, ETPUDERDR, ETPUDERSR, DRDIS must be asserted beforehand if SDM concurrent accesses can happen, by host or microengine. After the registers are read, DRDIS may be negated again to allow new samplings. 0 SDM error report sampling enabled. 1 SDM error report sampling disabled.
23 DEDD	Data Error Detection Disable. This bit disables SDM error detection. The disabled state turns ineffective any action due to error detection, including consequently error correction. 0 SDM error detection enabled. 1 SDM error detection disabled.
24 CEIE	Code Error Injection Enable. This bit enables error injection into the SCM read path, either on microengine accesses (see eTPU Module Configuration Register (eTPU_SCR_MCR) bit VIS=0) or host accesses (VIS = 1) The eTPU Code Error Injection Address Register (eTPU_MESR_CEIAR) register specifies the address where the error is injected, while registers eTPU Code Error Injection Data Pattern Register (eTPU_MESR_CEIDPR) and eTPU Code Error Injection Parity Pattern Register (eTPU_MESR_CEIPPR) define the bit flips on data and parity, respectively. **NOTE:** Code error injection is also performed when code error detection is disabled (CEDD=1). 0 SCM error injection disabled. 1 SCM error injection enabled.
25–28 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
29 CTEND	Thread End on Code Non-correctable Error. This bit controls the behavior of the microengine when a non-correctable code error occurs. For more details, see Memory Error Support. **NOTE:** CTEND must only be changed when the engine is stopped or halted, or when all channels are disabled. 0 Thread continues on instruction fetch non-correctable error. 1 Thread ends on instruction fetch non-correctable error.
30 CRDIS	Code Error Report Disable. This bit disables the sampling of SCM error address, data and syndrome into the registers eTPU Code Error Report Address Register (eTPU_MESR_CERAR), eTPU Code Error Report Data Register (eTPU_MESR_CERDR), and eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR). It does not, however, disable the setting of ETPUMESR bits CCERR and CNCERR (see eTPU Memory Error Status Register (eTPU_MESR_MESR)). **NOTE:** To guarantee coherent reading of registers ETPUCERAR, ETPUCERDR and ETPUCERSR, CRDIS must be asserted beforehand if SCM concurrent accesses can happen, by host or microengine. After the registers are read, CRDIS may be negated again to allow new samplings.

Table continues on the next page...

eTPU_MESR_MECR field descriptions (continued)

Field	Description
	0 SCM error report sampling enabled. 1 SCM error report sampling disabled.
31 CEDD	Code Error Detection Disable. This bit disables SCM error detection. The disabled state turns ineffective any action due to error detection, including consequently error correction. 0 SCM error detection enabled. 1 SCM error detection disabled.

43.10.2 eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR)

This register holds the address where error is to be injected into the SDM read data path. The two least significant bits are 0 (read-only), so that the address is always word-aligned.

NOTE

This register must only be modified when ETPUMECR bits HDEIE and MDEIE are both negated.

Address: 0h base + 104h offset = 104h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																INJ_ADDR														0	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_DEIAR field descriptions

Field	Description
0–17 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
18–29 INJ_ADDR	Injection Address. INJ_ADDR holds the (word) address where an error is injected into the SDM read data path, as specified by registers ETPUDEIDPR eTPU Data Error Injection Data Pattern Register (eTPU_MESR_DEIDPR) and ETPUDEIPPR eTPU Data Error Injection Parity Pattern Register (eTPU_MESR_DEIPPR). For more information on error injection, see Error Injection.
30–31 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

43.10.3 eTPU Data Error Injection Data Pattern Register (eTPU_MESR_DEIDPR)

This register is used to specify the bits in which error is injected into the SDM read data path.

NOTE

This register must only be modified when ETPUMECR bits HDEIE and MDEIE are both negated.

Address: 0h base + 108h offset = 108h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DFLIP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_DEIDPR field descriptions

Field	Description
0–31 DFLIP	Data Flip bits. Each bit corresponds to a bit in the actual data read from the SDM address defined in the eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR) register, specifying whether it should be inverted or not. The inversion only occurs if the SDM error injection is enabled by the eTPU Memory Error Control Register (eTPU_MESR_MECR) register bits HDEIE and/or MDEIE. For more information on error injection, see Error Injection. 0 Read data bit is not inverted. 1 Read data bit is inverted.

43.10.4 eTPU Data Error Injection Parity Pattern Register (eTPU_MESR_DEIPPR)

This register is used to specify the bits in which error is injected into the SDM read parity path.

NOTE

This register must only be modified when ETPUMECR bits HDEIE and MDEIE are both negated.

Address: 0h base + 10Ch offset = 10Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			PFLIP_19_15					0			PFLIP_14_10					0			PFLIP_9_5					0			PFLIP_4_0			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_DEIPPR field descriptions

Field	Description
0–2 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
3–7 PFLIP_19_15	Parity Flip bits. Each bit corresponds to a bit in the actual parity read from the SDM address defined in the eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR) register, specifying whether it should be inverted or not. The inversion only occurs if the SDM error injection is enabled by the eTPU Memory Error Control Register (eTPU_MESR_MECR) register bits HDEIE and/or MDEIE. Bits PFLIP[19:15] corresponds to the parity bits for the most significant byte, PFLIP[14:10] to the next most significant byte and so on until PFLIP[4:0] for the least significant byte. For more information on error injection, see Error Injection. 0 Parity data bit is not inverted. 1 Parity data bit is inverted.
8–10 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
11–15 PFLIP_14_10	Parity Flip bits. Each bit corresponds to a bit in the actual parity read from the SDM address defined in the eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR) register, specifying whether it should be inverted or not. The inversion only occurs if the SDM error injection is enabled by the eTPU Memory Error Control Register (eTPU_MESR_MECR) register bits HDEIE and/or MDEIE. Bits PFLIP[19:15] corresponds to the parity bits for the most significant byte, PFLIP[14:10] to the next most significant byte and so on until PFLIP[4:0] for the least significant byte. For more information on error injection, see Error Injection. 0 Parity data bit is not inverted. 1 Parity data bit is inverted.
16–18 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
19–23 PFLIP_9_5	Parity Flip bits. Each bit corresponds to a bit in the actual parity read from the SDM address defined in the eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR) register, specifying whether it should be inverted or not. The inversion only occurs if the SDM error injection is enabled by the eTPU Memory Error Control Register (eTPU_MESR_MECR) register bits HDEIE and/or MDEIE. Bits PFLIP[19:15] corresponds to the parity bits for the most significant byte, PFLIP[14:10] to the next most significant byte and so on until PFLIP[4:0] for the least significant byte. For more information on error injection, see Error Injection. 0 Parity data bit is not inverted. 1 Parity data bit is inverted.
24–26 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–31 PFLIP_4_0	Parity Flip bits.

Table continues on the next page...

eTPU_MESR_DEIPPR field descriptions (continued)

Field	Description
	Each bit corresponds to a bit in the actual parity read from the SDM address defined in the eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR) register, specifying whether it should be inverted or not. The inversion only occurs if the SDM error injection is enabled by the eTPU Memory Error Control Register (eTPU_MESR_MECR) register bits HDEIE and/or MDEIE. Bits PFLIP[19:15] corresponds to the parity bits for the most significant byte, PFLIP[14:10] to the next most significant byte and so on until PFLIP[4:0] for the least significant byte. For more information on error injection, see Error Injection. 0 Parity data bit is not inverted. 1 Parity data bit is inverted.

43.10.5 eTPU Data Error Report Address Register (eTPU_MESR_DERAR)

This register holds the address where the last non-correctable error was detected from the SDM It also captures the address of the last correctable error detected, as long as the eTPU Memory Error Status Register (eTPU_MESR_MESR) bit DNCERR is 0. The two least significant bits are 0 (read-only), so that the address is always word-aligned.

Address: 0h base + 110h offset = 110h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR_ACC		0	ERR_CHANNEL					0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		ERR_ADDR												0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_DERAR field descriptions

Field	Description
0–1 ERR_ACC	Error Access Source Indication. This two-bit field indicates the source of the access when the error occurred. 00 Host - direct access 01 Engine 1 10 Engine 2 11 CDC
2 Reserved	Reserved.

Table continues on the next page...

eTPU_MESR_DERAR field descriptions (continued)

Field	Description
	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 ERR_CHANNEL	Error Channel Access Indication. This field holds the number of the channel being serviced (not necessarily CHAN) when the error occurred. Its content is meaningful only when the field ERR_ACC indicates Engine 1 or Engine 2.
8–17 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
18–29 ERR_ADDR	Error Address. ERR_ADDR holds the (word) address where the last captured SDM read error was detected. The last captured error is a non-correctable one if eTPU Memory Error Status Register (eTPU_MESR_MESR) register bit DNCERR=1, or a correctable one if DNCERR=0 and DCERR=1. The field is meaningless if both DNCERR and DCERR are 0. For more information on error detection, see Memory Error Support.
30–31 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

43.10.6 eTPU Data Error Report Data Register (eTPU_MESR_DERDR)

This register holds the data read with the last non-correctable error detected from the SDM. It also captures the data of the last correctable error read detection, as long as the eTPU Memory Error Status Register (eTPU_MESR_MESR) bit DNCERR is 0. In both cases, the data is captured after the error correction logic. The data is meaningless if both DNCERR and DCERR are 0. For more information on error detection, see Memory Error Support.

NOTE

Data is sampled before the error correction logic.

Address: 0h base + 114h offset = 114h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_DERDR field descriptions

Field	Description
0–31 DATA	DATA.

eTPU_MESR_DERDR field descriptions (continued)

Field	Description
	NOTE: Individual data bytes are not meaningful when their respective BE bits in the eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR) register are not asserted.

43.10.7 eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR)

This register holds the syndrome read with the last non-correctable error detected from the SDM. It also captures the syndrome of the last correctable error read detection, as long as the eTPU Memory Error Control Register (eTPU_MESR_MECR) bit DNCERR is 0.

Address: 0h base + 118h offset = 118h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BE3	0		SYND3					BE2	0		SYND2				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BE1	0		SYND1					BE0	0		SYND0				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_DERSR field descriptions

Field	Description
0 BE3	Byte Enable 3. Each BE bit corresponds to a report byte in a 32-bit aligned data word. BE3 corresponds to the most significant one. BE qualifies the error report for its respective byte. 0 Byte was not selected for read, therefore its respective error report fields are not meaningful. 1 Byte was selected for read, and the error report for the respective byte is meaningful.
1–2 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
3–7 SYND3	Error Syndrome 3. SYND3 holds the syndrome of an error detected in the SDM read data path, as defined in Error Correction Code (ECC) and Syndrome Definition. The field is meaningless if both eTPU Memory Error Status Register (eTPU_MESR_MESR) bits DNCERR and DCERR are 0. For more information on error detection, see Memory Error Support.

Table continues on the next page...

eTPU_MESR_DERSR field descriptions (continued)

Field	Description
	NOTE: Syndrome fields which have their respective BE bits negated are not meaningful.
8 BE2	Byte Enable 2. Each BE bit corresponds to a report byte in a 32-bit aligned data word. BE3 corresponds to the most significant one. BE qualifies the error report for its respective byte. 0 Byte was not selected for read, therefore its respective error report fields are not meaningful. 1 Byte was selected for read, and the error report for the respective byte is meaningful.
9–10 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
11–15 SYND2	Error Syndrome 2. SYND2 holds the syndrome of an error detected in the SDM read data path, as defined in the “Error Correction Code (ECC) and Syndrome Definition”section. The field is meaningless if both the eTPU Memory Error Status Register (eTPU_MESR_MESR) bit DNCERR and DCERR are 0. For more information on error detection, see Memory Error Support. **NOTE:** Syndrome fields which have their respective BE bits negated are not meaningful.
16 BE1	Byte Enable 1. Each BE bit corresponds to a report byte in a 32-bit aligned data word. BE3 corresponds to the most significant one. BE qualifies the error report for its respective byte. 0 Byte was not selected for read, therefore its respective error report fields are not meaningful. 1 Byte was selected for read, and the error report for the respective byte is meaningful.
17–18 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
19–23 SYND1	Error Syndrome 1. SYND1 holds the syndrome of an error detected in the SDM read data path, as defined in the “Error Correction Code (ECC) and Syndrome Definition”section. The field is meaningless if both the eTPU Memory Error Status Register (eTPU_MESR_MESR) bits DNCERR and DCERR are 0. For more information on error detection, see Memory Error Support. **NOTE:** Syndrome fields which have their respective BE bits negated are not meaningful.
24 BE0	Byte Enable 0. Each BE bit corresponds to a report byte in a 32-bit aligned data word. BE3 corresponds to the most significant one. BE qualifies the error report for its respective byte. 0 Byte was not selected for read, therefore its respective error report fields are not meaningful. 1 Byte was selected for read, and the error report for the respective byte is meaningful.
25–26 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–31 SYND0	Error Syndrome 0.

Table continues on the next page...

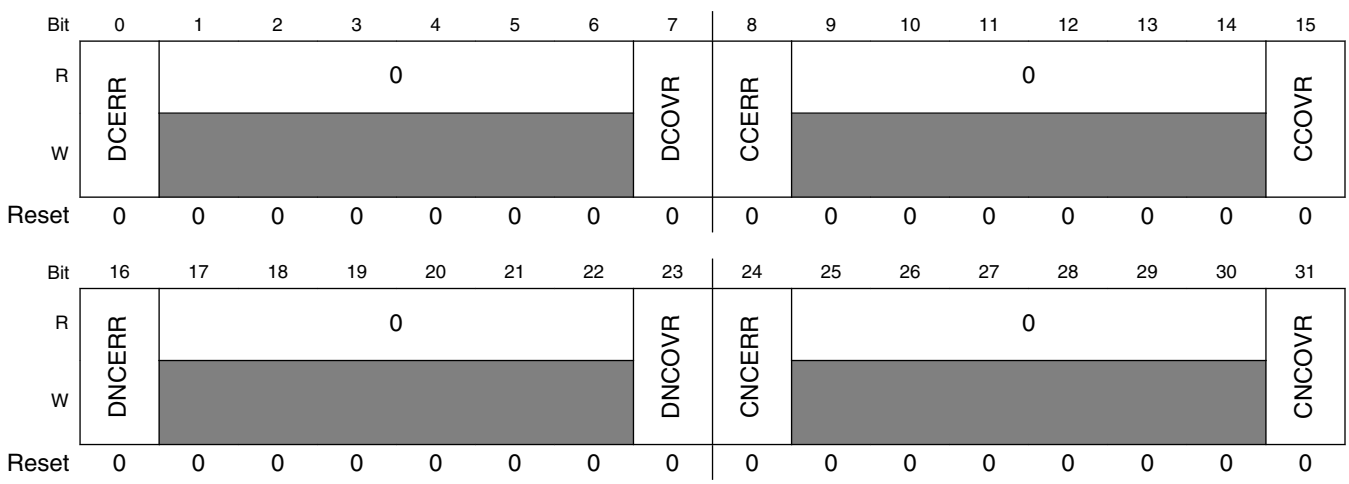
eTPU_MESR_DERSR field descriptions (continued)

Field	Description
	SYND0 holds the syndrome of an error detected in the SDM read data path, as defined in the “Error Correction Code (ECC) and Syndrome Definition”section. The field is meaningless if both the eTPU Memory Error Status Register (eTPU_MESR_MESR) bits DNCERR and DCERR are 0. For more information on error detection, see Memory Error Support. **NOTE:** Syndrome fields which have their respective BE bits negated are not meaningful.

43.10.8 eTPU Memory Error Status Register (eTPU_MESR_MESR)

This register gathers status bits of both SCM and SDM error detection and correction support.

Address: 0h base + 120h offset = 120h



eTPU_MESR_MESR field descriptions

Field	Description
0 DCERR	Data Correctable Error flag. When read: 0.....No SDM correctable error occurred. 1.....An SDM correctable error occurred. Writing 1 to this bit clears the DCERR status flag. DCERR asserts even if DNCERR is asserted beforehand, but in this case the address and data of the erroneous access is not captured (see the eTPU Memory Error Support registers and following sections). DCERR is cleared by writing a 1 to it. Assertion of this bit can also assert the Memory Error Correction Interrupt (see “Interrupt Types and Sources” Interrupt Types and Sources). **NOTE:** A correctable error is not considered for flagging if accompanied of a non-correctable one. Therefore, if both correctable and non-correctable data errors occur on the same access (in distinct bytes), the DNCERR flag asserts, but not the DCERR flag.

Table continues on the next page...

eTPU_MESR_MESR field descriptions (continued)

Field	Description
	0 Do not clear DCERR flag. 1 Clears DCERR flag
1–6 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
7 DCOVR	Data Correctable Overrun flag. Writing 1 to this bit clears the DCOVR status flag. 0 Data correctable error occurred while DCOVR = 1. 1 No data correctable error occurred while DCOVR = 1. DCOVR asserts when an SDM correctable error occurs and the DCERR flag is already asserted. Assertion of this bit causes no interrupt. **NOTE:** A correctable error is not considered for flagging if accompanied by a non-correctable one. Therefore, if both correctable and non-correctable data errors occur on the same access (in distinct bytes), the DCOVR does not assert, even if DCERR is asserted. The DNCOVR flag, on the other hand, asserts if DNCERR is asserted.
8 CCERR	Code Error flag. >When read: 0.....No SCM correctable error occurred. 1.....An SCM correctable error occurred. When written: 0.....Do not clear CCERR flag. 1.....Clears CCERR flag. The CCERR flag asserts even if CNCERR is asserted beforehand, but in this case the address and data of the erroneous access is not captured (see the eTPU Code Error Report Address Register (eTPU_MESR_CERAR) register and the following sections). CCERR is cleared by writing a 1 at the same position. Assertion of this bit can also assert the Memory Error Correction Interrupt (see Interrupt Types and Sources).
9–14 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
15 CCOVR	Code Correctable Overrun flag This bit asserts when an SCM correctable error occurs and the CCERR flag is already asserted. Assertion of this bit causes no interrupt. When read: 0.....No SCM correctable error overrun occurred. 1.....An SCM correctable error overrun occurred. Writing 1 to this bit clears the CCOVR status flag: 0 Do not clear CCOVR flag. 1 Clear CCOVR flag.

Table continues on the next page...

eTPU_MESR_MESR field descriptions (continued)

Field	Description
16 DNCERR	Data Non-Correctable Error flag DNCERR flags a non-correctable error. When read: 0.....No SDM non-correctable error occurred. 1.....An SDM non-correctable error occurred; registers eTPU Global Channel Registers are not updated by correctable error detections. DNCERR must be cleared (by writing a 1 to it) to allow correctable error updates to registers ETPUDERAR and ETPUDERDR. 0 Do not clear DNCERR flag. 1 Clear DNCERR flag.
17–22 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
23 DNCOVR	Data Non-Correctable Overrun flag. DNCOVR asserts when an SDM non-correctable error occurs and the DNCERR flag is already asserted. Assertion of this bit causes no interrupt. When read: 0.....No SDM non-correctable error overrun occurred. 1.....An SDM non-correctable error overrun occurred. DNCOVR is cleared by writing a 1 to it. 0 Do not clear DNCOVR flag. 1 Clear DNCOVR flag.
24 CNCERR	Code Non-Correctable Error flag CNCERR flags a non-correctable error: When read: 0.....An SCM non-correctable error occurred; the eTPU Global Channel Registers are not updated by correctable error detections. 1....an SCM non-correctable error occurred; registers ETPUCERAR and ETPUCERD are not updated by correctable error detections. CNCERR must be cleared to allow correctable error updates to registers ETPUCERAR and ETPUCERDR. CNCERR is cleared by writing a 1 to it. 0 Do not clear CNCERR flag. 1 Clear CNCERR flag.
25–30 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
31 CNCOVR	Code Non-Correctable Overrun flag.

Table continues on the next page...

eTPU_MESR_MESR field descriptions (continued)

Field	Description
	CNCOVR asserts when an SCM non-correctable error occurs and the CNCERR flag is already asserted. Assertion of this bit causes no interrupt. When read: 0.....an SCM non-correctable error overrun occurred. 1.....An SCM non-correctable error overrun occurred. When written: 0 Do not clear CNCOVR flag. 1 Clear CNCOVR flag.

43.10.9 eTPU Code Error Injection Address Register (eTPU_MESR_CEIAR)

This register holds the address where error is to be injected into the SCM read data path. The two least significant bits are 0 (read-only), so that the address is always word-aligned.

NOTE

This register must only be modified when eTPU Memory Error Control Register (eTPU_MESR_MECR) bit CEIE is negated.

Address: 0h base + 124h offset = 124h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																INJ_ADDR														0	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_CEIAR field descriptions

Field	Description
0–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16–29 INJ_ADDR	Injection Address. INJ_ADDR holds the (word) address where an error is injected in the SCM read data path, as specified by registers eTPU Code Error Injection Data Pattern Register (eTPU_MESR_CEIDPR) and eTPU Code Error Injection Parity Pattern Register (eTPU_MESR_CEIPPR). For more information on error injection, see Error Injection.
30–31 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

43.10.10 eTPU Code Error Injection Data Pattern Register (eTPU_MESR_CEIDPR)

This register is used to specify the bits in which error is injected into the SCM read data path.

NOTE

This register must only be modified when eTPU Memory Error Control Register (eTPU_MESR_MECR) bit CEIE is negated.

Address: 0h base + 128h offset = 128h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DFLIP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_CEIDPR field descriptions

Field	Description
0–31 DFLIP	Data Flip bits. Each bit corresponds to a bit in the actual data read from the SCM address defined in the eTPU Code Error Injection Address Register (eTPU_MESR_CEIAR) register, specifying whether it should be inverted or not. The inversion only occurs if the SCM error injection is enabled by eTPU Memory Error Control Register (eTPU_MESR_MECR) register bit CEIE. For more information on error injection, see Error Injection. 0 Read data bit is not inverted. 1 Read data bit is inverted.

43.10.11 eTPU Code Error Injection Parity Pattern Register (eTPU_MESR_CEIPPR)

This register is used to specify the bits in which error is injected into the SCM read parity path.

NOTE

This register must only be modified when eTPU Memory Error Control Register (eTPU_MESR_MECR) bit CEIE is negated.

Address: 0h base + 12Ch offset = 12Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																									PFLIP						
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_CEIPPR field descriptions

Field	Description
0–24 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
25–31 PFLIP	Parity Flip bits. Each bit corresponds to a bit in the actual parity/code read from the SCM address defined in the eTPU Code Error Injection Address Register (eTPU_MESR_CEIAR) register, specifying whether it should be inverted or not. The inversion only occurs if the SCM error injection is enabled by eTPU Memory Error Control Register (eTPU_MESR_MECR) bit CEIE. For more information on error injection, see Error Injection. 0 Parity data bit is not inverted. 1 Parity data bit is inverted.

43.10.12 eTPU Code Error Report Address Register (eTPU_MESR_CERAR)

This register holds the address where the last non-correctable error was detected from the SCM. It also captures the address of the last correctable error detected, as long as the eTPU Memory Error Control Register (eTPU_MESR_MECR) bit CNCERR is 0. The two least significant bits are 0 (read-only), so that the address is always word-aligned.

Address: 0h base + 130h offset = 130h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR_ACC		0	ERR_CHANNEL					0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR_ADDR														0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_CERAR field descriptions

Field	Description
0–1 ERR_ACC	Error Access Source Indication. This two-bit field indicates the source of the access when the error occurred. 00 Host 01 Engine 1

Table continues on the next page...

eTPU_MESR_CERAR field descriptions (continued)

Field	Description
	10 Engine 2 11 MISC
2 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
3–7 ERR_CHANNEL	Error Channel Access Indication. This field holds the number of the channel being serviced (not necessarily CHAN) when the error occurred. Its content is meaningful only when the field ERR_ACC indicates Engine 1 or Engine 2.
8–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16–29 ERR_ADDR	Error Address. ERR_ADDR holds the (word) address where the last captured SCM read error was detected. The last captured error is a non-correctable one if eTPU Memory Error Control Register (eTPU_MESR_MECR) register bit CNCERR = 1, or a correctable one if CNCERR = 0 and CCERR = 1. The field is meaningless if both CNCERR and CCERR are 0. For more information on error detection, see Memory Error Support.
30–31 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

43.10.13 eTPU Code Error Report Data Register (eTPU_MESR_CERDR)

This register holds the data read with the last non-correctable error detected from the SCM. It also captures the data of the last correctable error read detection, as long as eTPU Memory Error Control Register (eTPU_MESR_MECR) bit CNCERR is 0. In both cases, the data is captured after the error correction logic. The data may be meaningless if both CNCERR and CCERR are 0. For more information on error detection, see Memory Error Support.

NOTE

CODE is sampled before the error correction logic.

Address: 0h base + 134h offset = 134h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CODE																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_CERDR field descriptions

Field	Description
0–31 CODE	CODE[31-0].

43.10.14 eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR)

This register holds the syndrome and global parity read with the last non-correctable error detected from the SCM. It also captures the syndrome of the last correctable error read detection, as long as eTPU Memory Error Control Register (eTPU_MESR_MECR) bit CNCERR is 0

Address: 0h base + 138h offset = 138h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																									SYND						
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_CERSR field descriptions

Field	Description
0–24 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
25–31 SYND	Error Syndrome. SYND holds the syndrome of an error detected in the SCM read data path, as defined in Error Correction Code (ECC) and Syndrome Definition. The field is meaningless if both eTPU Memory Error Control Register (eTPU_MESR_MECR) bits CNCERR and CCERR are 0. For more information on error detection, see Memory Error Support.

43.10.15 eTPU Code Error Fix Register (eTPU_MESR_CEFR)

This register is used to fix single-bit error in a specified SCM position. The fix is done automatically, concurrently with code execution, by the MISC logic (see Error Fixing).

Address: 0h base + 13Ch offset = 13Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CFIXM		0													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CFIX_ADDR														0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_MESR_CEFR field descriptions

Field	Description
0–1 CFIXM	Code Fix Mode. This field specifies the SCM MISC fix logic operation mode as shown in the table below. **NOTE:** The CFIXM field can only be changed to/from disabled mode. **Table 43-99. CFIXM - Code Fix Mode** Value / Mode / Description: 00 / Disabled / SCM error fix disabled. 01 / Reserved / Reserved value, must not be used. 10 / Reserved / Reserved value, must not be used. 11 / Automatic / The MISC logic rewrites the code word value read (already fixed by the error correction mechanism) into the same address whenever it finds a correctable error.
2–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16–29 CFIX_ADDR	Code Fix Address. This field is used for hardware debug purposes. CFIX_ADDR may change its contents and must not be written in automatic mode. For more information on error detection and correction, see Memory Error Support.
30–31 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

43.11 Global Channel memory map and registers

The registers in this section group, by type, the interrupt status and enable bits from all the channels. This organization eases management of all channels or groups of channels by a single interrupt handler routine. These bits, except the service and watchdog status, are mirrored in the individual channel registers, grouped by channel.

eTPU_GCR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
200	eTPU Channel Interrupt Status Register for Engine 1 (eTPU_GCR_CISR_ENG1)	32	R/W	0000_0000h	43.11.1/ 1409
204	eTPU Channel Interrupt Status Register for Engine 2 (eTPU_GCR_CISR_ENG2)	32	R/W	0000_0000h	43.11.2/ 1409
210	eTPU Channel Data Transfer Request Status Register for Engine 1 (eTPU_GCR_CDTRSR_ENG1)	32	R/W	0000_0000h	43.11.3/ 1410
214	eTPU Channel Data Transfer Request Status Register for Engine 2 (eTPU_GCR_CDTRSR_ENG2)	32	R/W	0000_0000h	43.11.4/ 1411
220	eTPU Channel Interrupt Overflow Status Register for Engine 1 (eTPU_GCR_CIOSR_ENG1)	32	R/W	0000_0000h	43.11.5/ 1412
224	eTPU Channel Interrupt Overflow Status Register for Engine 2 (eTPU_GCR_CIOSR_ENG2)	32	R/W	0000_0000h	43.11.6/ 1412
230	eTPU Channel Data Transfer Request Overflow Status Register for Engine 1 (eTPU_GCR_CDTROSR_ENG1)	32	R/W	0000_0000h	43.11.7/ 1413
234	eTPU Channel Data Transfer Request Overflow Status Register for Engine 2 (eTPU_GCR_CDTROSR_ENG2)	32	R/W	0000_0000h	43.11.8/ 1414
240	eTPU Channel Interrupt Enable Register for Engine 1 (eTPU_GCR_CIER_ENG1)	32	R/W	0000_0000h	43.11.9/ 1414
244	eTPU Channel Interrupt Enable Register for Engine 2 (eTPU_GCR_CIER_ENG2)	32	R/W	0000_0000h	43.11.10/ 1415
250	eTPU Channel Data Transfer Request Enable Register for Engine 1 (eTPU_GCR_CDTRER_ENG1)	32	R/W	0000_0000h	43.11.11/ 1416
254	eTPU Channel Data Transfer Request Enable Register for Engine 2 (eTPU_GCR_CDTRER_ENG2)	32	R/W	0000_0000h	43.11.12/ 1416
260	eTPU Watchdog Status Register for Engine 1 (eTPU_GCR_WDSR_ENG1)	32	R/W	0000_0000h	43.11.13/ 1417
264	eTPU Watchdog Status Register for Engine 2 (eTPU_GCR_WDSR_ENG2)	32	R/W	0000_0000h	43.11.14/ 1417
280	eTPU Channel Pending Service Status Register for Engine 1 (eTPU_GCR_CPSSR_ENG1)	32	R	0000_0000h	43.11.15/ 1418
284	eTPU Channel Pending Service Status Register for Engine 2 (eTPU_GCR_CPSSR_ENG2)	32	R	0000_0000h	43.11.16/ 1419
290	eTPU Channel Service Status Register for Engine 1 (eTPU_GCR_CSSR_ENG1)	32	R	0000_0000h	43.11.17/ 1419
294	eTPU Channel Service Status Register for Engine 2 (eTPU_GCR_CSSR_ENG2)	32	R	0000_0000h	43.11.18/ 1420

43.11.1 eTPU Channel Interrupt Status Register for Engine 1 (eTPU_GCR_CISR_ENG1)

Host interrupt status from all channels are grouped in ETPUCISR.These bits are mirrored from the Channel Status Control register (see eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C*n*SCR_ENG1)) and the Host must write 1 to clear a status bit.

For details about interrupts, see the Channel Interrupt and Data Tranfer Requests section.

Address: 0h base + 200h offset = 200h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CISn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CISR_ENG1 field descriptions

Field	Description
0–31 CISn	Channel n Interrupt Status. When read, each bit: 0.....Indicates that channel n has no pending interrupt to the CPU. 1.....Indicates that channel n has a pending interrupt to the CPU. When written: 0.....Keep interrupt status bit unaltered. 1.....Clear interrupt status bit.

43.11.2 eTPU Channel Interrupt Status Register for Engine 2 (eTPU_GCR_CISR_ENG2)

Host interrupt status from all channels are grouped in ETPUCISR.These bits are mirrored from the Channel Status Control register (see eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C*n*SCR_ENG2))) and the Host must write 1 to clear a status bit.

For details about interrupts, see the Channel Interrupt and Data Tranfer Requests section.

Address: 0h base + 204h offset = 204h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CISn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CISR_ENG2 field descriptions

Field	Description
0–31 CISn	Channel n Interrupt Status. When read, each bit: 0.....Indicates that channel n has no pending interrupt to the CPU. 1.....Indicates that channel n has a pending interrupt to the CPU. When written: 0.....Keep interrupt status bit unaltered. 1.....Clear interrupt status bit.

43.11.3 eTPU Channel Data Transfer Request Status Register for Engine 1 (eTPU_GCR_CDTRSR_ENG1)

Data Transfer request status (see the Interrupts and Data Transfer Requests section) from all channels are grouped in ETPUCDTRSR. Their bits are mirrored from the Channel Status Control registers (see eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C*n*SCR_ENG1)) and Host must write 1 to clear a status bit.

For details about interrupts, see the Interrupts section.

Address: 0h base + 210h offset = 210h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DTRSn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CDTRSR_ENG1 field descriptions

Field	Description
0–31 DTRSn	Channel n Data Transfer Request Status. These bits mimic the corresponding ETPU DMA requests. DTRSn can be cleared by software writing a 1 to it or by the assertion of corresponding DMA completion acknowledge line. For details, see the Interrupts and Data Transfer Requests section. When read: 0.....Indicates that channel n has no pending data transfer request. 1.....Indicates that channel x has a pending data transfer request. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.

43.11.4 eTPU Channel Data Transfer Request Status Register for Engine 2 (eTPU_GCR_CDTRSR_ENG2)

Data Transfer request status(see the Interrupts and Data Transfer Requests section) from all channels are grouped in ETPUCDTRSR.These bits are mirrored from the Channel Status Control registers (see eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C*n*SCR_ENG2)) and Host must write 1 to clear a status bit.

For details about interrupts, see the Interrupts section.

Address: 0h base + 214h offset = 214h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DTRSn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CDTRSR_ENG2 field descriptions

Field	Description
0–31 DTRSn	Channel n Data Transfer Request Status. These bits mimic the corresponding ETPU DMA requests. DTRSn can be cleared by software writing a 1 to it or by the assertion of corresponding DMA completion acknowledge line. For details, see the Interrupts and Data Transfer Requests section. When read: 0.....Indicates that channel n has no pending data transfer request. 1.....Indicates that channel x has a pending data transfer request. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.

43.11.5 eTPU Channel Interrupt Overflow Status Register for Engine 1 (eTPU_GCR_CIOSR_ENG1)

Interrupt Overflow status (see the Interrupts and Data Transfer Requests section) from all channels are grouped in ETPUCIOSR. These bits are mirrored from the Channel Status Control registers (see eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C*n*SCR_ENG1)) and the Host must write 1 to clear a status bit.

Address: 0h base + 220h offset = 220h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CIOSn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CIOSR_ENG1 field descriptions

Field	Description
0–31 CIOSn	Channel n Interrupt Overflow Status When read: 0.....Indicates that no interrupt overflow occurred in the channel. 1.....Indicates that interrupt overflow occurred in the channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.

43.11.6 eTPU Channel Interrupt Overflow Status Register for Engine 2 (eTPU_GCR_CIOSR_ENG2)

Interrupt Overflow status (see the Interrupts and Data Transfer Requests section) from all channels are grouped in ETPUCIOSR. These bits are mirrored from the Channel Status Control registers (see eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C*n*SCR_ENG2)) and the Host must write 1 to clear a status bit.

Address: 0h base + 224h offset = 224h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CIOSn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CIOSR_ENG2 field descriptions

Field	Description
0–31 CIOSn	Channel n Interrupt Overflow Status When read: 0.....Indicates that no interrupt overflow occurred in the channel. 1.....Indicates that interrupt overflow occurred in the channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.

43.11.7 eTPU Channel Data Transfer Request Overflow Status Register for Engine 1 (eTPU_GCR_CDTROSR_ENG1)

Data Transfer Request Overflow status (see the Interrupts and Data Transfer Requests section) from all channels are grouped in ETPUCDTROSR. These bits are mirrored from the Channel Status Control registers (see eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C*n*SCR_ENG1)) and the CPU must write 1 to clear a status bit.

Address: 0h base + 230h offset = 230h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DTROSn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CDTROSR_ENG1 field descriptions

Field	Description
0–31 DTROSn	Channel n Data Transfer Request Overflow Status. When read: 0.....Indicates that no data transfer request overflow occurred in the channel. 1.....Indicates that data transfer request overflow occurred in the channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.

43.11.8 eTPU Channel Data Transfer Request Overflow Status Register for Engine 2 (eTPU_GCR_CDTROSR_ENG2)

Data Transfer Request Overflow status (see the Interrupts and Data Transfer Requests section) from all channels are grouped in ETPUCDTROSR. These bits are mirrored from the Channel Status Control registers (see eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C*n*SCR_ENG2)) and the CPU must write 1 to clear a status bit.

Address: 0h base + 234h offset = 234h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DTROSn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CDTROSR_ENG2 field descriptions

Field	Description
0–31 DTROSn	Channel n Data Transfer Request Overflow Status. When read: 0.....Indicates that no data transfer request overflow occurred in the channel. 1.....Indicates that data transfer request overflow occurred in the channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.

43.11.9 eTPU Channel Interrupt Enable Register for Engine 1 (eTPU_GCR_CIER_ENG1)

Host interrupt enable (see the Interrupts and Data Transfer Requests section) from all channels are grouped in ETPUCIER. These bits are mirrored from the Channel Status Control registers (see eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C*n*SCR_ENG1)).

Address: 0h base + 240h offset = 240h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CIEn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CIER_ENG1 field descriptions

Field	Description
0–31 CIEn	Channel n Interrupt Enable. 0 Interrupt disabled for channel n. 1 Interrupt enabled for channel n.

43.11.10 eTPU Channel Interrupt Enable Register for Engine 2 (eTPU_GCR_CIER_ENG2)

Host interrupt enable (see the Interrupts and Data Transfer Requests section) from all channels are grouped in ETPUCIER. These bits are mirrored from the Channel Status Control registers (see eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C*n*SCR_ENG2)).

Address: 0h base + 244h offset = 244h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CIEn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CIER_ENG2 field descriptions

Field	Description
0–31 CIEn	Channel n Interrupt Enable. 0 Interrupt disabled for channel n. 1 Interrupt enabled for channel n.

43.11.11 eTPU Channel Data Transfer Request Enable Register for Engine 1 (eTPU_GCR_CDTRER_ENG1)

Data Transfer request enable (see the Interrupts and Data Transfer Requests section) from all channels are grouped in ETPUCDTRER. These bits are mirrored from the Channel Status Control registers (see eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C*n*SCR_ENG1)).

Address: 0h base + 250h offset = 250h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DTREn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CDTRER_ENG1 field descriptions

Field	Description
0–31 DTREn	Channel n Data Transfer Request Enable. 0 Data Transfer request disabled for channel n. 1 Data Transfer request enabled for channel n.

43.11.12 eTPU Channel Data Transfer Request Enable Register for Engine 2 (eTPU_GCR_CDTRER_ENG2)

Data Transfer request enable (see the Interrupts and Data Transfer Requests section) from all channels are grouped in ETPUCDTRER. These bits are mirrored from the Channel Status Control registers (see eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C*n*SCR_ENG2)).

Address: 0h base + 254h offset = 254h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DTREn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_CDTRER_ENG2 field descriptions

Field	Description
0–31 DTREn	Channel n Data Transfer Request Enable.

eTPU_GCR_CDTRER_ENG2 field descriptions (continued)

Field	Description
	0 Data Transfer request disabled for channel n. 1 Data Transfer request enabled for channel n.

43.11.13 eTPU Watchdog Status Register for Engine 1 (eTPU_GCR_WDSR_ENG1)

ETPUWDSR indicates the watchdog influence on each of the engine channels.

Address: 0h base + 260h offset = 260h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WDSn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_WDSR_ENG1 field descriptions

Field	Description
0–31 WDSn	Channel n Watchdog Status. For details about Watchdog mechanism, see the Watchdog section. **NOTE:** The WDSn bits set in case of a watchdog timeout independently of the ETPUWDTR bit WMO. When read: 0.....Watchdog did not expire during execution of a channel x thread since reset or last WDSn clear. 1.....Watchdog expired during execution of a channel n thread. When written: 0.....Keep watchdog status bit unaltered. 1.....Clear watchdog status bit.

43.11.14 eTPU Watchdog Status Register for Engine 2 (eTPU_GCR_WDSR_ENG2)

ETPUWDSR indicates the watchdog influence on each of the engine channels.

Address: 0h base + 264h offset = 264h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WDSn																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_GCR_WDSR_ENG2 field descriptions

Field	Description
0–31 WDSn	Channel n Watchdog Status. For details about Watchdog mechanism, see the Watchdog section. **NOTE:** The WDSn bits set in case of a watchdog timeout independently of the ETPUWDTR bit WMO. When read: 0.....Watchdog did not expire during execution of a channel x thread since reset or last WDSn clear. 1.....Watchdog expired during execution of a channel n thread. When written: 0.....Keep watchdog status bit unaltered. 1.....Clear watchdog status bit.

43.11.15 eTPU Channel Pending Service Status Register for Engine 1 (eTPU_GCR_CPSSR_ENG1)

ETPUCPSSR is a read-only register that holds the status of the pending Channel Service Requests (see the Functions and Threads section).

Address: 0h base + 280h offset = 280h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SRn																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

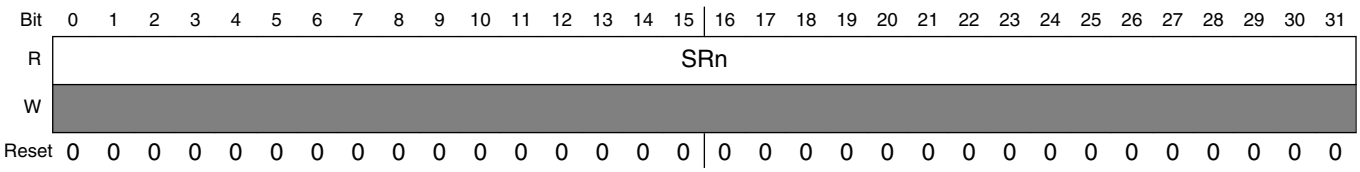
eTPU_GCR_CPSSR_ENG1 field descriptions

Field	Description
0–31 SRn	Pending Service Request n. Indicates a pending Service Request for channel n. Pending SR status is a logic OR of all service requests pending: if only HSR is active, SRn clears only at the end of the thread. SRn clear due to the other request sources is microcode dependent. **NOTE:** The pending service status bit for a channel is 1 when a Service Request is pending, even if the Channel is disabled (CPRn = 0). **NOTE:** There can be a delay of one clock between writing HSR greater than 0 in register ETPUCxHSRR of a channel and its respective bit being asserted in ETPUCPSSR. 0 No Service Request pending for channel n. 1 Pending Service Request for channel n.

43.11.16 eTPU Channel Pending Service Status Register for Engine 2 (eTPU_GCR_CPSSR_ENG2)

ETPUCPSSR is a read-only register that holds the status of the pending Channel Service Requests (see the Functions and Threads section).

Address: 0h base + 284h offset = 284h



eTPU_GCR_CPSSR_ENG2 field descriptions

Field	Description
0–31 SRn	Pending Service Request n. Indicates a pending Service Request for channel n. Pending SR status is a logic OR of all service requests pending: if only HSR is active, SRn clears only at the end of the thread. SRn clear due to the other request sources is microcode dependent. **NOTE:** The pending service status bit for a channel is 1 when a Service Request is pending, even if the Channel is disabled (CPRn = 0). **NOTE:** There can be a delay of one clock between writing HSR greater than 0 in register ETPUCxHSRR of a channel and its respective bit being asserted in ETPUCPSSR. 0 No Service Request pending for channel n. 1 Pending Service Request for channel n.

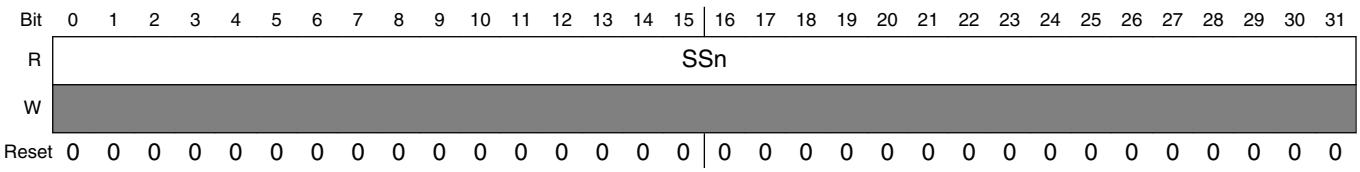
43.11.17 eTPU Channel Service Status Register for Engine 1 (eTPU_GCR_CSSR_ENG1)

ETPUCSSR holds the current channel service status on whether it is being serviced or not (see the Functions and Threads section). Only one bit may be asserted in this register at a given time. When no channel is being serviced, the register read value is 0x0000_0000. ETPUCSSR is a read-only register. The register can be read during normal eTPU operation for monitoring the scheduler activity.

NOTE

Channel Service Status does not always reflect decoding of the CHAN register, since the latter can be changed by the service thread microcode.

Address: 0h base + 290h offset = 290h



eTPU_GCR_CSSR_ENG1 field descriptions

Field	Description
0–31 SSn	Service Status n. Indicates that channel n is currently being serviced. It is updated at the 1st microcycle of a Time Slot Transition (see Time Slot Transition), or when the microengine ends the thread. 0 Channel n is not currently being serviced. 1 channel n is currently being serviced.

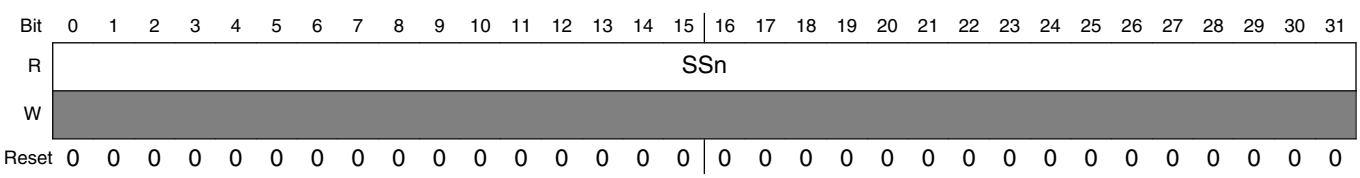
43.11.18 eTPU Channel Service Status Register for Engine 2 (eTPU_GCR_CSSR_ENG2)

ETPUCSSR holds the current channel service status on whether it is being serviced or not (see the Functions and Threads section). Only one bit may be asserted in this register at a given time. When no channel is being serviced, the register read value is 0x0000_0000. ETPUCSSR is a read-only register. The register can be read during normal eTPU operation for monitoring the scheduler activity.

NOTE

Channel Service Status does not always reflect decoding of the CHAN register, since the latter can be changed by the service thread microcode.

Address: 0h base + 294h offset = 294h



eTPU_GCR_CSSR_ENG2 field descriptions

Field	Description
0–31 SSn	Service Status n. Indicates that channel n is currently being serviced. It is updated at the 1st microcycle of a Time Slot Transition (see Time Slot Transition), or when the microengine ends the thread.

eTPU_GCR_CSSR_ENG2 field descriptions (continued)

Field	Description
	0 Channel n is not currently being serviced. 1 channel n is currently being serviced.

43.12 Channel Configuration Control memory map and registers

Each channel has a group of 3 registers used to control, configure and check status of that channel as shown in the following table. This organization eases individual channel management.

Table 43-120. Channel Registers Structure

Channel Offset	Register Name
0x00	ETPUCnCR - eTPU Channel Configuration Register
0x04	ETPUCnSCR - eTPU Channel Status/Control Register
0x08	ETPUCnHSRR - eTPU Channel Host Service Request Register
0x0C	RESERVED

One contiguous area is used to map all channel registers of each eTPU engine as shown in the following table.

Table 43-121. Channel Registers Map

Offset	Registers Structure
0x400	eTPU 1 Channel 0 Registers Structure
0x410	eTPU 1 Channel 1 Registers Structure
0x420	eTPU 1 Channel 2 Registers Structure
0x430	-----
0x5E0	eTPU 1 Channel 30 Registers Structure
0x5F0	eTPU 1 Channel 31 Registers Structure
0x600	RESERVED
0x800	eTPU 2 Channel 0 Registers Structure
0x810	eTPU 2 Channel 1 Registers Structure
0x820	-----
0x9E0	eTPU 2 Channel 30 Registers Structure
0x9F0	eTPU 2 Channel 31 Registers Structure
0xA00	RESERVED

There are 64 structures defined, one for each available channel in the eTPU System (32 for each Engine). The base address for the structure presented can be calculated by using the following equation:

Channel_Register_Base = ETPU_Engine_Channel_Base + (channel_number * 0x10), where:

- ETPU_Engine_Channel_Base = ETPU_Base + 0x400 for Engine 1.
- ETPU_Engine_Channel_Base = ETPU_Base + 0x800 for Engine 2.

eTPU_CCCR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C0CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
404	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C0SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
408	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C0HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
410	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C1CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
414	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C1SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
418	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C1HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
420	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C2CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
424	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C2SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
428	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C2HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
430	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C3CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
434	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C3SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
438	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C3HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
440	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C4CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
444	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C4SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
448	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C4HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
450	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C5CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
454	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C5SCR_ENG1)	32	R/W	See section	43.12.2/ 1434

Table continues on the next page...

eTPU_CCCR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
458	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C5HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
460	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C6CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
464	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C6SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
468	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C6HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
470	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C7CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
474	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C7SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
478	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C7HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
480	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C8CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
484	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C8SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
488	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C8HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
490	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C9CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
494	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C9SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
498	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C9HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
4A0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C10CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
4A4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C10SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
4A8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C10HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
4B0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C11CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
4B4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C11SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
4B8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C11HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
4C0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C12CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
4C4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C12SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
4C8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C12HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436

Table continues on the next page...

eTPU_CCCR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4D0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C13CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
4D4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C13SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
4D8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C13HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
4E0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C14CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
4E4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C14SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
4E8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C14HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
4F0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C15CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
4F4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C15SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
4F8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C15HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
500	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C16CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
504	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C16SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
508	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C16HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
510	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C17CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
514	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C17SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
518	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C17HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
520	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C18CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
524	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C18SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
528	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C18HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
530	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C19CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
534	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C19SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
538	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C19HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
540	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C20CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431

Table continues on the next page...

eTPU_CCCR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
544	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C20SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
548	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C20HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
550	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C21CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
554	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C21SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
558	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C21HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
560	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C22CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
564	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C22SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
568	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C22HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
570	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C23CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
574	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C23SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
578	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C23HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
580	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C24CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
584	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C24SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
588	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C24HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
590	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C25CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
594	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C25SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
598	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C25HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
5A0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C26CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
5A4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C26SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
5A8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C26HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
5B0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C27CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
5B4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C27SCR_ENG1)	32	R/W	See section	43.12.2/ 1434

Table continues on the next page...

eTPU_CCCR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5B8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C27HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
5C0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C28CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
5C4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C28SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
5C8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C28HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
5D0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C29CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
5D4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C29SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
5D8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C29HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
5E0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C30CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
5E4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C30SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
5E8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C30HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
5F0	eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C31CR_ENG1)	32	R/W	0000_0000h	43.12.1/ 1431
5F4	eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C31SCR_ENG1)	32	R/W	See section	43.12.2/ 1434
5F8	eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C31HSRR_ENG1)	32	R/W	0000_0000h	43.12.3/ 1436
800	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C0CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
804	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C0SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
808	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C0HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
810	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C1CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
814	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C1SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
818	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C1HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
820	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C2CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
824	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C2SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
828	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C2HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442

Table continues on the next page...

eTPU_CCCR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
830	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C3CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
834	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C3SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
838	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C3HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
840	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C4CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
844	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C4SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
848	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C4HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
850	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C5CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
854	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C5SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
858	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C5HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
860	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C6CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
864	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C6SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
868	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C6HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
870	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C7CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
874	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C7SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
878	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C7HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
880	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C8CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
884	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C8SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
888	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C8HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
890	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C9CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
894	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C9SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
898	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C9HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
8A0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C10CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437

Table continues on the next page...

eTPU_CCCR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8A4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C10SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
8A8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C10HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
8B0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C11CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
8B4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C11SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
8B8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C11HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
8C0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C12CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
8C4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C12SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
8C8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C12HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
8D0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C13CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
8D4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C13SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
8D8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C13HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
8E0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C14CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
8E4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C14SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
8E8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C14HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
8F0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C15CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
8F4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C15SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
8F8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C15HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
900	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C16CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
904	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C16SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
908	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C16HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
910	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C17CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
914	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C17SCR_ENG2)	32	R/W	See section	43.12.5/ 1440

Table continues on the next page...

eTPU_CCCR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
918	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C17HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
920	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C18CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
924	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C18SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
928	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C18HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
930	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C19CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
934	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C19SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
938	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C19HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
940	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C20CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
944	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C20SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
948	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C20HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
950	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C21CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
954	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C21SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
958	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C21HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
960	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C22CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
964	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C22SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
968	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C22HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
970	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C23CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
974	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C23SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
978	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C23HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
980	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C24CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
984	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C24SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
988	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C24HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442

Table continues on the next page...

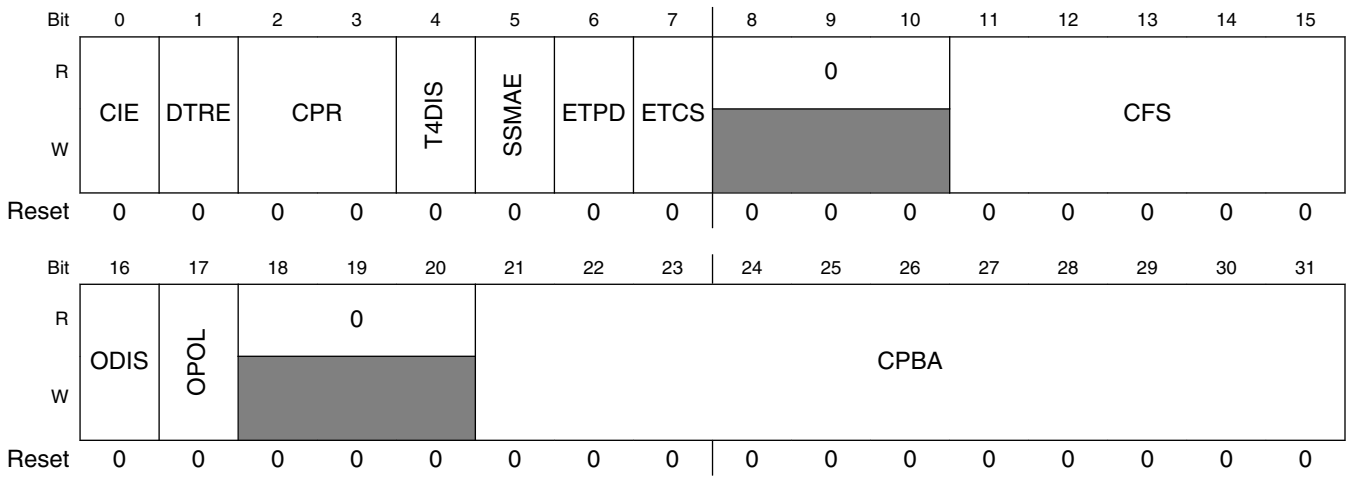
eTPU_CCCR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
990	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C25CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
994	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C25SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
998	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C25HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
9A0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C26CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
9A4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C26SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
9A8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C26HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
9B0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C27CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
9B4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C27SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
9B8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C27HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
9C0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C28CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
9C4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C28SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
9C8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C28HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
9D0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C29CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
9D4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C29SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
9D8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C29HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
9E0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C30CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
9E4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C30SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
9E8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C30HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442
9F0	eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C31CR_ENG2)	32	R/W	0000_0000h	43.12.4/ 1437
9F4	eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C31SCR_ENG2)	32	R/W	See section	43.12.5/ 1440
9F8	eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C31HSRR_ENG2)	32	R/W	0000_0000h	43.12.6/ 1442

43.12.1 eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1)

ETPUCnCR gathers configurations set individually per channel.

Address: 0h base + 400h offset + (16d × i), where i=0d to 31d



eTPU_CCCR_C*n*CR_ENG1 field descriptions

Field	Description
0 CIE	Channel Interrupt Enable (see the eTPU Channel Interrupt Enable Register (ETPUCIER)). This bit is mirrored from eTPU Channel Interrupt Enable Register for Engine 2 (eTPU_GCR_CIER_ENG2). 0 Disable interrupt for this channel. 1 Enable interrupt for this channel.
1 DTRE	Channel Data Transfer Request Enable (see the “Channel Interrupt and Data Transfer Requests" section). This bit is mirrored from eTPU Channel Data Transfer Request Enable Register for Engine 2 (eTPU_GCR_CDTRER_ENG2). 0 Disable data transfer request for this channel. 1 Enable data transfer request for this channel.
2–3 CPR	Channel Priority. This field defines the priority level for the channel, used by the Hardware Scheduler (see the Scheduler section). 00 Disabled 01 Low 10 Middle 11 High

Table continues on the next page...

eTPU_CCCR_C*n*CR_ENG1 field descriptions (continued)

Field	Description
4 T4DIS	T4 channel action Disable. This bit forces the channel to work on T2 timing mode, regardless of the ETPUECR fields FCSS and CDFC (see eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2)), or of the ETPUTBCR field TCR1CS (see eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2)). For more details see The Channel Timing Modes section. 0 Channel works either in T2 or T2/T4 timing modes, depending on ETPUTBCR field TCR1CS and ETPUECR fields FCSS and CDFC. 1 Channel works in T2 timing mode, regardless of ETPUTBCR field TCR1CS or ETPUECR fields FCSS and CDFC.
5 SSMAE	Shared Subsystem Microengine Access Enable. This bit enables the microengine access to the Shared Subsystem Area (see the Shared Subsystems Support section). When this bit is negated, DIOB address values pointing to the Shared Subsystem Area actually access SDM mirrors, in legacy mode. **NOTE:** SSMAE value must only be changed when the engine is either stopped or halted. 0 An SDM mirror is accessed when the Shared Subsystem Area is addressed by the microengine. 1 Enables microengine access to the Shared Subsystem Area.
6 ETPD	Entry Table Pin Direction. This bit selects which channel signal, input or output, is used in the Entry Point selection. The ETPD value has to be compatible with the function chosen for the channel, selected in the field CFS. For details about Entry Table and condition encoding schemes, refer to the “Entry Points” section. 0 Use PSTI for Entry Point selection. 1 Use PSTO for Entry Point selection.
7 ETCS	Entry Table Condition Select. This bit determines the channel condition encoding scheme that selects, according to channel conditions, the Entry Point to be taken in an Entry Table. ETCS value has to be compatible with the function chosen for the channel, selected in field CFS. Two condition encoding schemes are available. For details about Entry Table and condition encoding schemes, refer to the “Entry Points” section. 0 Select Standard Entry Table Condition encoding scheme. 1 Select Alternate Entry Table Condition encoding scheme.
8–10 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
11–15 CFS	Channel Function Select. This field defines the function to be performed by the channel (see the Functions and Threads section). The Function assigned to the channel has to be compatible with the channel condition encoding scheme, selected by field ETCS.
16 ODIS	Output Disable. This bit enables the channel to have its output forced to the value opposite to OPOL when the output disable input signal corresponding to the channel group that it belongs is active (see eTPU Channel Output Disable Signals and .

Table continues on the next page...

eTPU_CCCR_C*n*CR_ENG1 field descriptions (continued)

Field	Description
	0 Turns off the output disable feature for the channel. 1 Turns on the output disable feature for the channel.
17 OPOL	Output Polarity. Determines the output signal polarity. The activation of the output disable signal forces, when enabled by the ODIS bit, the channel output signal to the opposite of this polarity (see). 0 Output active low (output disable drives output to high). 1 output active high (output disable drives output to low).
18–20 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
21–31 CPBA	Channel n Parameter Base Address. The value of this field times 8 specifies the SDM parameter base host (byte) address for channel n (2-parameter granularity; see the SDM Organization section.) As seen by the Host, the channel parameter base (byte) address is (without parameter sign extension) ETPU_Base + 0x8000 + CPBA * 8 or (with parameter sign extension), ETPU_Base + 0xC000 + CPBA*8.

43.12.2 eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C*n*SCR_ENG1)

ETPUCnSCR gathers the interrupt status bits of the channel, and also the Function Mode definition (read-write). Bits CIS, CIOS and DTRS for each channel can be also accessed from ETPUCISR (see eTPU Channel Interrupt Status Register for Engine 1 (eTPU_GCR_CISR_ENG1)), ETPUCIOSR (see eTPU Channel Interrupt Overflow Status Register for Engine 1 (eTPU_GCR_CIOSR_ENG1)) and ETPUCDTRSR (see eTPU Channel Data Transfer Request Status Register for Engine 1 (eTPU_GCR_CDTRSR_ENG1)). Host must write 1 to clear a status bit.

Address: 0h base + 404h offset + (16d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CIS	CIOS	0						DTRS	DTROS	0					
W	w1c	w1c							w1c	w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IPS	OPS	0	0											FM	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_CCCR_C*n*SCR_ENG1 field descriptions

Field	Description
0 CIS	Channel Interrupt Status. These bits are mirrored in ETPUCISR. Also, see the Channel Interrupt and Data Transfer Requests section. When read: 0.....Channel has no pending interrupt to the Host CPU. 1.....Channel has a pending interrupt to the Host CPU. When written: 0.....Keep interrupt status bit unaltered.

Table continues on the next page...

eTPU_CCCR_C*n*SCR_ENG1 field descriptions (continued)

Field	Description
	1.....Clear interrupt status bit.
1 CIOS	Channel Interrupt Overflow Status. These bits are mirrored in eTPU Channel Interrupt Overflow Status Register for Engine 1 (eTPU_GCR_CIOSR_ENG1). Also, see the Interrupt and Data Request Overflow section. When read: 0.....Interrupt overflow negated for this channel. 1.....Interrupt overflow asserted for this channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.
2–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8 DTRS	Data Transfer Request Status. These bits are mirrored in eTPU Channel Data Transfer Request Status Register for Engine 1 (eTPU_GCR_CDTRSR_ENG1). Also, see the Channel Interrupt and Data Requests section. When read: 0.....Channel has no pending data transfer request. 1.....Channel has a pending data transfer request. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.
9 DTROS	Data Transfer Request Overflow Status. These bits are mirrored in eTPU Channel Data Transfer Request Overflow Status Register for Engine 1 (eTPU_GCR_CDTROSR_ENG1). Also, see the Interrupt and Data Transfer Request Overflow section. When read: 0.....Data transfer request overflow negated for this channel. 1.....Data transfer request overflow asserted for this channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.
10–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16 IPS	Channel Input Pin State. This bit shows the current value of the filtered channel input signal state.
17 OPS	Channel Output Pin State.

Table continues on the next page...

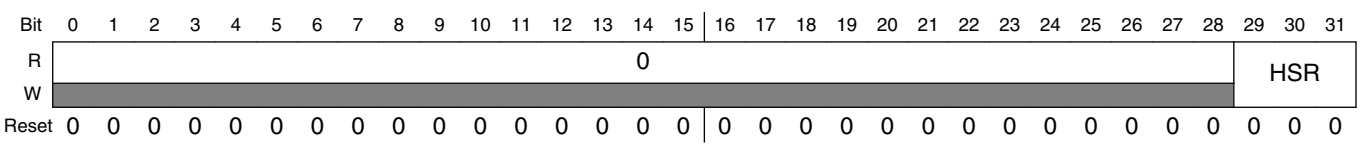
eTPU_CCCR_C*n*SCR_ENG1 field descriptions (continued)

Field	Description
	This bit shows the current value driven in the channel output signal, including the effect of the external output disable feature (see eTPU Channel Output Disable Signals). If the channel input and output signals are connected to the same pad, OPS reflects the value driven to the pad (if OBE=1). This is not necessarily the actual pad value, which drives the value in the bit IPS.
18 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
19–29 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
30–31 FM	Channel Function Mode. Each function uses this field for a specific configuration. These bits can be tested by microengine code (see Conditional/Unconditional Branch). These bits are equivalent to the TPU/TPU2/TPU3 Host Sequence (HSQ) bits.

43.12.3 eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C*n*HSRR_ENG1)

ETPUCnHSRR is used by the Host to issue service requests to the channel.

Address: 0h base + 408h offset + (16d × i), where i=0d to 31d



eTPU_CCCR_C*n*HSRR_ENG1 field descriptions

Field	Description
0–28 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
29–31 HSR	Host Service Request This field is used by the Host CPU to request service to the channel (see the Host Service Requests section). HSR value turns to 000 automatically at the end of microengine service for that channel, but only if the thread started due to an HSR. Host should write HSR greater than 0 only when HSR=0. Writing HSR=000 withdraws a pending request if scheduler did not begin to resolve the Entry Point yet, but it does not abort the service thread from that point on. For more details, see the Entry Points and Host Service Requests sections.

Table continues on the next page...

eTPU_CCCR_C*n*HSRR_ENG1 field descriptions (continued)

Field	Description
	000 No Host Service Request pending. 001 Function-dependent Host Service Request pending. --- --- 111 Function-dependent Host Service Request pending.

43.12.4 eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2)

ETPUCnCR gathers configurations set individually per channel.

Address: 0h base + 800h offset + (16d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CIE	DTRE	CPR		T4DIS	SSMAE	ETPD	ETCS	0			CFS				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ODIS	OPOL	0			CPBA										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_CCCR_C*n*CR_ENG2 field descriptions

Field	Description
0 CIE	Channel Interrupt Enable (see the Channel Interrupts ad Data Transfer Requests section). This bit is mirrored from eTPU Channel Interrupt Enable Register for Engine 2 (eTPU_GCR_CIER_ENG2). 0 Disable interrupt for this channel. 1 Enable interrupt for this channel.
1 DTRE	Channel Data Transfer Request Enable (see the “Channel Interrupt and Data Transfer Requests" section). This bit is mirrored from eTPU Channel Data Transfer Request Enable Register for Engine 2 (eTPU_GCR_CDTRER_ENG2). 0 Disable data transfer request for this channel. 1 Enable data transfer request for this channel.

Table continues on the next page...

eTPU_CCCR_C*n*CR_ENG2 field descriptions (continued)

Field	Description
2–3 CPR	Channel Priority. This field defines the priority level for the channel, used by the Hardware Scheduler (see the Scheduler section). 00 Disabled 01 Low 10 Middle 11 High
4 T4DIS	T4 channel action Disable. This bit forces the channel to work on T2 timing mode, regardless of the ETPUECR fields FCSS and CDFC, or of the ETPUTBCR field TCR1CS. 0 Channel works either in T2 or T2/T4 timing modes, depending on ETPUTBCR field TCR1CS and ETPUECR fields FCSS and CDFC. 1 Channel works in T2 timing mode, regardless of ETPUTBCR field TCR1CS or ETPUECR fields FCSS and CDFC.
5 SSMAE	Shared Subsystem Microengine Access Enable. This bit enables the microengine access to the Shared Subsystem Area (see the Shared Subsystems Support section). When this bit is negated, DIOB address values pointing to the Shared Subsystem Area actually access SDM mirrors, in legacy mode. **NOTE:** SSMAE value must only be changed when the engine is either stopped or halted. 0 An SDM mirror is accessed when the Shared Subsystem Area is addressed by the microengine. 1 Enables microengine access to the Shared Subsystem Area.
6 ETPD	Entry Table Pin Direction. This bit selects which channel signal, input or output, is used in the Entry Point selection. The ETPD value has to be compatible with the function chosen for the channel, selected in the field CFS. For details about Entry Table and condition encoding schemes, refer to the “Entry Points” section. 0 Use PSTI for Entry Point selection. 1 Use PSTO for Entry Point selection.
7 ETCS	Entry Table Condition Select. This bit determines the channel condition encoding scheme that selects, according to channel conditions, the Entry Point to be taken in an Entry Table. ETCS value has to be compatible with the function chosen for the channel, selected in field CFS. Two condition encoding schemes are available. For details about Entry Table and condition encoding schemes, refer to the “Entry Points” section. 0 Select Standard Entry Table Condition encoding scheme. 1 Select Alternate Entry Table Condition encoding scheme.
8–10 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
11–15 CFS	Channel Function Select.

Table continues on the next page...

eTPU_CCCR_C*n*CR_ENG2 field descriptions (continued)

Field	Description
	This field defines the function to be performed by the channel (see the Functions and Threasds section). The Function assigned to the channel has to be compatible with the channel condition encoding scheme, selected by field ETCS.
16 ODIS	Output Disable. This bit enables the channel to have its output forced to the value opposite to OPOL when the output disable input signal corresponding to the channel group that it belongs is active (see eTPU Channel Output Disable Signals and . 0 Turns off the output disable feature for the channel. 1 Turns on the output disable feature for the channel.
17 OPOL	Output Polarity. Determines the output signal polarity. The activation of the output disable signal forces, when enabled by the ODIS bit, the channel output signal to the opposite of this polarity (see). 0 Output active low (output disable drives output to high). 1 output active high (output disable drives output to low).
18–20 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
21–31 CPBA	Channel n Parameter Base Address. The value of this field times 8 specifies the SDM parameter base host (byte) address for channel n (2-parameter granularity; see the SDM Organization section.) As seen by the Host, the channel parameter base (byte) address is (without parameter sign extension) ETPU_Base + 0x8000 + CPBA * 8 or (with parameter sign extension), ETPU_Base + 0xC000 + CPBA*8.

43.12.5 eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C*n*SCR_ENG2)

ETPUCnSCR gathers the interrupt status bits of the channel, and also the Function Mode definition (read-write). Bits CIS, CIOS and DTRS for each channel can be also accessed from ETPUCISR (see eTPU Channel Interrupt Status Register for Engine 2 (eTPU_GCR_CISR_ENG2)), ETPUCIOSR (see eTPU Channel Interrupt Overflow Status Register for Engine 2 (eTPU_GCR_CIOSR_ENG2)) and ETPUCDTRSR (see eTPU Channel Data Transfer Request Status Register for Engine 2 (eTPU_GCR_CDTRSR_ENG2)). Host must write 1 to clear a status bit.

Address: 0h base + 804h offset + (16d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CIS	CIOS	0						DTRS	DTROS	0					
W	w1c	w1c							w1c	w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IPS	OPS	0	0											FM	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_CCCR_C*n*SCR_ENG2 field descriptions

Field	Description
0 CIS	Channel Interrupt Status. These bits are mirrored in eTPU Channel Interrupt Status Register for Engine 2 (eTPU_GCR_CISR_ENG2). Also, see the Channel Interrupt and Data Transfer Requests section. When read: 0.....Channel has no pending interrupt to the Host CPU. 1.....Channel has a pending interrupt to the Host CPU. When written: 0.....Keep interrupt status bit unaltered.

Table continues on the next page...

eTPU_CCCR_C*n*SCR_ENG2 field descriptions (continued)

Field	Description
	1.....Clear interrupt status bit.
1 CIOS	Channel Interrupt Overflow Status. These bits are mirrored in eTPU Channel Interrupt Overflow Status Register for Engine 2 (eTPU_GCR_CIOSR_ENG2). Also, see the Interrupt and Data Transfer Request Overflow section. When read: 0.....Interrupt overflow negated for this channel. 1.....Interrupt overflow asserted for this channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.
2–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8 DTRS	Data Transfer Request Status. These bits are mirrored in eTPU Channel Interrupt Status Register for Engine 2 (eTPU_GCR_CISR_ENG2). Also, see the Channel Interrupt and Data Transfer Requests section. When read: 0.....Channel has no pending data transfer request. 1.....Channel has a pending data transfer request. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.
9 DTROS	Data Transfer Request Overflow Status. These bits are mirrored in eTPU Channel Data Transfer Request Overflow Status Register for Engine 2 (eTPU_GCR_CDTROSR_ENG2). Also, see the Interrupt and Data Transfer Request Overflow section. When read: 0.....Data transfer request overflow negated for this channel. 1.....Data transfer request overflow asserted for this channel. When written: 0.....Keep status bit unaltered. 1.....Clear status bit.
10–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16 IPS	Channel Input Pin State. This bit shows the current value of the filtered channel input signal state.
17 OPS	Channel Output Pin State.

Table continues on the next page...

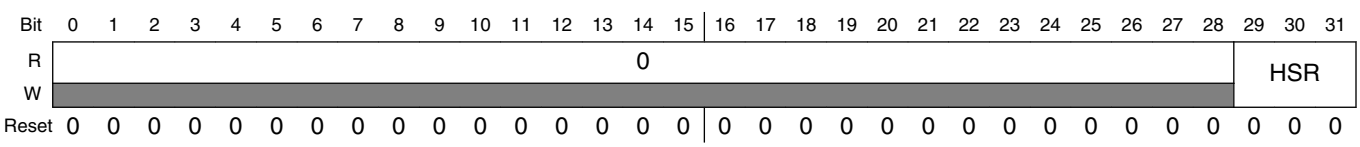
eTPU_CCCR_C*n*SCR_ENG2 field descriptions (continued)

Field	Description
	This bit shows the current value driven in the channel output signal, including the effect of the external output disable feature (see eTPU Channel Output Disable Signals). If the channel input and output signals are connected to the same pad, OPS reflects the value driven to the pad (if OBE=1). This is not necessarily the actual pad value, which drives the value in the bit IPS.
18 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
19–29 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
30–31 FM	Channel Function Mode. Each function uses this field for a specific configuration. These bits can be tested by microengine code (see Conditional/Unconditional Branch). These bits are equivalent to the TPU/TPU2/TPU3 Host Sequence (HSQ) bits.

43.12.6 eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C*n*HSRR_ENG2)

ETPUCnHSRR is used by the Host to issue service requests to the channel.

Address: 0h base + 808h offset + (16d × i), where i=0d to 31d



eTPU_CCCR_C*n*HSRR_ENG2 field descriptions

Field	Description
0–28 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
29–31 HSR	Host Service Request This field is used by the Host CPU to request service to the channel (see the Host Service Requests section). HSR value turns to 000 automatically at the end of microengine service for that channel, but only if the thread started due to an HSR. Host should write HSR greater than 0 only when HSR=0. Writing HSR=000 withdraws a pending request if scheduler did not begin to resolve the Entry Point yet, but it does not abort the service thread from that point on. For more details, see the Entry Points and Host Service Requests sections.

Table continues on the next page...

eTPU_CCCR_C*n*HSRR_ENG2 field descriptions (continued)

Field	Description
	000 No Host Service Request pending. 001 Function-dependent Host Service Request pending. --- --- 111 Function-dependent Host Service Request pending.

43.13 Shared Subsystem Integration memory map and registers

Shared Subsystems Integration Registers configure aspects of how the shared modules interface with the eTPU. There is one integration register for each subsystem unit, which can be, depending on the specific subsystem, a whole IP module, a sub-module, or a channel within a module. For more details, Shared Subsystems Support.

eTPU_SSIR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C00	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR0)	32	R/W	0000_0000h	43.13.1/ 1444
C04	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR1)	32	R/W	0000_0000h	43.13.1/ 1444
C08	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR2)	32	R/W	0000_0000h	43.13.1/ 1444
C0C	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR3)	32	R/W	0000_0000h	43.13.1/ 1444
C10	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR4)	32	R/W	0000_0000h	43.13.1/ 1444
C14	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR5)	32	R/W	0000_0000h	43.13.1/ 1444
C18	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR6)	32	R/W	0000_0000h	43.13.1/ 1444
C1C	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR7)	32	R/W	0000_0000h	43.13.1/ 1444
C20	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR8)	32	R/W	0000_0000h	43.13.1/ 1444
C24	eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR9)	32	R/W	0000_0000h	43.13.1/ 1444

43.13.1 eTPU Shared Subsystem n Integration Register (eTPU_SSIR_ETPUSSSIR*n*)

This register is used to configure the interface of a subsystem unit with the eTPU.

Address: 0h base + C00h offset + (4d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								LENG		0	LCHAN				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eTPU_SSIR_ETPUSSSIR*n* field descriptions

Field	Description
0–23 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
24–25 LENG	Link Engine. LENG specifies the engine that receives a Link Service Request when an interrupt is issued by the Shared Unit, or disables Shared Unit linking. The channel that receives the Link Service Request is specified in the LCHAN field. 00 Shared Unit Link Service Request disabled 01 Engine 1 receives Shared Unit Link Service Request 10 Engine 2 receives Shared Unit Link Service Request 11 Reserved
26 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
27–31 LCHAN	Link Channel. LCHAN specifies the channel linked when an interrupt is issued by the Shared Unit, if enabled in the LENG field. The field is meaningless if LENG disables Shared Unit linking. For more information on Link Service Requests, see Channel Link.

43.14 Functional Description

43.14.1 Functions and Threads

eTPU processing is event-driven, in the sense that eTPU microcode only runs to service a request from an event. Service Requests may result from the occurrence of any of the following events:

- Host CPU writing a non-zero value to the channel HSR (Host Service Request) field in ETPUCxHSR register.
- occurrence of a time base match, an input signal transition, or a specific combination of them (depending on the Channel Mode currently configured).
- a Link Service Request.

A given event is always associated to only one Channel:

- There is one HSR register field for each Channel
- Each signal is associated with only one Channel, which has its own Match registers and independent mode configuration.
- Each Link Service Request can have only one Channel as a target.

Service Request processing is done by a set of microengine routines. A set of related routines that implement a specific channel application is called a **Function** . One or more Functions reside on SCM, limited only by the SCM space available, size of microcode Functions and the number of entry points available. Each Engine can be controlled by up to 32 Functions at a time.

A Function can be assigned to several channels, but only one Function can be assigned to a given Channel at a time. This is defined by the Host through the Channel Configuration Registers (see Channel Configuration and Control registers).

The term **Thread** will be used hereafter to refer to a service routine of a Function, or its execution. A Thread is constructed of a specific number of microinstructions, typically the code necessary to calculate the next phase of waveform to be input to, or output from, a given channel. Once a Thread begins, its execution cannot be interrupted. A Thread normally finishes when an END microinstruction is executed.

A given Thread is selected and called by the Scheduler depending on the following:

- the type of event that generated the service request.
- the Function assigned to the target channel.

- target channel pin state.
- the state of the channel logic.
- the priority assigned to the target channel, relative to the priorities of other channels with pending service requests

The mechanism to select a thread based on the channel Function and type of event is described in the Entry Points.

The priority mechanism that determines the order of Thread execution amongst pending service requests is described in Scheduler.

43.14.1.1 Entry Points

43.14.1.1.1 Entry Table

Each Thread has its **Entry Point**, which contains the SCM address of its first instruction, besides other information. For a complete Entry Point description, see Entry Point Format.

Once the Scheduler chooses a channel among pending Service Requests, the Entry Point is taken from an **Entry Table**, based on the Function assigned for the channel and other conditions. Entry Table layout is shown in the following figure.

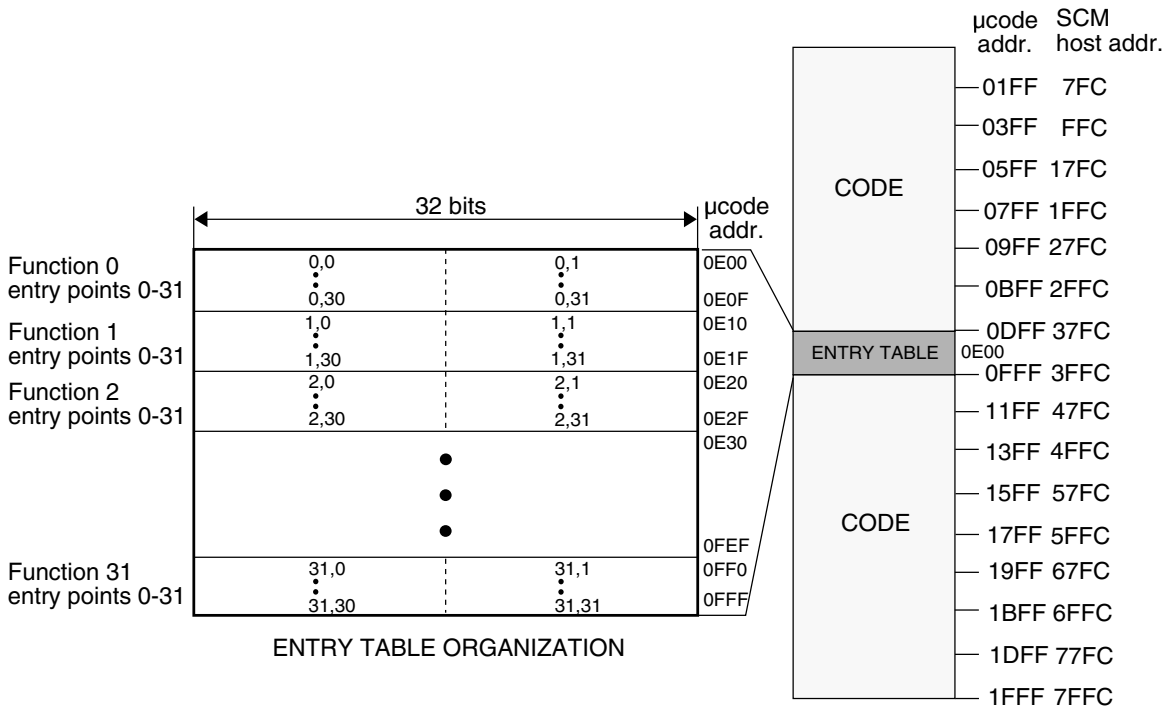


Figure 43-291. Entry Table

The Entry Table is organized by Functions. Each Function can have up to 32 Entry Points of 16 bits each, corresponding to 32 possible Threads per Function. Each Entry Point location in the table corresponds to a combination of events and channel states (see Entry Point Address Generation). A single Thread can be associated to more than one combination, having its Entry Point repeated in the table. Each 32-bit word in the Entry Table holds two Entry Points.

Note that the Entry Table can be placed in any SCM address multiple of the Entry Table size, determined by the field ETB in the register ETPUECR. However, it is recommended to place the Entry Table at the start of the SCM to get continuous code memory and to ease the eventual migration of the code from larger parts down to smaller ones without rearranging the binary image, but this is not a restriction. Unused Entry Points may be used for microcode, so this organization extends the microcode continuous area to the unused area of the Entry Table. For this purpose, Function numbers should be selected from 0 up to 31. If, for example, only 8 functions are implemented, only the Entry Table locations for Functions 0 to 7 are used, and the Entry Table locations for functions 8 to 31 can be used as microinstruction memory (adding extra continuous 1536 bytes for microprogram usage).

One way of implementing different sets of Functions is having more than one Entry Table, and configuring the eTPU with the appropriate one for the application by changing ETPUECR register field ETB. Note that the Engines can use different Entry Tables, with or without the same set of Functions.

43.14.1.1.2 Entry Point Address Generation

The Entry Point address within the Entry Table is determined by the Function assigned to the Channel, the state of the Channel, the type of event, and the condition encoding scheme. Together with the Entry Table base address, they form the Entry Point Address at the SCM, as shown in the following figure.

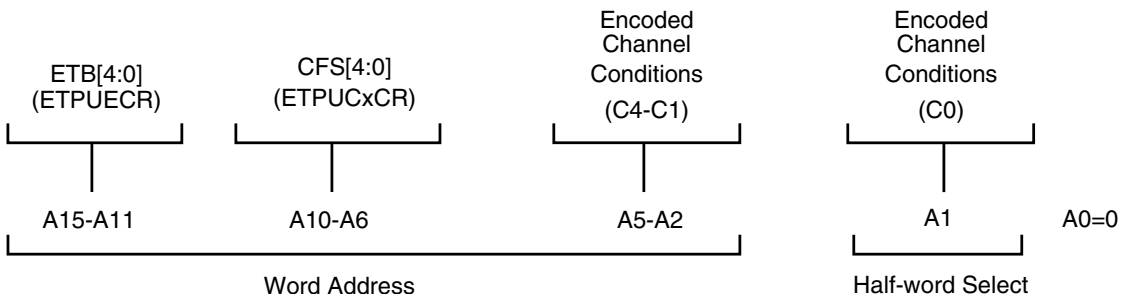


Figure 43-292. Entry Point Address (host address offset)

The type of event and channel state are coded in the Encoded Channel Conditions field C4-C0, according to one of two encoding schemes:

- Standard Entry Table Condition encoding scheme, shown in Table 43-332, which privileges Host Service Requests.
- Alternate Entry Table Condition encoding scheme, shown in Table 43-333, which focus on other events and state decoding.

The events that take part on condition encoding generate a Service Request, and have four origins:

1. Match Recognition (caused by greater/equal match, or equal-only, between the value TCR1/2 and the value stored in the channel match registers). eTPU channels support single and double match in various modes of match recognition; see Match Recognition.
2. Transition Detect Service Request (channel input signal transition detection of a selected edge). The eTPU channels support single and double transition, which together with the double match options provide various modes of transition detection; see Transition Detection and Time Base Capture.
3. Channel Linking Service Request (microcode writing the channel number to the LINK register). Link service request allows one channel to activate another (see Channel Link). Links may also come from Shared Subsystem units (see Shared Subsystems Support").

4. Host Service Request (Host writes a non-zero value to the HSR bits of the channel; see Host Service Requests).

Note

Even if a Transition or Match Service Request is inhibited (by channel mode/state or SRI), the Transition Detection and Match Recognition are taken into account for condition encoding. That is, the MRLA/B and TDLA/B flags are used, not their respective Service Requests.

Columns Host Request Bits, Link Request, MatchA/TransB, and MatchB/TransA determine the type of event. A non-zero value in these columns represents the recognition of the event, while "x" indicates that its recognition is irrelevant. Values 1 and 0 mean that event was recognized or not, respectively. Note that Match and Transition events may occur and not be recognized, and in this case it assumes value 0 for the condition encoding. The recognition of such an occurred event depends on the channel mode assigned and other conditions, as described in Enhanced Channels.

The **Host Service Request Bits** column refers to the value written by the Host CPU to the Host Service Request Register (ETPUCxHSRR) of the Channel being serviced. Note that the bits on this row are coded (3-bit representation). If the value of HSR is not zero, then the Host actually requested service.

The **Link Request** column refers to the occurrence of a Channel Link or SSS Link requests.

The **MatchA/TransB** column refers to the recognition of either a Match event specified by MatchA channel register or the detection of a channel input signal event specified by the IPACB configuration register (see Pin Control Registers).

The **MatchB/TransA** column refers to the recognition of either a Match event specified by MatchB channel register or the detection of a channel input signal event specified by the IPACA configuration register (see Pin Control Registers).

For the channel input signal, MatchA and MatchB provide double timeout conditions which depend on the channel mode programming (see Channel Modes). If the channel is used for output only, there are no transition detections, so the MatchB/TransA column represents only Match B, and MatchA/TransB column the Match A. In this case Match A and Match B are separated to give better state resolution in double match output functions. For more information about channel requests refer to Enhanced Channels.

Besides those events, the following channel state conditions help to determine the Entry Point:

1. Channel Flags 0 and 1: these are channel-internal flags (not in SDM) associated with a channel. Their values are set by microcode (see Channel Flags Operations).
2. Input Pin state or Output Flip Flop: the state (0 or 1) of the channel input signal after the Enhanced Filter (see Enhanced Digital Filter - EDF), or the state driven to the output signal. Which one (input or output) is used is selected by the ETPUCxCR bit ETPD.

The two Entry Table Condition encoding schemes combine events and state conditions differently, as detailed in the following sections.

43.14.1.1.3 Standard Condition Encoding Scheme

In this scheme, shown in the following table, all 7 HSR combinations are used and other event type columns are marked "x" when HSR is non-zero, indicating that Host Service Request has priority over any other type of event. However, when an HSR service thread is called (entry numbers 0 to 9), other events may also have been recognized, and it is microcode responsibility to check them.

When HSR is 0, i.e., Host did not issue a Service Request to the channel, the other event conditions, the input signal state and channel flags determine the Entry Point. Note that channel flag 1 does not influence the encoding in this scheme.

Table 43-332. Standard Channel Condition Encoding Scheme

No.	Encoded Channel Conditions [C4-C0]	Host Service Request Bits	Link Request	MatchA/ TransB	MatchB/ TransA	In/Output Pin State[1]	Channel Flag1	Channel Flag0
0	**001**	**00000**	**x**	**x**	**x**	**0**	**x**	**0**
1	**00001**	**001**	**x**	**x**	**x**	**0**	**x**	**1**
2	**00010**	**001**	**x**	**x**	**x**	**1**	**x**	**0**
3	**00011**	**001**	**x**	**x**	**x**	**1**	**x**	**1**
4	**00100**	**010**	**x**	**x**	**x**	**x**	**x**	**x**
5	**00101**	**011**	**x**	**x**	**x**	**x**	**x**	**x**
6	**00110**	**100**	**x**	**x**	**x**	**x**	**x**	**x**
7	**00111**	**101**	**x**	**x**	**x**	**x**	**x**	**x**
8	**01000**	**110**	**x**	**x**	**x**	**x**	**x**	**x**
9	**01001**	**111**	**x**	**x**	**x**	**x**	**x**	**x**
10	**01010**	000	**1**	1	1	**x**	**x**	0
11	**01011**	000	**1**	1	1	**x**	**x**	1
12	**01100**	**000**	**0**	**0**	1	**0**	**x**	**0**
13	**01101**	000	**0**	**0**	1	0	**x**	1
14	01110	000	**0**	**0**	1	1	**x**	0

Table continues on the next page...

Table 43-332. Standard Channel Condition Encoding Scheme (continued)

No.	Encoded Channel Conditions [C4-C0]	Host Service Request Bits	Link Request	MatchA/ TransB	MatchB/ TransA	In/Output Pin State[1]	Channel Flag1	Channel Flag0
15	01111	000	**0**	**0**	1	1	**x**	1
16	**10000**	000	**0**	1	0	**0**	**x**	**0**
17	**10001**	000	**0**	1	0	0	**x**	1
18	**10010**	000	**0**	1	0	1	**x**	0
19	**10011**	000	**0**	1	0	1	**x**	1
20	**10100**	**000**	**0**	1	1	**0**	**x**	**0**
21	**10101**	000	**0**	1	1	0	**x**	1
22	**10110**	000	**0**	1	1	1	**x**	0
23	**10111**	000	**0**	1	1	1	**x**	1
24	**11000**	000	1	0	0	**0**	**x**	**0**
25	**11001**	000	1	0	0	0	**x**	1
26	**11010**	000	1	0	0	1	**x**	0
27	**11011**	000	1	0	0	1	**x**	1
28	**11100**	000	1	0	1	**x**	**x**	**0**
29	**11101**	000	1	0	1	**x**	**x**	1
30	11110	000	1	1	**0**	**x**	**x**	0
31	11111	000	1	1	0	**x**	**x**	1
		Host Service Request						

1. The ETPUCxCR bit ETPD selects between input and output pin state.

43.14.1.1.4 Alternate Condition Encoding Scheme

This scheme is shown in the following table. Because the HSR bits cannot be tested by microcode, only three distinct Host Service Request can be used:

1. HSR=010 or 011, which are coded into the same Entry Points (0 to 3)
2. HSR=100,101 or 001, which are all coded into Entry Point 4
3. HSR=110 or 111, which are both coded into Entry Point 5

The remaining Entry Points use both channel flags for better state decoding, making this scheme better suited for Functions which need more states and/or faster state decoding, without needing many HSRs.

Table 43-333. Alternate Channel Condition Encoding Scheme

No.	Encoded Channel Conditions [C4-C0]	Host Service Request Bits	Link Request	MatchA/ TransB	MatchB/ TransA	In/Output Pin State[1]	Channel Flag1	Channel Flag0
0	**00000**	**01x**	**x**	**x**	**x**	**0**	**x**	**0**
1	**00001**	**01x**	**x**	**x**	**x**	**0**	**x**	**1**
2	**00010**	**01x**	**x**	**x**	**x**	**1**	**x**	**0**
3	**00011**	**01x**	**x**	**x**	**x**	**1**	**x**	**1**
4	**00100**	**10x/001**	**x**	**x**	**x**	**x**	**x**	**x**
5	**00101**	**11x**	**x**	**x**	**x**	**x**	**x**	**x**
6	**00110**	000	**1**	0	0	0	x	x
7	**00111**	000	**1**	0	0	1	x	x
8	**01000**	**000**	**x**	1	0	0	**0**	**0**
9	**01001**	000	**x**	1	0	0	0	1
10	**01010**	000	**x**	1	0	0	1	0
11	**01011**	000	**x**	1	0	0	1	1
12	**01100**	**000**	**x**	1	0	1	**0**	**0**
13	**01101**	000	**x**	1	0	1	0	1
14	01110	000	**x**	1	0	1	1	0
15	01111	000	**x**	1	0	1	1	1
16	**10000**	000	**x**	0	1	0	0	0
17	**10001**	000	**x**	0	1	0	0	1
18	**10010**	000	**x**	0	1	0	1	0
19	**10011**	000	**x**	0	1	0	1	1
20	**10100**	**000**	**x**	0	1	1	**0**	**0**
21	**10101**	000	**x**	0	1	1	0	1
22	**10110**	000	**x**	0	1	1	1	0
23	**10111**	000	**x**	0	1	1	1	1
24	**11000**	000	**x**	1	1	0	**0**	**0**
25	**11001**	000	**x**	1	1	0	0	1
26	**11010**	000	**x**	1	1	0	1	0
27	**11011**	000	**x**	1	1	0	1	1
28	**11100**	000	**x**	1	1	1	**0**	**0**
29	**11101**	000	**x**	1	1	1	0	1
30	11110	000	**x**	1	1	1	1	0
31	11111	000	**x**	1	1	1	1	1
		Host Service Request						

1. The ETPUCxCR bit ETPD selects between input and output pin state.

43.14.1.1.5 Entry Point Format

Entry Point information includes a Preload-Parameter selection field, a Match Enable field, and the first microcode address of the thread. The Entry Point format is illustrated in the following figure.

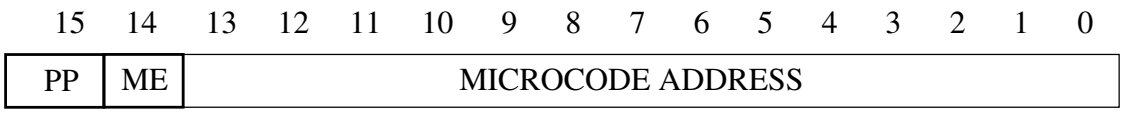


Figure 43-293. Entry Point Format

PP - Preload Parameter

PP indicates which pair of channel parameters are loaded into registers P and DIOB from the SDM prior to the execution of a thread. Preloading occurs during the time-slot transition period (see Time Slot Transition).

1 = microengine register P is preloaded from parameter 2 and DIOB from parameter 3.

0 = microengine register P is preloaded from parameter 0 and DIOB from parameter 1.

The parameter numbers are offsets from the channel parameter base address. For more info, see Parameter Access.

ME - Match Enable

ME specifies whether match event recognitions are enabled or disabled for the thread associated with the entry point during the thread execution. If they are disabled, a match recognition can only occur after channel service. For more details refer to Match Recognition.

1 = Matches are enabled during the thread.

0 = Matches are disabled during the thread.

The disabling of Match A/B recognition by MEF is dependent on IPACA/B configuration on the serviced channel (see Pin Control Registers). If IPACA=1xx, Match A is not disabled by ME=0. Likewise, IPACB=1xx overrides the effect of ME on Match B to "always on" If IPACA/B=0xx, Match A/B is disabled for two microcycles during TST (see Time Slot Transition) and is re-enabled when Entry Point is loaded, if

ME=1. Note that if the comparator is in equal-only mode and the time base reaches the value of the Match register during the time that recognition is disabled (beginning of TST, plus whole thread if ME=0), the match recognition is lost. If the comparator is in greater-equal mode, the match event may be recognized after the disabling period if it satisfies the "greater-than" condition.

MICROCODE ADDRESS - Starting Microcode Address

This field specifies the microcode address on which the thread is to begin execution.

43.14.1.2 Time Slot Transition

The Time Slot Transition period (also called TST for short) is the interval between the servicing of two channels, during which all Channel-specific context is loaded for the new serviced Channel. The primary tasks completed during this period include:

- Set MEF for the TST wait-states, if any.
- Reset MEF for the first two microcycles (after eventual wait-states).

- Update of the CHAN register with the number of the new channel to be serviced.
- Parallel update of ERTA and ERTB from CaptureA and CaptureB registers of the new serviced channel.
- Sampling of the branch conditions of the new channel to be serviced into the branch logic (this means flags TDLA/B, MRLA/B, LSR, FM[1], FM[0], and PSS). The branch conditions are coherent with the timebase capture values sampled into ERTA/B (if MRLA/B, TDLA/B are set at the same time of the sampling, either both old flag state and capture values are sampled, or both new values are sampled).
- Formation of the entry point address.
- Copy the ME bit in the Entry Point into MEF.
- Access to the entry point location and getting the first microinstruction address.
- Preload of two parameters from the SDM into P (32 bits) and DIOB (24 bits).
- Fetch the first instruction of the thread to be executed for the new channel.
- Preset the RAR register value (see RAR Register).

The preload operation is 32-bit wide for P and 24-bit wide for DIOB. The P register is loaded with all the 32-bit parameter. The DIOB register is loaded with the lower 24-bits of the parameter. The microcode can switch at any time to access the lower 24-bits, upper byte, or all the 32-bits of any parameter in the SDM. Preload of P-DIOB pair of parameters is atomic with respect to Host and CDC accesses, and so are coherent with their dual-parameter coherent transfers. For more details see Parameter Sharing and Coherency.

No instructions are executed at the Engine where the time slot transition period occurs, but the other Engine can execute normally. Match A/B is unconditionally disabled on the first two TST microcycles, if IPACA/B=0xx (respectively). During the rest of time slot transition, match recognition can be disabled or not, depending on IPACA/B field and ME. See Match Recognition .

Time Slot Transition takes a minimum of 3 microcycles (6 eTPU clocks), which may be extended due to SDM arbitration wait-states for the first preload access (see SDM Arbitration). When no wait-states are received (Figure 43-294), DIOB is preloaded twice, one for each PP value, and the correct value remains in DIOB when the Entry Point is loaded. Figure 43-295 and Figure 43-296 show the timing for one and two wait-states, respectively.

Registers B, C, D and SR are not altered by TST and keep their values from the previous thread. The values of registers A, MACL and MACH are not guaranteed at the thread start.

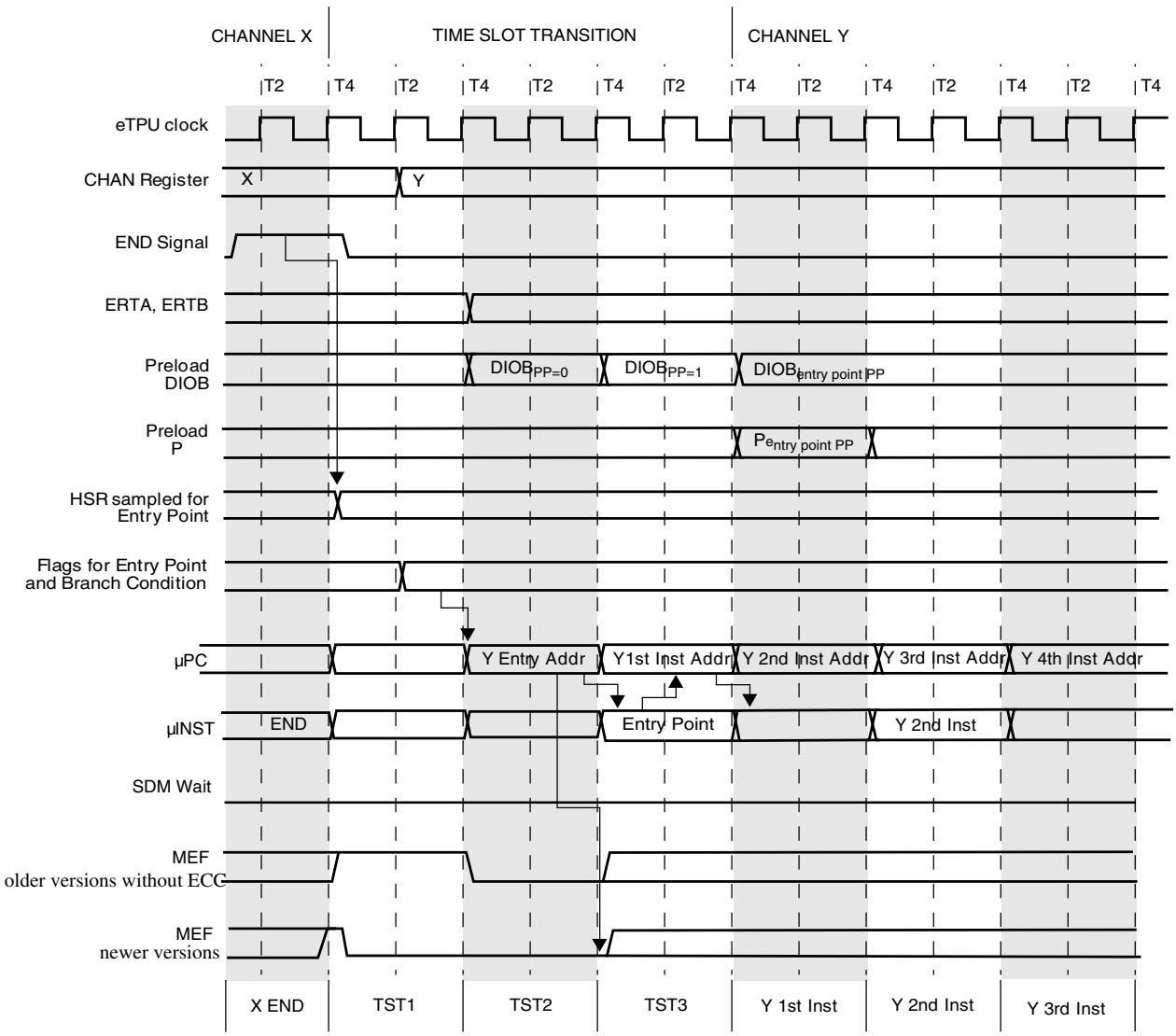


Figure 43-294. TST Timing - No Wait-states

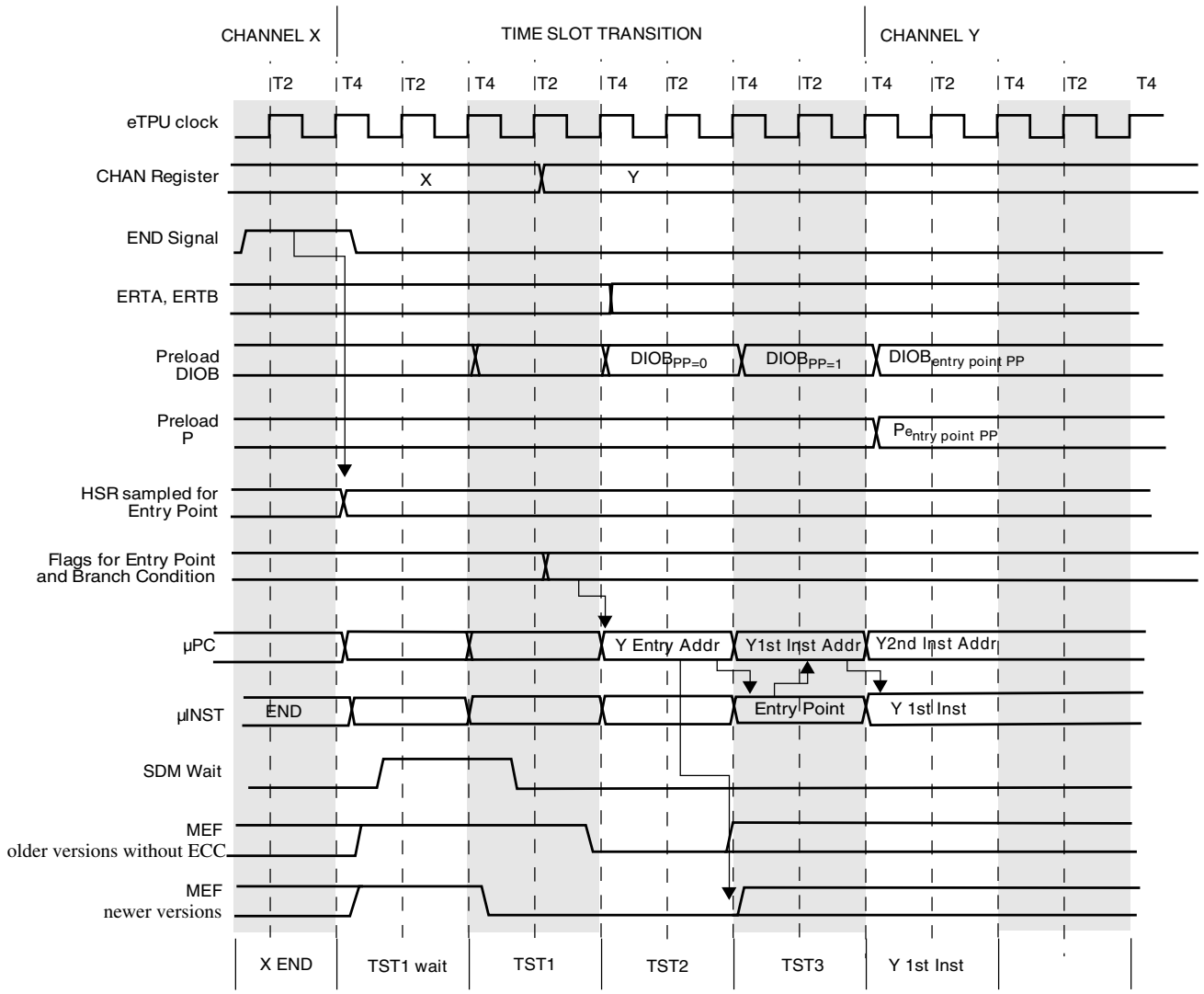


Figure 43-295. TST Timing - 1 Wait-State

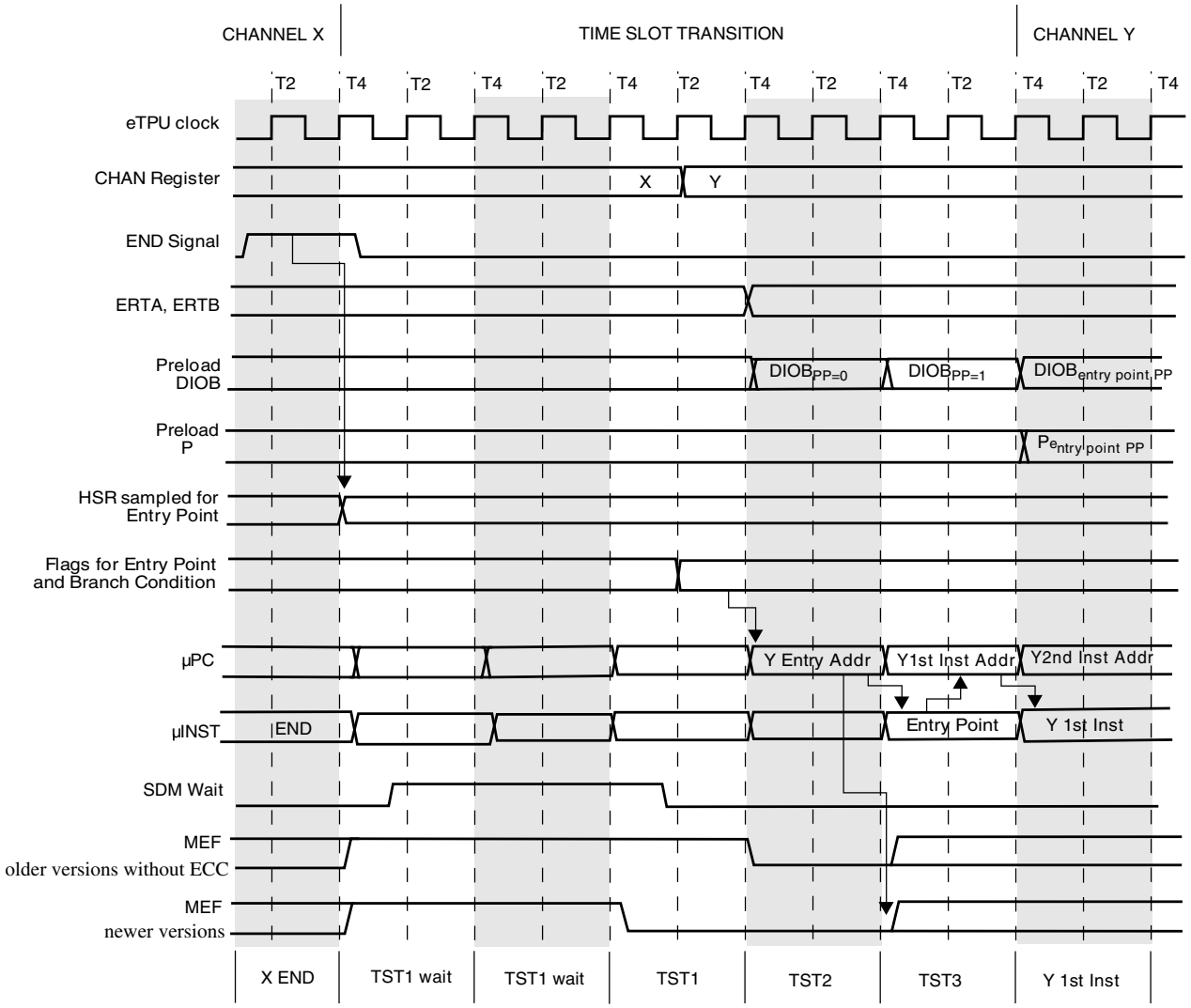


Figure 43-296. TST Timing - 2 Wait-states

For more information on Channel-specific registers and flags, refer to Enhanced Channels. For more information on P, ERTA/B and DIOB registers refer to Registers.

43.14.1.2.1 Thread Gap

The MEF timing has changed from previous eTPU/eTPU2 versions, now being reset for two microcycles from the start of TST. Therefore, two consecutive threads with ME=0 serving the same channel could block any match until the end of the second thread (see Figure 43-294). This could cause latency problems and also break compatibility with legacy code, so the thread transition logic was modified to introduce a one-microcycle gap between consecutive threads servicing the same channel, as shown in the following figure. During the gap MEF is set to one, regardless of the previous or the next entry point ME field. The thread gap is inserted only between threads serving the same channel.

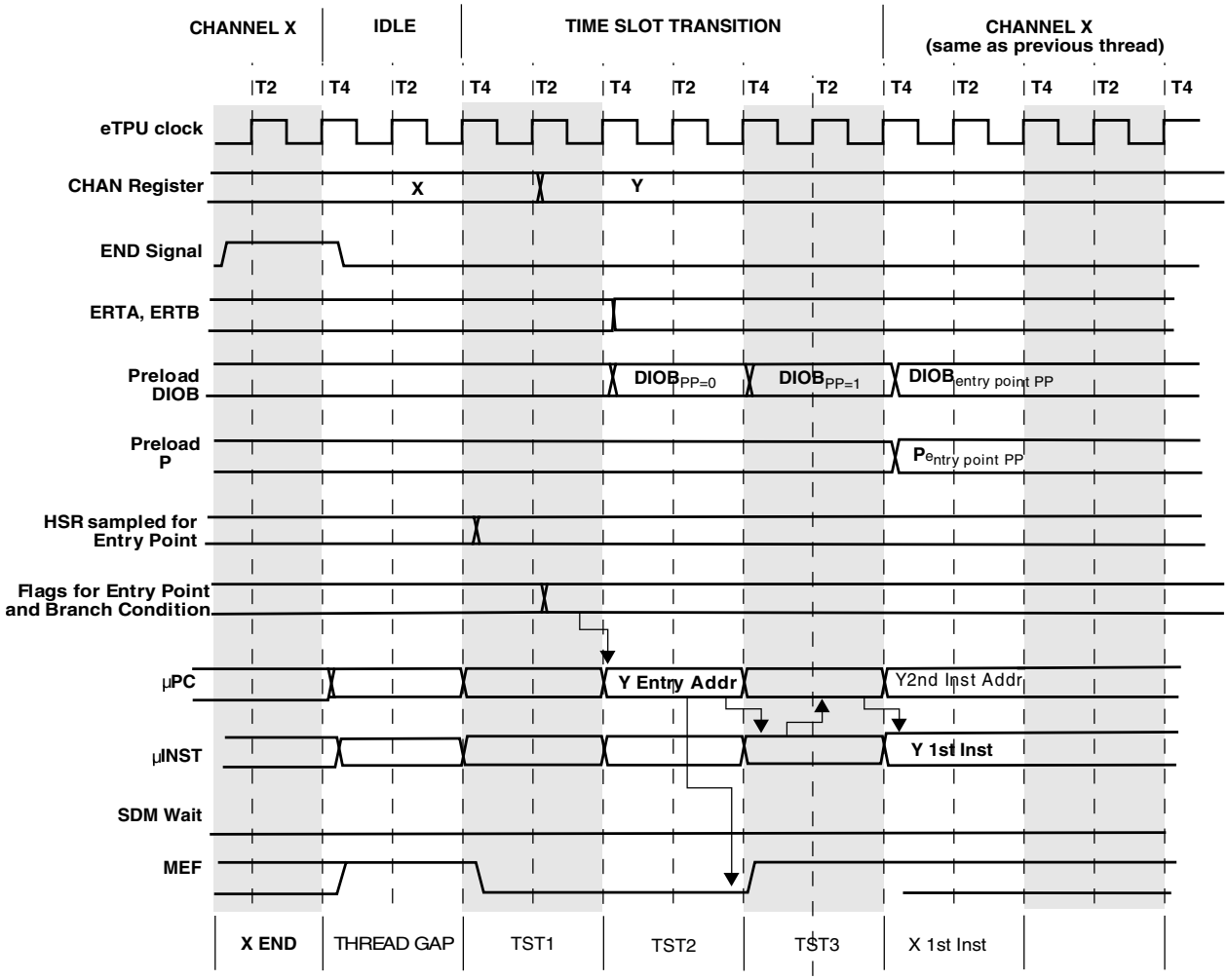


Figure 43-297. Thread Gap

43.14.1.3 Thread Ending

Threads can finish by either:

- an instruction with the END field active (see Ending Current Thread - END).
- a forced END by host writing to the ETPUECR bit FEND (see ETPUECR register).
- a forced END caused by Watchdog timeout (see Watchdog).
- a forced END caused by a non-correctable memory error (see Memory Error Support).

43.14.1.4 Watchdog

Each engine has a watchdog mechanism to prevent a thread or a sequence of threads from running too long, impacting the latency of the other channel services. The watchdog is configured through the register ETPUWDTR (see ETPUWDTR register). When the watchdog is enabled, an internal counter increments on each microcycle when a thread is executing. If the count is greater than the value specified in the ETPUWDTR field WDCNT and a thread is still executing, the watchdog:

1. if ETPUWDTR bit WMO is negated, forces an END of the thread, allowing other channels to be serviced
2. sets the WDS status bit of the serviced channel in the ETPUWDSR register (see ETPUWDSR register).
3. if ETPUWDTR bit WMO is negated, the channel is disabled, not initiating any thread until its WDS bit is cleared (its CPR field in ETPUCxCR is not changed, however).
4. issues a Global Exception and sets the ETPUMCR bit WDTO1 (engine 1) or WDTO2 (engine 2) (see ETPUMCR register, regardless of the WMO bit state.

If ETPUWDTR bit WMO is asserted, the watchdog can be used to monitor the execution time threshold of a thread or thread sequence without interfering with the eTPU code execution.

The watchdog can be configured in one of the following modes, defining how the internal watchdog count is reset:

- **Thread Length Mode**: the watchdog count is reset at the end of each thread.
- **Busy Length Mode**: the watchdog count is reset when the microengine goes idle. A sequence of threads, one right after another, keeps the count running. The counter is also reinitialized when a thread is forced to end, so that a new count begins if another TST initiates at the following microcycle.

The following applies to the watchdog mechanism:

- microcycles during TST and SDM access wait-states (on TST or instruction execution) are counted.
- if the watchdog count equals WDCNT in the last microinstruction (with SDM wait-states or not) of a thread servicing a channel, its WDS bit is not set.

- if the watchdog count expires (gets greater than WDCNT) during the TST, the thread is forced end on its first instruction.
- the watchdog count does not wrap, so that a thread (in thread length mode) or a thread sequence (in busy length mode) that lasts for more than the maximum value of WDCNT does get a forced end.

Note

Watchdog must not be enabled when the microengine enters halt mode.

The counter does not run when the engine is stopped, and resets when the watchdog is disabled.

43.14.2 Host Interface

43.14.2.1 System Configuration

System Configuration Registers are described in the System Configuration Registers section. Detailed explanation of the configured functionalities is found throughout Functional Description and a specification for the initial configuration sequence is found in Configuration Sequence.

43.14.2.2 Interrupts and Data Transfer Requests

43.14.2.2.1 Interrupt Types and Sources

Each one of the eTPU channels can be a source of two requests: **Channel Interrupt** request and **Data Transfer Request**.Channel Interrupts are targeted to a Host CPU. Data Transfer Requests may be targeted to a data transfer module (e.g., a DMA controller). Interrupt and Data Transfer registers are used by the Host to enable interrupts and data transfer requests, indicate their status and service them. Interrupt and Data Transfer requests have the same sets of registers and external signals, and are handled in the same way. They differ only by the fact that Data Transfer Requests are also cleared by the assertion of respective DMA completion acknowledge line .

Note

Interrupt and Data Transfer requests can be cleared even when Engines are in Module Disable Mode, through the Global Channel Registers, and also DMA completion for Data Transfer requests.

Channel Interrupts and Data Transfer Requests can only be issued by eTPU microcode, through one of the Channel Control instruction fields (see Channel Interrupt and Data Transfer Requests).

Both Channel Interrupt and Data Transfer requests can be individually enabled for each channel.

eTPU Interrupt and Data Transfer Registers are mirrored in two organizations: grouped by Channel and grouped by type (interrupt status, interrupt enable, data transfer status, data transfer enable). This allows either "channel-oriented" or "bundled channel" Host interrupt service schemes, or a combination of them. For a detailed description, refer to the Channel Register Layout and Global Channel Registers sections.

eTPU can also assert a **Global Exception** interrupt indicating a global illegal state. There are six possible sources for a Global Exception:

- Execution of an illegal instruction by the microengine (see Illegal Instructions). This Global Exception source is flagged by the bits ILF1 and ILF2 in register ETPUMCR.
- An SCM signature mismatch detected by the Multiple Input Signature Calculator - MISC. See SCM Test - Multiple Input Signature Calculator. This source is flagged by the bit SCMMISF in register ETPUMCR.
- Microcode request, through microinstruction field CIRC (see Channel Interrupt and Data Transfer Requests). This Global Exception source is flagged by bits MGE1(Engine 1) and MGE2(Engine 2) in register ETPUMCR. The cause of this illegal state is application-dependent. The microcode may write an error code into the SDM to indicate the cause of the exception, for instance.
- A Watchdog timeout, flagged by the ETPUMCR bits WDTO1 and WDTO2 (see Watchdog).
- An SDM or SCM non-correctable error on a microengine or host access, flagged by the ETPUMCR bits SDMERR and SCMERR, respectively (see Memory Error Support).
- The latency measured by any of either engine's Latency Monitors is greater than the specified maximum, flagged by ETPUMCR bits LTO1 (Engine 1) and LTO2 (Engine 2) (see Latency Monitor).

Global Exceptions cannot be directly disabled within eTPU, except by disabling its sources (Memory Error, SSS, MISC and microcode), and it is cleared by writing 1 to the GEC bit in ETPUMCR. Clearing Global Exception clears all Global Exception source status bits (ILF1, ILF2, SCMMISF, MGE1, MGE2, SCMERR, SDMERR). If GEC is written 1 at the same time any of the sources issues a Global Exception, both the interrupt and the status bit of that source remains asserted. The assertion of Global Exception by one of the sources above does not prevent the others from asserting it too, so any number of them, in any combination, can be flagged.

Note

There can be a race between the clear of a Global Exception and occurrence of a new set condition, such that the set happens just before the clear and cannot be sensed by the Host. Therefore, Global Exception cannot be used as a normal interrupt source. It should only be used for emergency procedures.

Memory Error Correction support (see Memory Error Support) has a distinct interrupt source, which can activate on any correctable error occurrence, either to SCM or SDM accesses, but does not activate on non-correctable errors (which assert Global Exception instead). Memory Error Correction interrupt assert when it is enabled through the ETPUMECR bit MECIE (see the ETPUMESR register) and either ETPUMESR bit DCERR or bit CCERR is asserted. The interrupt is negated when both DCERR and CCERR are negated, or MECIE is negated.

43.14.2.2.2 Interrupt and Data Transfer Request Overflow

If a Channel Interrupt was issued, its status bit is still set, and microcode issues another Channel Interrupt, the Interrupt Overflow status bit is set for that channel. Interrupt Overflow status can be checked by the Host in Channel Status register ETPUCxSCR bit CIOS (see the ETPUCnSCR register), mirrored in the ETPUCIOSR register. Interrupt Overflow status is not cleared automatically when Interrupt Status is cleared. The same mechanism and respective registers (ETPUCDTROSR) are available for Data Transfer Requests.

If interrupt is set and cleared at the same time, set prevails and overflow is not altered (keeps the same state as it was before, asserted or not).

Global Exception has no overflow status.

43.14.2.3 Parameter Access

43.14.2.3.1 Parameter Access Widths

From the Host side the SDM address space is mapped in bytes, and each 32-bit parameter occupies 4 contiguous, aligned bytes. The Host can read/write the SDM by 8-, 16-, or 32-bit accesses in aligned addresses.

In 32-bit access, Host can access all 32 bits or only the lower 24 bits with an automatic sign extension (see Parameter Sign Extension Area).

43.14.2.3.2 Parameter Addresses and Endianness

To access parameter number xxx, eTPU Microengine(s) would select address xxx. The Host would add *(xxx*4)* to the SDM base address to access the same parameter. For example, parameter 0x101 is seen by the Host in *(SDM base address +0x404)* . An example of SDM memory map is shown in Figure 43-298 . The Host can access the SDM with a 32-bit-wide bus cycle to a four-byte aligned address, 16-bit-wide bus cycle to a two-byte aligned address, or 8-bit wide bus cycle to any byte address.

The address of the 24-bit parameters and the most significant byte depends on the endianness of the MCU. For more details, see the Endianness.

43.14.2.3.3 Parameter Concurrency

Host accesses to parameters may occur in parallel with eTPU Microengine accesses. Readings taken from a group of parameters while they are being simultaneously updated may lack coherency. eTPU provides mechanisms to ensure parameter coherency in accesses from both Host side and Microengine side, including the use of a coherent dual-parameter transfer mechanism, described in detail on Parameter Sharing and Coherency.

43.14.2.3.4 Parameter Sign Extension Area

The SDM address space to the Host is mirrored in a Parameter Sign Extension - PSE - area (see Registers Memory Map). Accesses from the Host to the PSE area differ from accesses to the standard SDM address space as follows:

- **Writes**: the most significant byte of the parameters is not written, and the SDM retains the old byte value, regardless of the Host access size.
- **Reads**: the most significant bit of the 24-bit parameter (that is, the MSB of the second most significant 32-bit parameter byte) is repeated in the 8 most significant bits of the read value on all 32-bit reads and most significant 16- and 8-bit reads.

The same parameters written in the standard SDM address space are read from the PSE area with the same offsets, and vice-versa. See Table 43-426 for a reference of the address offsets in big and little endian machines.

This feature reliefs the Host from extending the signal of 24-bit eTPU parameters before calculations, and from read-modify-write accesses to modify 24-bit parameters at the SDM.

43.14.2.4 SDM Organization

The SDM internal partition for channel allocation is dynamic and programmed in the Channel Registers (see the ETPUCnCR register).

The Host application is responsible for allocating a different parameter base address to each channel during the initial eTPU configuration, and to allocate enough parameters for the selected Function, with no unintentional overlapping between parameters of different functions.

Besides channel parameters, global areas may have to be allocated for parameters that are shared by more than one channel, in one or both Engines. Also, temporary parameter areas should be reserved to be used by the coherent parameter transfer mechanisms described in Parameter Sharing and Coherency, if necessary. An SDM organization example is shown in the following figure.

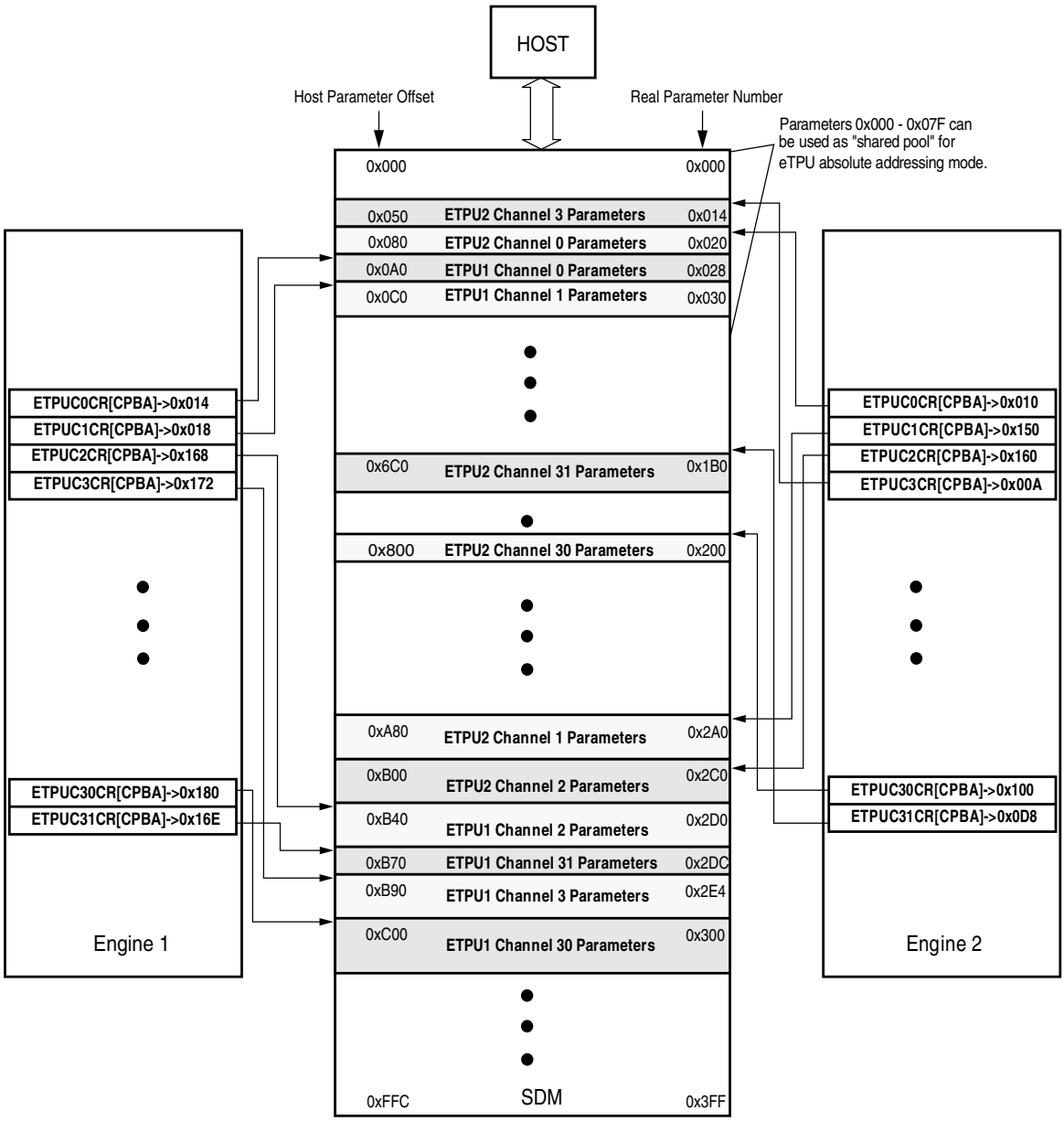


Figure 43-298. SDM Organization Example (4 KB)

Single-engine eTPU or dual eTPU system may require less parameters than the maximum number provided by the SDM. Since the SDM partition is fully dynamic, there is no limitation of fixed channel addresses, and the reduced array can be fully utilized.

43.14.2.5 Host Service Requests

Host CPU can request immediate service from a channel by writing a non-zero value to the Host Service Request register field - HSR (see the ETPUCnHSRR register). There is one HSR field for each channel, so that writing to it generates a Service Request to the respective channel only. A zero value in HSR means no Host Service Request is pending for the channel.

HSR value turns to 000 automatically at the end of microengine service for that channel, but only if the thread started due to an HSR.

The meaning of a non-zero HSR value depends on the Function assigned for the channel. These bits are part of the conditions which select the Function entry point, and cannot be tested by microcode. For more details, refer to Entry Points.

If Host writes HSR=000 when a thread for the same channel is already running, the thread runs until the end and is not aborted. If Host writes HSR>000 when an HSR thread for the same channel is already running, HSR value resets at the end of the thread, and no new HSR will be pending. If HSR is written before its value is resolved by the scheduler during TST, the entry point will obey the new HSR value, and if this new value is 000, no service thread is executed for the HSR.

The scheduling of HSRs is completely asynchronous with Host accesses, and there is no race-free manner to change an HSR value before service thread execution, so generally the safe way is: write HSR>0 only when HSR=0. Error recovery or emergency host procedures may require one to the safely abort service and reset channel state when an HSR is already pending or executing. In these cases, the procedure below should be followed:

1. Disable the channel, writing CPR=00 in register ETPUCxCR. That will prevent any pending HSR to be serviced.
2. Check if the channel is currently being serviced, reading its service status bit in register ETPUCSSR. If it is, wait for the time necessary to finish the service pending, or check again until HSR == 0, or channel service bit in ETPUCSSR is cleared.
3. Write HSR with the error recover value. This value should, possibly combined with other host-defined flags in SDM or FM bits, initiate a channel reset or error recovery procedure.
4. Re-enable the channel, writing CPR value > 0 in register ETPUCxCR.

43.14.2.6 SCM access

Only Host can access SCM as data. Depending on the specific device, SCM may be implemented as a RAM or ROM. This determines Host accesses to the SCM as shown below.

43.14.2.6.1 SCM RAM Implementations

When SCM is implemented as RAM, the Host may read or write to SCM by setting ETPUMCR bit VIS=1. If VIS=0 and Host tries to access SCM space, a bus error is issued, writes are ineffective and read data is meaningless. Both Engines must be stopped or halted to set VIS=1.

Note

It is necessary to turn VIS bit on to set software breakpoints (see Software Breakpoints).

43.14.2.6.2 SCM Low Power

SCM turns off its internal clocks when both Engines are stopped (ETPUECR bit STF asserted), VIS=0 at ETPUMCR, and MISC is not enabled

(SCMMISEN=0). The SCM clocks are automatically turned on if either one of the STF bits is negated or VIS turns to 1, or SCMMISEN turns to 1.

SCM clocks are not turned off if any of the Engines is not stopped, even if they are both halted.

The conditions for SCM Clocks and MISC activation are summarized in the following table.

Table 43-334. SCM Clocks and MISC activation

ETPUECR_1 STF	ETPUECR_2 STF	ETPUMCR VIS	ETPUMCR SCMMISEN	SCM Clocks	MISC
0	x	0[1, 1]	1	On	On
0	x	0[1]	0	On	off
x	0	0[1]	1	On	On
x	0	0[1]	0	On	off
1	1	0	0	off	off
1	1	0	1	On	On
1[2, 2]	1[2]	1	0	On	off
1[2]	1[2]	1	1	On	off[3]
0	0	x	0	On	off
0	0	x	1	On	On

1. VIS cannot be written 1 if ETPUECR_1 bit STF=0 or ETPUECR_2 bit STF=0, and both HLTF bits are 0.
2. If VIS=1, neither MDIS can be written 0 nor the Engine leaves Stop Mode, regardless of device stop request .
3. MISC resets and stays so when VIS=1, restarting automatically when VIS goes 0 if SCMMISEN=1.

43.14.2.6.3 SCM Off-range Data

When read accesses are made, either by the Host or by a microengine, to addresses above the limit corresponding to the SCMSIZE value in ETPUMCR, the value read comes from the register ETPUSCMOFFDATAR. The Host can program the register at initialization with an opcode value with operations that try to protect or recover the system from runaway code, for instance: terminate the thread, clear channel flags, disable match and transition service requests, issue an interrupt, jump to an error recovery procedure[11] . Writes to unimplemented addresses do not return error and can write on unspecified mirror addresses, so they should be avoided.

43.14.2.7 Bus Error Conditions

Follows a summary of the possible causes of bus errors on host accesses to the eTPU memory space:

- read or write access to the SCM with ETPUMCR bit VIS=0.
- read or write access to the channel registers with ETPUECR bits MDIS=1, or bit STF=1.
- write access to the Time Base Registers with ETPUECR bits MDIS=1, or bit STF=1.
- non-correctable errors in the SCM on reads of any size or writes less than 32-bits wide when ETPUMECR bit CEDD=0.
- non-correctable errors in the SDM on reads when ETPUMECR bit DEDD=0.
- accesses to the Shared Subsystem Area, when propagating a bus error issued by the attached subsystem.

11. Only part of these suggested operations can be parallelized in a single instruction, see Microinstruction formats.

43.14.3 Scheduler

Every Function is composed of one or more Threads. A Thread consists of a group of instructions that, once begins execution, cannot be interrupted by host or channel events. Each active channel intents to be serviced, being granted time for Thread execution. Since one microengine handles several channels operating concurrently, the Function threads must be executed serially.

The task of the Scheduler is to recognize and prioritize the channels needing service and to grant execution time to each channel. The time given to an individual Thread for execution or service is called a **Time Slot** . The duration of a time slot is determined by the number of instructions executed in the Thread plus SDM wait-states received, and varies in length.

At any time, an arbitrary number of channels can require service. To request service, channel logic, eTPU microcode or Host application notifies the Scheduler by issuing a Service Request.

43.14.3.1 Channel Enabling and Priority Assignment

Every channel is assigned one of three priority levels - high, middle, or low - by the Host CPU, through the Channel Configuration Register field CPR (see the ETPUCnCR register). These registers are also used to disable the channel, which is equivalent to assigning it a "null" priority. In this case, the Scheduler does not grant any of its Service Requests.

It is possible to change the channel priority level or disable it dynamically. If the Host disables a channel when it is currently being serviced, channel service thread will complete. This means that it is possible for the output level of a channel signal to change, or a Host interrupt occur, even after its priority register was written to "null". For instance, if an output transition is scheduled, the transition will occur even after the channel is disabled.

Service requests previously pending or that occur while a channel is disabled remain asserted while the channel is disabled, and are serviced if the channel is enabled again, in due time determined by the priority scheme and concurrent requests from other channels. Channels are disabled after reset, and it is recommended to configure a Host Service Request for initialization of a channel before that channel is enabled to active priority (see Initialization/Application Information).

43.14.3.2 Channel Priority Schemes

The Scheduler holds a Service Grant register with one bit for each channel. Once the Scheduler grants a time slot to channel, the Service Grant bit for that channel is asserted in the Service Grant register. When the Service Grant bit of a channel is set, the channel may request new service but is not serviced again before its Service Grant bit is cleared.

When all channels in a same priority level are serviced, their Service Grant bits are cleared at the end of the thread, one eTPU clock before the next serviced channel is calculated, according to the scheme below[12] :

- Clear all grant bits of priority High if all channels of that priority that are requesting have their grant bits in 1.
- Clear all grant bits of priority Medium if all channels of that priority that are requesting have their grant bits in 1.
- Clear all grant bits of priority Low if all channels of that priority that are requesting have their grant bits in 1.
- Clear all grant bits of disabled channels.

This scheme assures that no channel is left with its grant bit forever asserted (preventing it from being serviced again), even if the channel priorities are reassigned during the execution.

Priority level is determined based on the maximum latency desired for each channel. A channel having a Function that requires the most frequent or more immediate service should be allocated a high priority level.

The eTPU employs a **primary** and a **secondary priority scheme**. These two schemes ensure frequent servicing of high-demand Functions and ensure a minimum time allocation to all channels requesting service, regardless of their priority level. The primary scheme prioritizes requesting channels that have different priority levels; the secondary scheme prioritizes requesting channels that have the same priority level.

Initially, a channel requests service and is granted a time slot by the Scheduler: Service Grant bit is asserted. If only high-level channels constantly receive service first because of their priority level, middle- and low-level channels would only be serviced by default, i.e., if no high-level channels request service. To ensure that each priority level receives an opportunity for servicing, every time slot has a fixed priority level that the Scheduler honors first. Divided into sets of seven, time slots are numbered from one to seven. The figure below illustrates the numbered time slots in sets of seven (fields A and B) and

12. grant bits are also cleared in the next clock, when the service channel is chosen, or when the microengine is idle, using the same scheme.

identifies their assigned default priority level. The high level has more time slots than the middle and low levels. Out of every seven time slots available, four are assigned to honor high-level channels first, two are assigned to honor middle-level channels first, and one is assigned to honor low-level channels first. Only one request (in each Engine) is serviced per time slot. When no channel requests service and the microengine is idle the priority scheme is initialized to time slot one, to prevent priority inversion on the next request[13].

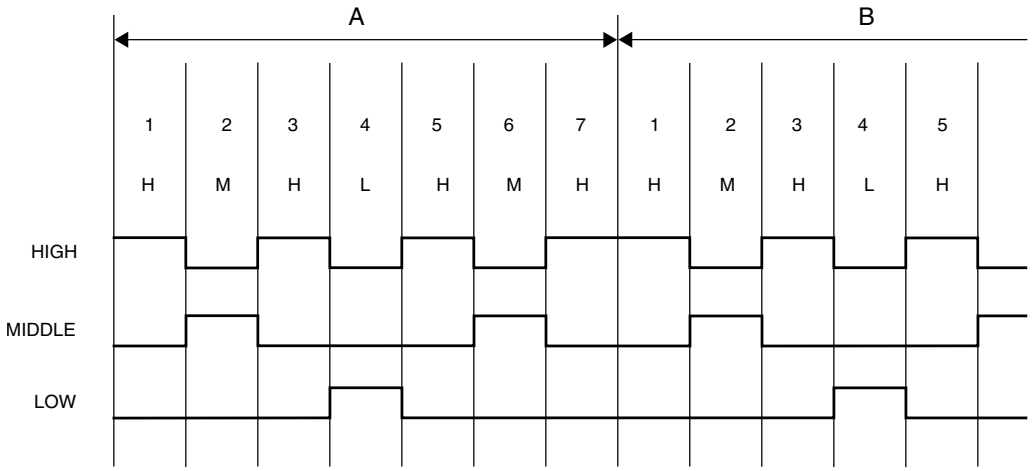


Figure 43-299. Time Slot Priority levels

43.14.3.2.1 Primary Scheme - Priority Among Channels on Different Levels

Although time slot priority assignment is fixed, the servicing priority is not. The primary scheme acknowledges the priority level assigned to a time slot, granting service first to a channel having the same priority. In Channel Priority Schemes, time slot one has a high-level assignment; therefore, a high-level channel requesting service is recognized first. However, if no high-level channel requests service, the Scheduler recognizes a requesting middle-level channel. If this level has no request, the Scheduler continues to the low-level. If no requests occur, the Scheduler truncates the seven state cycle and starts a new cycle at time slot one, waiting for the first request. Granting service to a different-level channel is called priority passing. The order of passing always gives the highest priority to the assigned level, and the second priority to the higher of the remaining requesting priority levels as shown in the following table.

13. Priority inversion would occur in the following situation: no channel is requesting service, and the current time slot is primarily assigned to a low-priority channel. If the Scheduler was not reset to time slot one and two channels requested service at the same time, one with high priority and the other with low priority, the channel to be serviced would be the low-priority channel.

Table 43-335. Priority Passing

Assigned Priority Level	Next Priority Level	Next Priority Level
High	Middle	Low
Middle	High	Low
Low	High	Middle

When priority is passed to another level, that level is serviced and the fixed-priority-level sequence is resumed with the next time slot.

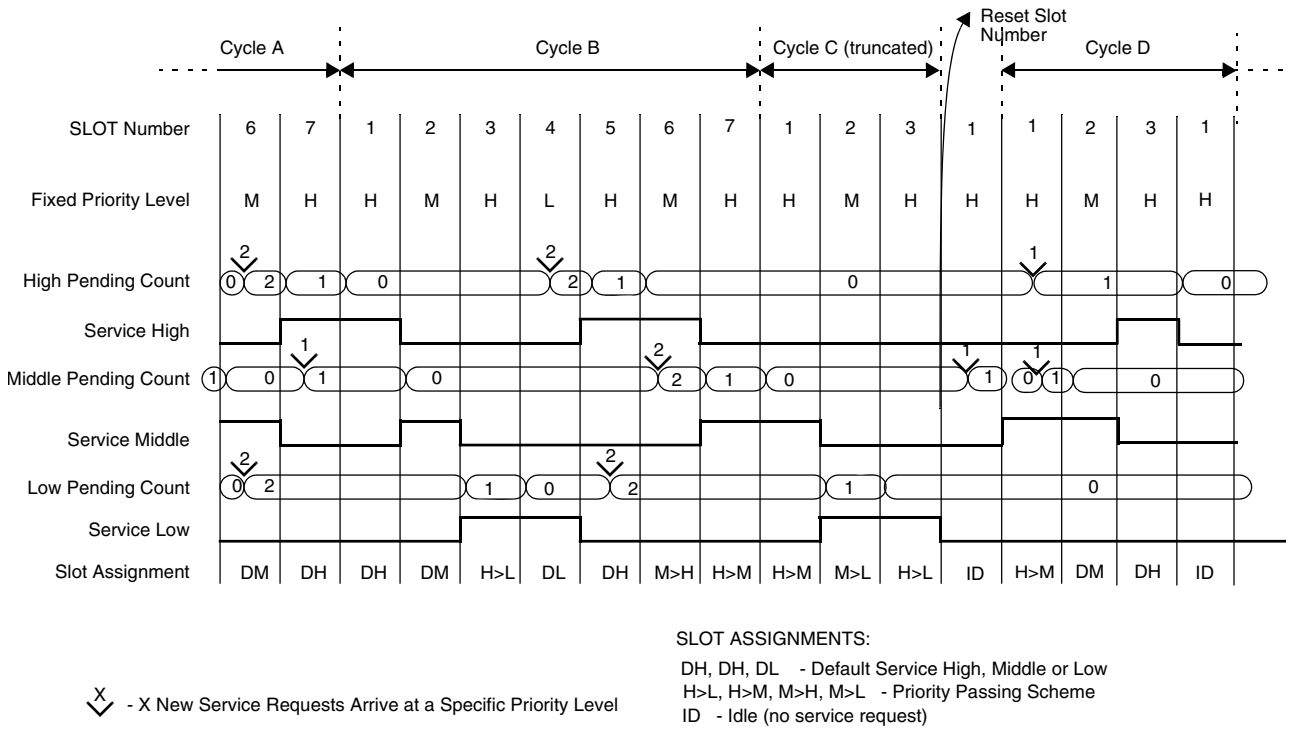


Figure 43-300. Priority Passing Example

Examples of priority passing are shown in the above figure. Each cycle contains seven time slots (or less if no service request exist). In cycle B, no high-level or middle-level service requests are present before time slot three which is assigned by default to high-level priority. Thus, time slot three is passed to the low level. In cycle B there are also no middle-level service requests before time slot six, so it passes the priority to a requesting high-level channel. During time slot six no more high level requests are left, but two new middle-level requests arrive, and there are also three low level pending service requests. Thus, time slot seven of cycle B and time slot one of cycle C are passed to the middle-level which is the next priority level after high. Time slots two and three of cycle C are passed to the low level which contains the three remaining channel service requests. At time slot three of cycle C the last low level request is serviced, and the Scheduler passes to idle state. At this point the cycle C is truncated and the Scheduler passes to time slot one of cycle D.

43.14.3.2.2 Priority Passing Disabling

The priority passing scheme allows a case where a high priority channel looses to a lower priority one right after another lower priority has been serviced, exemplified in the Cycle D on Figure 43-300. A middle priority channel wins time slot 1 due to priority passing from high to middle. While it is being serviced, two new service requests arrive, one high and one middle priority. The high priority request looses to the middle one on next time slot 2 by default priority assignment.

This priority inversion can be avoided by setting the ETPUECR register bit SPPDIS (see the ETPUECR register), which disables the priority passing mechanism. When priority passing is disabled, at the end of the thread the slot number is incremented until a time slot that matches the priority of one of the requesting channel(s). The time slot advance takes no extra clocks. If no channel requests service, the time slot counter stays at time slot 1. The priority selection scheme with disabled priority passing is summarized in the following table.

Table 43-336. Priority Passing Disabling

At the endof time slot	servici ng priority	if any request of priority	service it on time slot	else if any request of priority	service it on time slot	else if any request of priority	service it on time slot
1	High	Medium	2	High	3	Low	4
2	Medium	High	3	Low	4	Medium	6
3	High	Low	4	High	5	Medium	6
4	Low	High	5	Medium	6	Low	4
5	High	Medium	6	High	7	Low	4
6	Medium	High	7	Medium	2	Low	4
7	High	High	1	Medium	2	Low	4

An example of the priority passing disabling scheme is illustrated in the figure below. The sequence of service requests is the same as in the example of Figure 43-300, and although the time slot incrementing differs, the priorities granted are the same for cycle B. Cycle C has one of the low priority channels serviced before the second middle one. Cycle D, however, no longer has the priority inversion.

In cycle B, after the time slot 2 only a low priority request remains, so the time slot count advances directly to 4, which has a low priority assigned. Time slot keeps on 4 for the next service, as only a low priority request remains also, and only time slot 4 is assigned to low. Two high priority services contend for the next time slot 5 (assigned to High). The second high priority channel is serviced on the next time slot, jumped to 7 because there is no middle request, ending cycle B. Cycle C starts with time slot 2, as there are no high priority requests and two middle and two low ones. After the first middle service,

time slot count skips 3 assigned to high (no high requests), and services a low priority channel on time slot 4. It follows the same scheme until there are no other requests and cycle C is truncated, resetting the time slot counter to 1.

Cycle D begins with a middle request, jumping to time slot 2. During this service two requests arrive, one high and one middle. Unlike what happened with priority passing, the next serviced is the high priority channel, as the time slot increments to 3. The second middle priority channel request in cycle D is finally serviced next, on time slot 6.

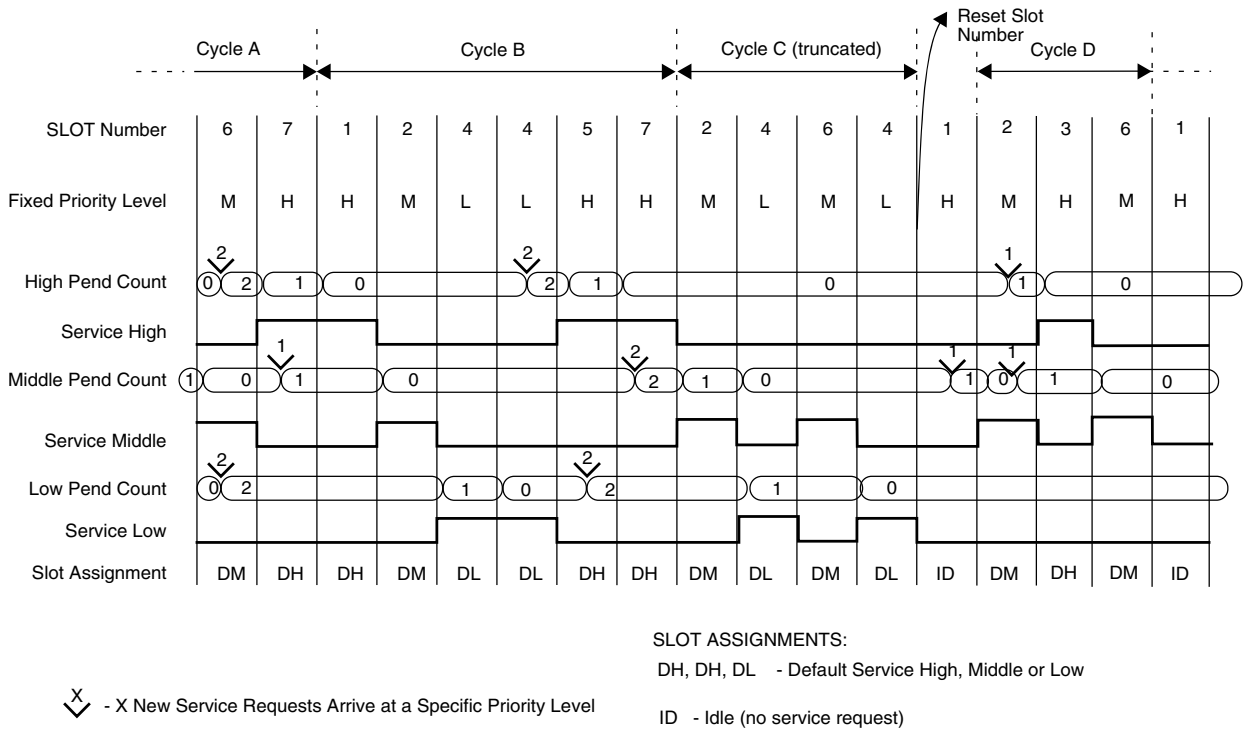


Figure 43-301. Priority Passing Disabling Example

43.14.3.2.3 Secondary Scheme - Priority Among Channels on the Same Level

Because channels can randomly request service, channels having the same priority level will inevitably request service simultaneously. A secondary scheme prioritizes these requests. The Scheduler services channels on each of the three priority levels, beginning with the lowest numbered channel on that level.

43.14.3.2.4 Priority Scheme Example

The overall priority scheme simultaneously incorporates both primary and secondary schemes. Combining both schemes in the following example conveys their correlation.

1. One high-priority and one low priority channels request service, while the Scheduler is in time slot one. Having its service request bit asserted, a single high-level channel is granted the time slot, which has high-level priority (primary scheme) and its service grant bit is asserted. At the end of the thread, the service grant bit is negated (no more requests of high priority level channels).
2. The Scheduler proceeds to time slot two, which has middle-level priority; however, no middle-level channel is requesting service. Priority is passed to the high level, but no high-level channel is requesting service; therefore, priority is passed again, and service is granted to the single requesting low-level channel. Once serviced, this channel's grant bit is negated (no more low-level requests).
3. The Scheduler resumes with the fixed-priority sequence on time slot three; however, no channels are requesting service. The Scheduler returns to time slot one, waiting for requests.
4. Two high-level and two middle-level channels simultaneously request service. Being in time slot one which is assigned high priority, the Scheduler finds the lowest numbered high-level channel (secondary scheme) and selects it for service. This channel's service grant bit is asserted.
5. The Scheduler continues to time slot two, which has middle priority (primary scheme), and allocates the slot to the lowest numbered middle-level channel requesting service (secondary scheme). The Scheduler notes the still unserviced middle-level channel and proceeds to time slot three.
6. Time slot three is allocated for high priority. The slot is allocated to the remaining unserviced high-priority channel, and the channel's service grant bit is asserted. The Scheduler checks again at the end of the thread. All service grant bits of high-level requested channels are asserted; therefore, all high-priority channels that requested have been allocated execution time. Under this condition, all service grant bits of the high-level serviced channels are negated. The Scheduler proceeds to time slot four.
7. Time slot four is allocated for low-priority channel; however, no low-level channel is requesting service. Priority is passed to the high level, but no high-level channel is requesting service; therefore, priority is passed again, and service is granted to the remaining middle-level channel which requests service. This channel's service grant bit is asserted. The Scheduler checks again at the end of the thread. All grant bits of middle-level requested channels are asserted; therefore, all middle-priority channels have been allocated execution time. Under this condition, all service grant bits of the middle-level serviced channels are negated. The Scheduler proceeds to time slot five. Meanwhile a low priority channel requests service.

8. Time slot five is allocated for high-priority channels, but there are no more requests from high-priority or middle priority channels. The single low-level channel which required service is granted time slot five. Once serviced, the channel's service grant bit is asserted. Next, the service grant bit is negated (no more requests of low priority level channels).
9. The Scheduler resumes with the fixed-priority sequence on time slot six; however, no channels are requesting service. The Scheduler returns to time slot one and waits for requests.

43.14.3.3 Time Slot Latency

Latency is the amount of time between a service request and the beginning of service on that channel. The following factors affect latency:

- Number of active channels
- Number of channels on a priority level
- Number of available time slots on a priority level
- Number of microcycles required to execute a thread of a Function
- Number of SDM accesses during execution of a Function thread
- eTPU clock frequency.

Each time slot may require a different number of microcycles, depending on the thread of a Function to be executed. This variation is shown in the figure below.

For more details on latency evaluation, see Estimating Worst Case Latency.

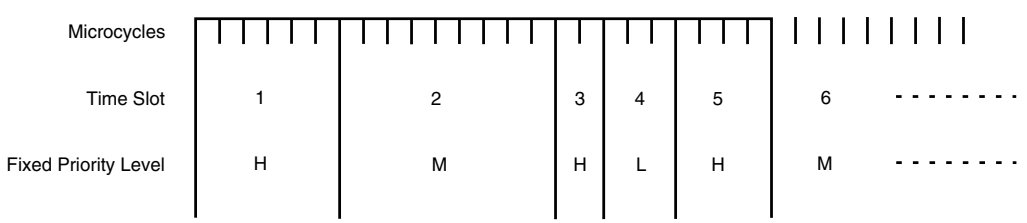


Figure 43-302. Time-Slot Variation

43.14.4 Parameter Sharing and Coherency

SDM can be concurrently accessed by Host and Microengines (two in a Dual eTPU Engine system). In general, there is no guaranteed order by which a group of parameters is accessed, which may lead to a lack of internal consistency if two or more related parameters are read when only part of them is updated.

eTPU provides mechanisms to guarantee parameter coherency. The most generic mechanisms for Host-eTPU coherency, suitable for any number of parameters, are:

- the use of Transfer Service Thread mechanism.
- the mailbox (or "software semaphore") mechanism.

These mechanisms, described in Multiple Parameter Coherency Methods, use microcode to transfer parameters from temporary buffers in SDM to their definitive locations (or vice-versa). These methods have the disadvantage of wasting processing and code memory resources.

eTPU also provides a **Coherent Dual-parameter Controller - CDC** - mechanism. It is used by Host to coherently transfer pairs of parameters from/to a parameter buffer located on SDM to/from the locations on SDM where parameters are accessed directly by the channels. Coherency is guaranteed by SDM access arbitration. Although limited to two parameters only, it has lower latency and wastes no microengine resources[14]. CDC usage is described in Coherent Dual-parameter Controller - CDC.

For parameters shared by both Engines, eTPU provides **Hardware Semaphores**. Coherency is assured given the semaphores are used to prevent concurrent access to the changing parameters. Microengine can request semaphores using specific microinstructions (see Semaphore Operations). Hardware Semaphores are described in detail in Hardware Semaphores.

Neither Host nor CDC have access to the hardware semaphores, but they can be combined with microcode transfer mechanisms if Host must coherently access parameters which are also shared by both Engines.

In order to ensure coherent access to a group of parameters by two or more contenders, **each contender must have atomic access** to the shared parameters. Atomicity conditions are discussed in Host Side Atomic Access and Microengine Side Atomic Accesses.

14. a microengine access to the SDM in the moment CDC is performing the transfer may suffer a maximum of two wait-states.

43.14.4.1 Host Side Atomic Access

Host side atomic accesses can be achieved by either of following ways:

- for one parameter, the SDM should be accessed by 32-bit-wide data transfers to ensure coherency
- for two parameters only, using the Coherent Dual Parameter Controller.
- indirectly, for any number of parameters, by requesting microcode to coherently access SDM in its behalf. The host side atomicity problem becomes, then, a microengine side atomicity problem. Some methods that use this approach to achieve coherency are described in Multiple Parameter Coherency Methods.

43.14.4.2 Microengine Side Atomic Accesses

43.14.4.2.1 Microengine Single Parameter Atomicity

SDM should be accessed by 32-bit-wide data transfers to ensure atomicity for 32-bit parameters. This applies either to Host-Microengine coherency or Microengine-Microengine coherency in a dual eTPU Engine system.

43.14.4.2.2 Microengine Dual Parameter Atomicity

Microengine has the ability to access two parameters coherently in back-to-back accesses, at random addresses: once it accesses SDM, it has priority over Host for another access in the next microcycle (see SDM Arbitration). Note that it applies **only to Microengine-Host** coherency. For Microengine-Microengine coherency in a dual eTPU Engine system, one must use Hardware Semaphores (see Hardware Semaphores).

Microengine dual back-to-back accesses are guaranteed to be atomic in relation to Host slave accesses or Coherent Dual-parameter Controller, regardless of semaphore usage: Host or CDC accesses cannot break-up a back-to-back Microengine access, neither Microengine can break a CDC transfer, due to the SDM arbitration mechanism described in SDM Arbitration.

Atomicity is not guaranteed if microengine enters halt state in the middle of a back-to-back access (see Microengine Halt State): Host can access SDM while microengine is halted in the middle of a back-to-back access.

43.14.4.2.3 Microengine Side Multiple Atomicity

Hardware Semaphores must be used for Microengine-Microengine coherency (more than 1 parameter) since two or more accesses from one Microengine are not atomic with respect to the other.

For multiple Microengine-Host coherency, the software methods described in Multiple Parameter Coherency Methods, or similar ones, must be used. Some of these methods rely on the fact that parameter access of a thread is atomic in relation to another thread in the same Engine, since a thread cannot be suspended (preempted).

For 1 parameter coherent access, or dual parameter coherency between only one Microengine and Host, the alternatives shown in previous sections apply.

43.14.4.3 Coherent Dual-parameter Controller - CDC

Dual-parameter coherency is supported by a Coherent Dual-parameter Controller hardware - CDC, which contends with microengine for SDM access. CDC atomically transfers, upon Host's command, two parameters from one area of the SDM to another. One area is a temporary (buffer) area, where the two parameters are directly read or written by the Host. This temporary area has to begin in an SDM address multiple of 2 words, and the two parameters must be sequential. The other area is the channel parameter area where the microcode normally accesses the parameters, usually with the channel relative address mode (see SDM Addressing Modes). In this area, the parameters transferred by CDC don't have to be sequential. A transfer from the temporary area to the channel area, when the Host sends data to the channel, is called a **write transfer**. Inversely, in a **read transfer** the parameters are copied from the channel area to the temporary area (channel to Host).

Coherency is guaranteed by the SDM access contention rules implemented in the SDM arbiter (see SDM Arbitration). CDC transfers are coherent in respect to the two Engines, so the target parameters in the channel area may be shared by channels on them both. During CDC operation, the Host may suffer from 3 up to 11 eTPU clocks wait states[15], and the Microengine(s) may suffer up to 2 microcycle wait-states[16] . CDC accesses are atomic with respect to Microengine(s) accesses to the SDM. Even when neither engine is in TST, CDC may suffer up to 4 eTPU clock internal wait-states from SDM arbiter, meaning 9 slave wait-states to Host, so that it does not break atomic back-to-back

15. The maximum number of Host wait states on CDC occurs when both microengines overlap their TSTs, delayed 3 eTPU clocks from each other.
16. One microcycle takes two eTPU clocks. Microengines get wait-states in multiples of microcycles, while Host and CDC wait-states are multiples of eTPU clocks.

accesses from microengine(s). CDC also cannot break TST preload accesses. Host can initiate CDC back-to-back transfers: there is no need of idle slave cycles between two transfers.

Note

SDM memory error detection and correction does not affect the number of wait-states to host due to CDC (see Memory Error Support).

CDC can be used to atomically fix SDM soft errors. For more details, see Error Fixing.

43.14.4.3.1 CDC Programming

The Coherent Dual-parameter Controller Register (see the ETPUCDCR register) is used to configure and initiate CDC transfers between the temporary area and channel parameter area. Host asserts STS bit in order to start the data transfer. CDC then contends for the SDM and starts the transfer. When the data transfer is complete, STS returns to 0. Host receives wait-states for writing STS=1 while CDC contends for SDM and during the transfer. The write access ends when CDC finishes the transfer. Host receives wait-states during the CDC transfer. If Host writes ETPUCDCR with STS=0 or does not write the STS byte, the CDC transfer does not occur. CDC programming can be summarized as follows:

1. if it is a write transfer, i.e., from Host to channel, write the two parameters into temporary area.
2. write ETPUCDCR with STS=1 and the remaining CDC programming parameters: parameter width (32 or 24 bits, field PWIDTH), transfer direction (read or write, field WR), temporary parameter area base address (field PBBASE), and the absolute addresses of the parameters to be transferred (concatenation of the fields CTBASE and PARM0/1).
3. if it is a read transfer, i.e., from channel to host, read the two parameters from the temporary area into Host memory/registers.

43.14.4.4 Hardware Semaphores

eTPU provides Hardware Semaphores accessible by the Microengine only. It is the responsibility of the application to ensure proper use of the semaphores (i.e., agree upon a specific semaphore and use it properly, to ensure coherency).

The eTPU microinstruction set has support for locking and freeing the semaphores, described in Semaphore Operations, and this is the only way to access them.

There are four semaphores available, which reduces the amount of collisions by assigning unrelated data transfers to different semaphores. Semaphores are used for parameters which can be shared by channels in different Engines, and for Engine-to-Engine synchronization. Semaphores are also the only way to ensure coherent access to parameters shared between the two Microengines.

Attempting to lock one semaphore (even not successfully) frees the other locked by the same Engine, ensuring one can lock just one semaphore at a time. That prevents deadlock conditions between the two Engines.

Microcode END command or Engine being in idle state (no thread executing) automatically releases all semaphores from one Engine side, even if a semaphore lock is done in parallel. However, it is recommended to write the microcode in a way which locks semaphores for the shortest required period, and frees them without waiting for the END command, to improve the performance of the other microengine. Semaphores are free after reset. An Engine can only free a sempaphore locked by itself.

Semaphore lock requests are always **non-blocking**, in the sense that they do not suspend the requester in case the semaphore is already locked. The semaphore status after the lock request - indicating if it was successfully locked or not - must be tested through the SMLCK microengine branch condition (see Branch Conditions).

43.14.4.5 SDM Arbitration

Up to four entities can access SDM:

- two Microengines (in a dual eTPU Engine system)
- the Coherent Dual-parameter Controller (CDC)
- the Host CPU (direct memory-mapped access)

The following rules specify the access priorities for contended access. They keep compatibility with the TPU3 dual parameter access atomicity, but only between the microengine and CDC (not Host accesses through slave bus).

1. Microengine accesses from the two eTPU Engines are interleaved between each other, but not with Host or CDC accesses;
2. The eTPU microengine(s) gives priority for SDM accesses to either the Host CPU or the CDC under any of the following conditions:

a. the microengine has completed accessing the second parameter in a back-to-back SDM access[17, 17].

b. the SDM was not accessed during the last arbitration slot for the microengine and the host does not loose the access to the other engine in the current arbitration slot[18, 18]

c. CDC is transferring data, after its first (read) access. Note that the CDC can be in middle of a data transfer of another pair of parameters, unrelated to the ones that microengine tries to access.

3. The eTPU microengine takes priority for SDM accesses under either of the following conditions:

a. the Host CPU or CDC has done a data transfer during the last access arbitration slot for the engine[18]. Also, the Host CPU does not hold a pending access against the other eTPU microengine.

b. the microengine is arbitrating for the access of its second parameter in a back-to-back access[1]. All pairs of back-to-back parameter accesses are coherent with respect to Host and CDC (not to the other microengine).

The direction (read or write) of any individual access by Host or microengine is irrelevant to the arbitration. The use of Normal or PSE SDM area by the Host is also irrelevant to the arbitration.

The first parameter preloading in a TST is considered first access by the arbiter, regardless of any access made at the END microinstruction of the previous thread, that is, the last access of a thread and the first preload are never considered a back-to-back access. On the other hand, the TST preload accesses are considered back-to-back and are, therefore, atomic with respect to Host or CDC.

Note

The Zero SDM operation (see Zero SDM Operation) is considered an SDM access for arbitration purposes both on writes and reads; the fact that read SDM data is discarded is irrelevant for arbitration.

17. if microengine tries to access the SDM in the following microcycles, the third and fourth consecutive accesses are considered the first and second of a new back-to-back dual access

18. The microengine access slot is between its own T4 and T2 edges (see Microcycle and I/O Timing).

43.14.5 Enhanced Channels

Enhanced Channels comprise hardware support for input digital signal processing and output signal generation. Each Channel is associated with one input and one output signal. Enhanced Channel logic is combined with Function microcode (and optionally Angle Mode logic) to implement Channel I/O functionality.

eTPU's Enhanced Channels are capable of **dual action**, meaning that each channel logic can handle two events at different times and/or cause two separated actions - these actions and events can be mutually dependent (with the first either blocking or enabling the other), or both independent, depending on the programmed **Channel Mode**.

Each Enhanced Channel contains event logic containing two **Event Register sets**, each set supporting one input and/or output action, the pair implementing dual action support. Each Event Register set contains two 24 bit registers: Match and Capture. The **Match register** holds the pending match value which is compared against one of the two time bases by an equal-only/greater-equal comparator. The **Capture register** captures one of the two time bases as a result of a Match or Transition detection. Service Requests are issued on particular combination of match and capture events, defined by the selected Channel Mode.

In the context of the eTPU channels, a **Match** is a comparison between a time base value and a channel Match register. If those two values are coincident, or the time base value is greater than the value of the Match register, a **Match Event occurs**. Depending on the channel mode of operation and current state of the channel logic, the match event may be recognized, i.e., change the state of the channel, or be ignored. A match event recognized by the channel logic is called a **Match Recognition**. Match Recognitions can cause, also depending on Channel Mode and current state, the channel to request service, configuring a **Match Service Request**.

eTPU uses two kinds of comparator to assert a Match Event: an **Equal comparator**, in which both the Match Register and the value of the selected time base must match exactly, and a **Greater-Equal comparator**. The Greater-Equal comparator considers any time base value between the range [N : N+0x800000-1] as a valid match against the value of N in the Match Register, even when the value N+0x800000-1 wraps around the point of origin (0x0). Refer to the following figure for an illustration of the matching values on a Greater-Equal comparator.

The second source of events for the eTPU channel is a **Transition** detected at the corresponding channel's input signal. Two distinct Transition detections can be programmed individually for each channel, allowing recognition of all possible combinations of edge detection. It is also possible to check the sampled state of an input

signal upon the occurrence of a Match: the sampling of the expected value is treated as a Transition, even if the input signal did not necessarily toggled at the time of the Match, or at any time at all.

Like Match Events, **Transitions Events** may or not be recognized by the channel logic. When they are, a **Transition Detection** occurs. As well as Match Recognitions, Transition Detections can issue a Channel Service Request, depending on Channel Mode and current state.

Transition Detections and Match Recognitions are sometimes simply called Transitions and Matches throughout this document, for short.

Input and output signals can be processed separately by the channel logic and microcode, and can also be combined such that Matches and Transitions are used to cause **output signal actions**. The output signals can also be directly controlled by microcode. Many event combinations are allowed for a channel, given the possibility of configuring pairs of matches and transitions for the dual action logic, where each event is able to block or enable the next one. There is a full set of Channel Modes described in Channel Modes, exploring all the capabilities mentioned here.

Each channel has its own set of registers and flags. They are selected, and made accessible to the Microengine, according to the value written into the microengine **CHAN Register** that points to the desired channel . Every time the CHAN register is written, even if with the same previous value, a channel is selected and its flags and registers are updated. For further detail, see the Channel Selection Register-CHAN topic in the General Channel Registers section .

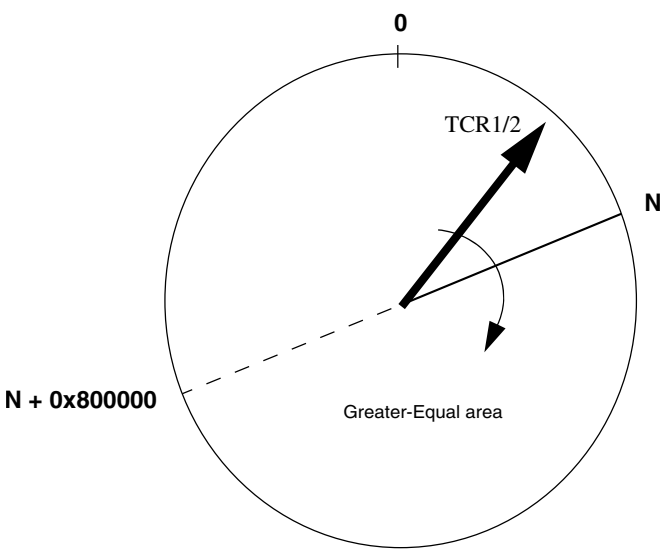


Figure 43-303. Greater-Equal Comparator

Beyond the request of services due to the signal and timing internal to each channel, one eTPU channel microcode can explicitly request service from another channel through the microengine **LINK Register**. A microcode write to the LINK Register asserts a service request to the channel whose number matches the contents of LINK. Refer to Channel Link for a complete description of this mechanism.

These service requests originated in the eTPU Enhanced Channels (either time base match, input signal transition, or link service request) result in a call to the corresponding channel service routine, which is the sequence of microinstructions that is called a **Thread**. For further detail, refer to Functions and Threads.

In addition to Event Logic, each Channel has an Output Buffer Enable signal, controlled by microcode, and an Enhanced Digital Filter, which eliminates spurious glitches on input pin signal. See the chip-specific eTPU information for how the Output Buffer Enable signal is controlled.

A high level diagram of Channel logic and registers is shown in the following figure.

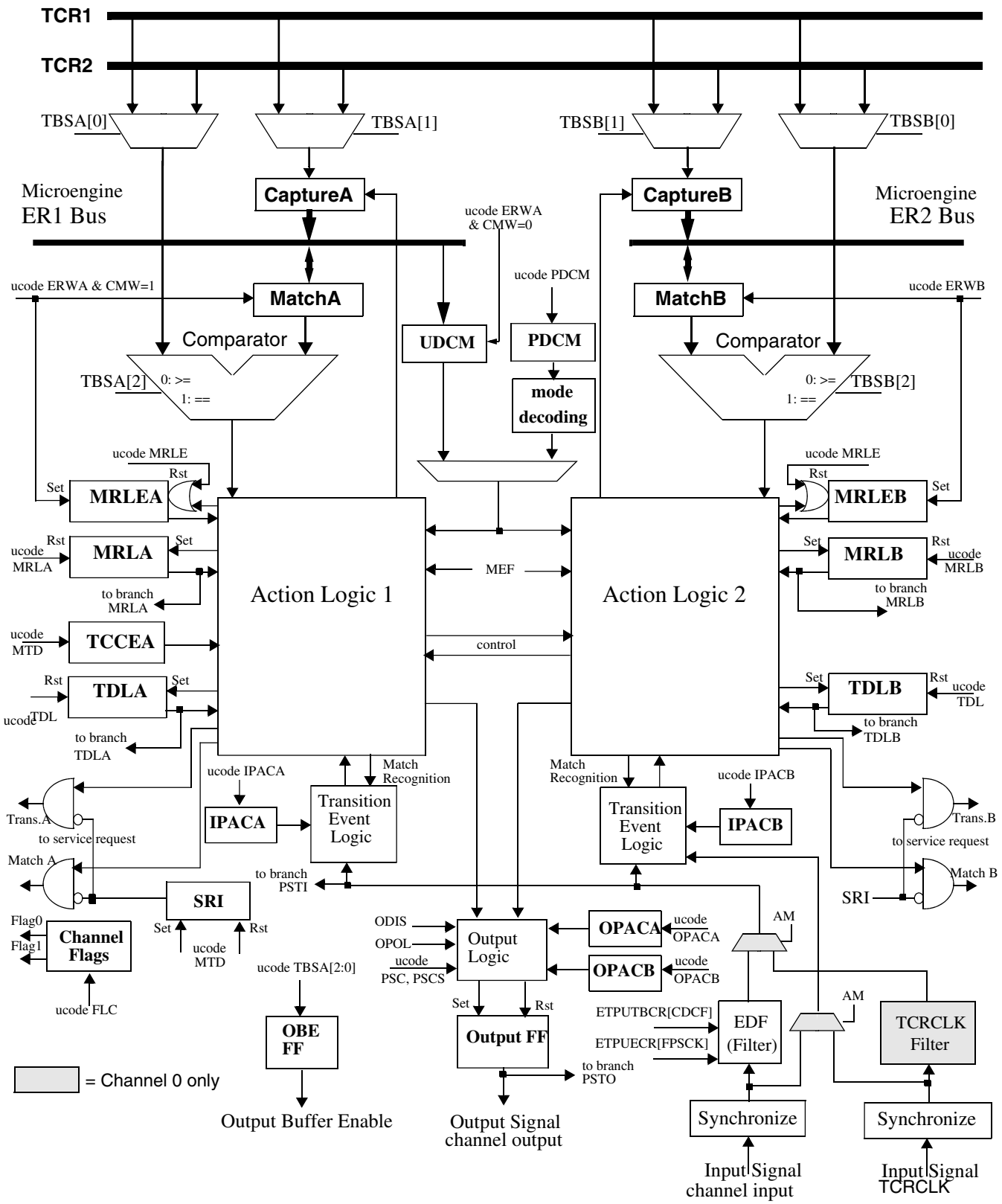


Figure 43-304. Channel Logic Block Diagram

43.14.5.1 Channel Registers and Flags

Channel configuration and control registers can be divided in the following groups:

- **Host Configuration and Control registers**, which define channel Function and parameter allocation in SDM, input signal filtering, manage Host interrupts, and are used for Host Service Requests; they can only be accessed by Host, except for the Function Mode bits which can be also tested by microcode.
- **Event Registers**, which can only be accessed by eTPU Microengine, through dedicated Channel Control microinstruction operations (see Channel Control and Configuration Microoperations); these registers are directly used to implement channel functionality, and include channel event status latches which can be directly tested by Microengine branch instructions.
- **Pin Control registers**, which basically define pin state and transition polarity (but not input signal filtering); they are accessible only by dedicated Channel Control microinstruction operations.
- **Link registers**, which implement the channel link mechanism that allows one channel to request service to another one; they are accessible only by microinstruction operations.
- **General Channel registers**: CHAN, SRI, Flag0/1, PDCM, UDCM.

Most of those registers are channel exclusive, i.e., there is one copy of them for each channel. Microcode can access registers from only one channel at a time. The Channel Selection (CHAN) register (see Channel Selection Register - CHAN), accessible only by microcode, defines the channel whose registers are being accessed, with exception of link register and function mode. CHAN register assumes the value of the channel to be serviced at the beginning of TST.

The Service Request Inhibit - SRI - register controls the generation of Service Requests on matches and transitions, also affecting channel logic behavior. For a full description see SRI - Match/Transition Service Request Inhibit Latch. Flag0/1 are used to select channel service threads based on channel software state. See Flag1,Flag0 - Channel "state resolution" flags for more details.

Host Configuration and Control registers are described in the Channel Configuration and Control registers section.

Time Base configuration is common to all channels, and described in Time Bases. Time Base selection for matches and captures, however, is individual to each channel, and is part of the Event Registers.

Link registers are described in Channel Link.

The following sections describe the Event Registers and Pin Control registers.

43.14.5.1.1 ER - Event Registers

Each channel contains two identical Event Register sets, named ERA and ERB, corresponding to the two actions supported. Each Event Register set contains:

- a 24-bit Match register (Match A or Match B), which holds a match value. This value is compared against the selected match time base (TCR1 or TCR2).
- a 24-bit Capture register (CaptureA or CaptureB), which samples the selected capture time base (TCR1 or TCR2)
- a Time Base Selection register (TBSA or TBSB)
- a Match Recognition status flag (or latch) (MRLA or MRLB)
- a Match Recognition Enable latch (MRLEA or MRLEB)
- a Transition Detection flag (or latch) (TDLA or TDLB)
- a Transition Continuous Capture enable (TCCEA only)

ERA and ERB are associated with the first and second events in double action modes, not necessarily in that order. The order of Match events associated with ERA and ERB depends on the programmed channel mode, the MatchA and MatchB values, and the timebases selected by TBSA and TBSB. Similarly, the order of Transition events associated with ERA and ERB depends on the programmed channel mode, and the transition detection selected by IPACA and IPACB.

These registers are directly or indirectly accessed by the microcode. TBSA and TBSB registers are defined in TBSA and TBSB - Time Base Selection Registers. The other registers are explained in Match Recognition and Transition Detection and Time Base Capture.

Access to the Event Registers is qualified by the channel currently selected by the microengine (i.e., the channel value currently in the CHAN register). During the channel transition period (automatic CHAN assignment), or whenever CHAN is written by microcode, Capture values of the new selected channel are sampled into Microengine registers ERTA and ERTB, therefore becoming visible to the microcode. At the same time, updated values of MRLA, MRLB, TDLA and TDLB are sampled into the branch logic, making the register values and the flags coherent with respect to each other and with the thread selected by the Scheduler.[19]

19. the thread selected is determined by the Entry Point which, in turn, is determined partially by the channel latches. See Entry Point Address Generation.

Note

The Function Mode bits are also sampled from the Host interface on Time Slot Transition, so that they remain constant to microengine even when Host changes them.

During service, the microcode can access updated values of the Event Registers of any channel by writing the channel number to CHAN. Writing CHAN with the same value (CHAN := CHAN) updates ERTA and ERTB with the new captured values, the branch logic with updated MRLA/B and TDLA/B flags. Writing CHAN with a different value does the same with the values from the newly selected channel.

Match values are also accessed through ERTA and ERTB Microengine registers, which are copied to/from the channel MatchA and MatchB registers by specific microinstruction operations.

Microcode writes to the flags and selections (MRLA/B, TDLA/B and TBSA/B) are immediately effective to the channel. The MRLA/B and TDLA/B branch conditions are also immediately reset when their correspondent flags are reset by microcode. Match registers are indirectly written by microcode through ERTA/B. MRLEA/B is unconditionally asserted when respective Match register is updated from ERTA/B, and its negation is immediate.

The following table summarizes Event Registers accesses.

Table 43-337. Event Registers microcode accesses

Register	Access Type	Sampled from channel	Update to channel	Microcode fields[1]	Reset value[2]
CaptureA, CaptureB	read through ERTA/B	to ERTA/B on CHAN assignment	no	T2ABD	n.a.
MatchA, MatchB	read and write through ERTA/B	to ERTA/B by microcode	from ERTA/B by microcode	ERWA, CMW, ERWB, T4ABS	n.a.
MRLEA, MRLEB	write to 0 (negate) directly; write to 1 (assert) upon MatchA/B update from ERTA/B	no	immediate	MRLE, ERWA, ERWB	0, 0
TBSA, TBSB	write only	no	immediate	TBSA, TBSB	000, 000
MRLA, MRLB	flag test on branch, write to 0 (negate) only	on CHAN assignment	immediate	BCC (test) MRLA, MRLB (reset)	0, 0
TDLA, TDLB	flag test on branch, write to 0 (negate) only	on CHAN assignment	immediate	BCC (test) TDL (reset)	0, 0
TCCEA	write only	no	immediate	MTD	0

1. see Microinstruction Set.
2. n.a. means that value of the register is undetermined after reset.

43.14.5.1.1.1 MatchA and MatchB Registers

MatchA and MatchB registers hold a match value to be compared against the selected channel time base. A match value can only be written into the Match register by microcode, through ERTA/B microengine registers (see Write Channel Match and UDCM Registers). Microcode can also read the Match register as a special T4ABS source operation, when T4ABS=0101, and the source for T4ABS is selected from the second register set. In this operation, MatchA/B registers are copied into ERTA/B registers (see Selecting Sources and Destination). For more information on time base matches, see Match Recognition.

43.14.5.1.1.2 CaptureA and CaptureB Registers

CaptureA and CaptureB registers capture the selected channel time base. CaptureA/B registers cannot be directly written or read by microcode. During the Time Slot Transition (TST) or during CHAN assignment, CaptureA/B registers are copied into ERTA/B microengine registers. For more information, see Transition Detection and Time Base Capture.

43.14.5.1.1.3 TBSA and TBSB - Time Base Selection Registers

TBSA/B are 3-bit registers which have the following effect on channel configuration:

- Selection of the timebase (TCR1 or TCR2) to be compared against the match values in MatchA and/or MatchB registers.
- Selection of the timebase (TCR1 or TCR2) to be captured in the CaptureA and/or CaptureB registers by a match or transition detection event.
- Selection of comparator mode to be used with MatchA and MatchB registers: equal-only or greater-equal.

After reset TBSA/B are 000. The following table shows values of TBSA and TBSB for configuration selection. Note that the time base selection for capture is independent of the time base selected for matches.

Table 43-338. TBSA/B Programming States

TBS bit	2	1	0
bit value	Comparator selection	Capture time base selection	Match time base selection
0	greater or equal	TCR1	TCR1
1	equal-only	TCR2	TCR2

TBSA/B are written through the microcode fields TBSA/B (see Comparator and Time Base Selection). Note that microcode field TBSA is also used to control the OBE pin control register (see Pin Control Registers), which is separate from the TBSA register.

43.14.5.1.1.4 MRLA/B - Match Recognition Latches

See MRLA/B - Match Recognition Latches.

43.14.5.1.1.5 MRLEA/B - Match Recognition Latch Enable

See MRLEA/B - Match Recognition Latch Enable.

43.14.5.1.1.6 TDLA/B - Transition Detection Latch

See TDLA/B - Transition Detect Latches.

43.14.5.1.1.7 TCCEA - Transition Continuous Capture Enable

See TCCEA - Transition Continuous Capture Enable.

43.14.5.1.2 Pin Control Registers

Pin Control Registers are replicated one per channel, accessed only by microcode and qualified by the CHAN register in the same way as Event Registers.

The following table lists Pin Control Registers, explained in following subsections, and their accesses.

Table 43-339. Pin Control Registers microcode accesses

Register	Access Type	Sampled from channel	Update to channel	Microcode fields[1]	Reset value
IPACA, IPACB	write only	no	immediate	IPACA, IPACB	000,000
OPACA, OPACB	write only	no	immediate	OPACA, OPACB	000,000
PSTI	flag test on branch	no	no	BCC	0

Table continues on the next page...

Table 43-339. Pin Control Registers microcode accesses (continued)

Register	Access Type	Sampled from channel	Update to channel	Microcode fields[1]	Reset value
PSTO	flag test on branch, write	no	immediate	BCC (test) PSC, PSCS (set)	0
OBE[2]	write only	no	immediate	TBSA values 1000,1001	0 (negated)
PSS	flag test on branch	on CHAN assignment	no	BCC	n.a.[3]
PRSS	flag test on branch	on CHAN assignment	no	BCC	n.a.[4]

1. see Microinstruction Set .
2. Although this register exists for code compatibility reasons, OBE functionality is not available on this chip.
3. PSS is PSTI or PSTO sampled on CHAN assignments and at thread start.
4. PRSS is PSTI sampled on channel service request.

43.14.5.1.2.1 IPACA,IPACB and OPACA,OPACB - Input and Output Pin Action Control Registers

These registers determine the transition detections and output pin actions which takes place due to match or transition events. Each field is three bit wide. After reset, the IPACA/B and OPACA/B registers are set to 000. IPACA and IPACB registers are mutually independent and have identical encoding, and so do OPACA and OPACB. The table below shows IPAC and OPAC encoding. Note that transition detections are independent from the output actions, but some options of output actions are triggered by transition detections. Output actions can also be triggered by Matches. For a detailed definition of Transition Detections, see Transition Detection and Time Base Capture.

IPACA/B define the Transition events treated by channel logic. Although the name "transition" is generically used for the transition detections, IPAC options 100 and 101 do not really detect transitions: they actually sample the state of the input signal at the occurrence of the corresponding Match (Match A used for IPACA, Match B used for IPACB).

Table 43-340. IPACA/B and OPACA/B Encoding

value	IPAC meaning	OPAC meaning
000	Do not detect transitions	Do not change output signal
001	Detect rising edge only	Match[1] sets output signal high
010	Detect falling edge only	Match[1] sets output signal low
011	Detect either edge	Match[1] toggles output signal
100	Detect input signal = 0 on Match[1, 1]	Transition sets output signal low
101	Detect input signal = 1 on Match[1]	Transition detection sets output signal high

Table continues on the next page...

Table 43-340. IPACA/B and OPACA/B Encoding (continued)

value	IPAC meaning	OPAC meaning
110	reserved	Transition detection toggles output signal
111	n.a.[2, 2]	n.a.[2]

1. Match A is used for IPACA/OPACA, and Match B for IPACB/OPACB.
2. On the microinstruction fields IPACA/B and OPACA/B this value is neutral, meaning that IPAC/OPAC register values are not changed.

43.14.5.1.2.2 Output Pin Control Logic and Pin State Output Register - PSTO

The output signal control logic uses OPACA/B, the Pre-defined Channel Mode (PDCM) and the User Defined Channel Mode (UDCM), and the microcode Pin State Control (PSC and PSCS) fields. It is responsible for setting the Pin State Output (PSTO) register to the specified logic value required by microcode, by input events, or by Match A and/or Match B events. The PSTO register stores the driven pin state determined by the Pin Control logic. The Output Buffer Enable signal, if used at MCU integration, must be set by microcode (using TBSA field) to make the pad propagate the PSTO register output to the actual pin.

PSTO register output also goes to the microengine branch logic, enabling branching on the driven pin state . PSTO is set to 0 on reset.

The PSC and PSCS microcode fields are used for setting the PSTO register to a fixed value, or to the value specified by the OPACA or OPACB microcode field, as shown in the following table

Table 43-341. PSC and PSCS encoding

PSC	Output Pin Action
00	Force pin state according to OPACA (PSCS=0) or OPACB (PSCS=1).
01	Force pin high.
10	Force pin low.
11	Do not change pin state.

For details refer to Channel Control and Configuration Microoperations.

On occurrence of match recognitions or transition detections, the pin control logic sets PSTO value according to the event number (Match A/Transition A or Match B/Transition B) and the contents of OPACA/IPACA or OPACB/IPACB registers. There are cases in which two match or transition events may occur at the same time, each of them trying to force a different pin action. The resolution of the selected match event which sets the

value depends on the Pre-defined Channel Mode (PDCM) register and the User Defined Channel Mode (UDCM). For details refer to Match/Transition Pin Action Conflict Resolution.

43.14.5.1.2.3 PSTI and PSS - Pin State Input and Pin Sampled State Registers

During the time slot transition period, or whenever the CHAN register is written by microcode, the filtered[20] input signal PSTI or output signal PSTO (selected by the ETPUCxCR bit ETPD) is sampled into the branch logic as the PSS flag (see Table 43-409). The microcode can then branch on either the currently driven (PSTO) or input (PSTI) pin state, or on sampled pin state (PSS, which is stable as long as CHAN does not change).

Note

If a transition occurs simultaneously (after the filter) with the CHAN assignment, PSS samples the new pin value. Therefore, if TDLA/B is cleared simultaneously with the assignment CHAN := CHAN, the occurrence of a transition at this very moment can still be tested with PSS.

43.14.5.1.2.4 PRSS - Pin Request Service Sample

Channel logic can, depending on its state and programmed mode, request service to the eTPU microengine (see Functions and Threads). When the channel logic issues a service request, the filtered input signal PSTI is sampled into an internal channel flag PRSS. There is one such PRSS flag for each channel (see Pin Control registers). Channel PRSS keeps its value until all its service request sources are cleared and until a new channel service request rises.

Note

Only channel service requests update PRSS: it is updated neither by host service requests (HSRs) nor by link service requests. Moreover, a channel service request updates PRSS even if the channel is disabled (its CPR set to 0, see the ETPUCnCR register).

The channel PRSS flag is sampled into the branch logic as the PRSS flag (see Table 43-409) during the time slot transition period, or whenever the CHAN register is written by microcode.

20. The filter can be bypassed.

43.14.5.1.3 General Channel Registers

These registers control other aspects of channel logic. Except for CHAN, they are unique per channel. The following table summarizes the registers and access options.

Table 43-342. General Channel registers microcode access

Register	Access Type	Sampled from channel	Update to channel	Microcode fields[1]	Reset value
CHAN	read/write	n.a.[2, 2]	n.a.[2]	T4ABS, T4BBS, T2ABD	defaults to serviced channel at thread start
PDCM	write only	no	immediate	PDCM	1100 (sm_st)
UDCM	write only	no	from ERTA by microcode	CMW, ERWA	parameter value defined at integration[3]
SRI	write only	no	immediate	MTD	1
Flag1,Flag0	branch flag test, write	no	immediate	BCC,FLC	0, 0

1. See Microinstruction Set.
2. CHAN is common to all channels in the Engine.
3. See the eTPU Integration Guide for parameter details and default value.

43.14.5.1.3.1 Channel Selection Register - CHAN

CHAN is the register that holds the number of the channel that qualifies the context of most Channel Registers, including Pin Control and ER accesses, and is common to all channels in a same Engine.

When a thread starts to be executed, the contents of CHAN register is automatically updated on Time Slot transition to the number of the channel to be serviced. Serviced channel is constant during channel servicing, but the selected channel can be changed any time by microengine writing into CHAN register.

Some microinstructions are affected by the serviced channel instead of CHAN. These are:

- Conditional branch using LSR (see LINK Register) or Function Mode (see the ETPUCnSCR register.
- Negate channel flag LSR, Interrupt CPU and Data Transfer Request (see Channel Interrupt and Data Transfer Requests).

When CHAN register is written, accesses are qualified by the new CHAN register value from the instruction following CHAN assignment on, except CaptureA/B sampling into ERTA/B and Match register writing from ERTA/B (see CHAN assignment, Read Match and ERWA/B).

Writing CHAN (including with the same value, CHAN:= CHAN) updates ERTA and ERTB with the new captured values, the branch logic with updated MRLA/B and TDLA/B flags.

The following table shows the commands, flags and registers selected by the CHAN register value.

Table 43-343. CHAN-selected features

Feature Used	Selected by CHAN
channel-relative SDM access	YES
Branch using PSS, PRSS, PSTI and PSTO channel flags.	YES
Branch using MRLA/B, TDLA/B, Flag0/1[1]	YES
Branch on all other conditions[2]	no
ERTA/B Value	YES
configure (selected) channel	YES
channel commands applied to: MRLA/B, TDLA/B, TBSA/B, IPACA/B,OPACA/B, PSC, PSCS, OBE, PDCM, MRLE, Flag0/1	YES
channel command: set/reset SRI	YES
channel command: write to Match registers (ERWA/B)	YES[3]
channel command: read Match registers into ERTA/B	YES
clear LSR	no
channel interrupts and data transfer requests (CIRC)	optionally

1. In TPU, these conditions retained the old values.
2. Refer to Branch Operations
3. If write Match (ERWA/B) occurs at the same time of a CHAN assignment, the channel which is target of the write is the one selected by the new CHAN value. See CHAN assignment, Read Match and ERWA/B.

43.14.5.1.3.2 PDCM - Predefined Channel Mode

PDCM determines the channel mode assigned to the channel. Channel mode defines much of the channel logic behavior, specially how matches blocks and enables transitions and vice-versa, as well as occurrence of time base captures and service requests based on matches and transitions. For a complete description see Channel Modes.

PDCM is a 4-bit register set by the microcode field of the same name (see Predefined Channel Modes), and cannot be read or tested in branch instructions. The table below relates the PDCM value with channel modes. The second column specifies the mnemonic used to reference the mode, introduced in Channel Modes Overview . There is one PDCM for each channel, initialized with 1100 on reset.

PDCM is also used to select the User Programmable Channel Mode. If this selection is made, the channel behavior is defined by the settings of the UDCM register (see UDCM - User Defined Channel Mode).

Table 43-344. PDCM encoding

PDCM	Channel mode
0000	em_b_st
0001	em_b_dt
0010	em_nb_st
0011	em_nb_dt
0100	m2_st
0101	m2_dt
0110	bm_st
0111	bm_dt
1000	m2_o_st
1001	m2_o_dt
1010	User Defined Channel Mode
1011	reserved
1100	sm_st[1]
1101	sm_dt
1110	sm_st_e
1111	n.a.[2]

1. This is the reset value, also compatible with TPU channel behavior.
2. This value is used as a neutral (do not change) value in the PDCM microinstruction field.

43.14.5.1.3.3 UDCM - User Defined Channel Mode

UDCM holds the control signals that define the channel logic behavior in terms of Match and Transition blocking and enabling, captures and service requests triggered by events. The register layout is defined in the figure below. The individual fields are defined in Channel Mode Logic and Event Flags. There is one UDCM register for each channel, which can be independently programmed.

UDCM can only be written into the Match register by microcode, through ERTA/B microengine registers (see Write Channel Match and UDCM Registers).

13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSR		MCAP	M1ET	M1EM2	M1BM2	M2BM1	M2BT	T1BM1	T2BM1	TBM2	T1ET2	TSR	TCAP

Figure 43-305. UDCM Register

43.14.5.1.3.4 SRI - Match/Transition Service Request Inhibit Latch

SRI blocks channel service requests due to the assertion of MRLA/B and/or TDLA/B. SRI does not affect recognition of Link Service Requests or Host Service Requests, neither MRLA/B or TDLA/B microcode branch tests nor Entry Table selection[21] . SRI is asserted during reset and is controlled by microcode field MTD.

To unburden the microengine, SRI asserted configures a channel "dumb" regarding the servicing of match and capture channel service requests. Even with SRI=1, TDLA/B and MRLA/B can still be asserted, and the level specified by the OPAC (Output Pin Action Control) registers will be output to the pin.

43.14.5.1.3.5 Flag1,Flag0 - Channel "state resolution" flags

Each channel has a pair of flags, simply called Flag0 and Flag1, that can be set/reset by microcode through microinstruction field FLC. FLC sets/resets Flag0/1 of the channel selected by CHAN. These flags can be tested by microcode, and are also used to resolve the microcode entry point for the channel service (see Entry Points"). Flag0 and Flag1 are, so, typically used for fast state resolution. FLC microinstruction field also allows Flag1,Flag0 to be copied from selected bits of P register high byte, which is also meant to be used to hold application state. Flag0 and Flag1 are both zero out of reset.

43.14.5.2 Match Recognition

The match operation is performed every microcycle by comparing the channel MatchA and MatchB registers against the value of the TCR bus specified for each match. TCR1 or TCR2 bus is selected according to TBSA and TBSB fields. Both results have effect on the next clock cycle (see Microcycle and I/O Timing).

A Match A/B event is qualified by a set of match enabling conditions to the Match Recognition Registers MRLA/B. To recognize the match, the following match enabling conditions are required:

21. In TPU, SRI also blocked TDL and MDL branches and enabled any transition to capture time base.

- For IPACA/B=0xx, Match Enable Flag (MEF), qualified by the channel currently being serviced must be asserted. For IPACA/B =1xx, Match A/B is always enabled (even during Time Slot Transition - TST), regardless the state of the Match Enable Flag (MEF). See MEF - Match Enable Flag for the conditions of MEF assertion.
- Match Recognition Latch Enable 1/2 (MRLEA/B) is asserted. A match event recognition may only occur if its corresponding MRLEA/B bit is set, which only happens upon a write to a channel match register by the microcode, copied from ERTA/B. MRLEA/B is negated when the respective match occurs or, in some double match channel modes, when a match for the other Match register occurs. It ensures that the greater-equal comparison will not cause additional matches[22] .
- In selected modes (see Channel Modes), the particular conditions of MRL and TDL flags of the other event, i.e:
  - MRLA, TDLA enable or block MRLB;
  - MRLB, TDLB enable or block MRLA.
- The respective MRL is negated.
- In selected modes (see Channel Modes), the state of its respective TDL flag.

If the Match A and/or Match B conditions are met, the channel immediately forces the pin state if specified by the appropriate OPACA/B registers (Output Pin Action Control 1/2) and, in some cases, by IPACA/B registers. Refer to IPACA,IPACB and OPACA,OPACB - Input and Output Pin Action Control Registers.

If both Match A and Match B events occur at the same time, with conflicting pin actions, the priority over the pin action is mode dependent. For further details on pin action resolution refer to Match/Transition Pin Action Conflict Resolution.

43.14.5.2.1 MRLA/B - Match Recognition Latches

MRLA/B indicate the recognition of a match event detected by the comparator. They can be asserted either on T2 or T4 (see Microcycle and I/O Timing). Assertion of MRLA/B issues a match Service Request in specific channel modes, depending on previous events and state of SRI. After reset MRLA and MRLB are both negated.

22. Microcode can also negate MRLEA/B.

When MRLA or MRLB is asserted, it may change the output signal level according to the Input and Output Pin Action Control registers (refer to IPACA,IPACB and OPACA,OPACB - Input and Output Pin Action Control Registers). Assertion of MRLA/B causes a capture of one or two time bases, according to the selected mode capturing scheme (see Transition Detection and Time Base Capture)

A match recognition is self-blocking, regardless of Channel Mode: once MRLA (MRLB) has been asserted, it negates its associated MRLEA (MRLEB) register, preventing future match recognitions, until the associated match register is rewritten by microcode. The microcode has to enable new matches by updating the new match value in the MatchA (MatchB) register[23] . In addition, assertion of MRLA/B can block its twin MRLB/A, depending on the channel mode. In some double match blocking channel modes, Match A/B event blocks the occurrence of Match B/A in a "first win" scheme.

It is the transition from 0 to 1 in MRL that causes the Match actions, except the service request: apart from MRLEA/B negation(s) and service request, no action due to a Match occurs if MRL was already set to 1, even if the other MRL assert conditions are satisfied. However, if a Match and a microoperation negating its corresponding MRL occur at the same time, MRL negation by microcode overrides its assertion, but any dependable captures and pin action occurs anyway (if MRL was already negated before), and also the negation of MRLE(s) (the respective one and, in some channel modes, the other, regardless of MRL state before). Note that MRLE must have been set before (by writing a new Match value).

43.14.5.2.2 MEF - Match Enable Flag

MEF is a one-bit latch that is unique for all channels in an Engine.

MEF can selectively enable assertion of MRLA/B, depending on the IPACA/B field. For IPACA/B=0xx, MEF=1 enables assertion of MRLA/B for the scheduled channel during service. For IPACA/B=1xx, Match A/B is always enabled, regardless the state of the MEF, but it still depends on the other Match recognition conditions. Matches of channels not being serviced are never disabled by MEF.

MEF is not accessible by Microengine or Host. MEF is negated for two microcycles in the beginning of the time slot transition period. After two microcycles (plus wait-states) into TST, the ME bit in the entry point is copied to MEF to allow selective enabling of MRL for each thread (refer to Time Slot Transition). MEF is asserted unconditionally soon after a thread ends.

If a channel service needs to postpone a programmed match, MEF assures that microcode wins the race against match event after time slot transition (only for IPAC=0xx).

23. Before that, microcode should also negate MRLA (MRLB), otherwise an old match may be recognized by the scheduler and serviced as a new one

Note that a match event may be lost during the periods when MEF is negated only if:

- the match comparator is configured for "equal-only" behavior, and
- IPACA/B=0xx, and
- TCR increments at the rate of eTPU clock divided by 2 or faster.

When the comparator is configured as "greater-equal", the match event that occurred when MEF was negated may be recognized after MEF is asserted again, due to the "greater than" condition.

43.14.5.2.3 MRLEA/B - Match Recognition Latch Enable

MRLEA/B is negated upon the assertion of its respective MRLA/B. In blocking match channel modes it may also be negated together with the assertion of the twin MRLB/A. The MRLEA/B bits ensure that data captured due to the first match event will not be overwritten when MRLA/B is negated: due to greater-equal comparison, the match condition continues to be true, but should not cause another capture event.

In addition to negation by local match event, the microcode can negate both MRLEA and MRLEB, to block pending matches, and also MRLA/B, individually. This action will prevent future match events from the selected channel.

Writing the MatchA/B registers by microcode to schedule the next match values sets MRLEA and/or MRLEB and enables new matches. This setting overrides the MRLE negation conditions due to channel logic or microcode (see Channel Modes). By combining write to Match A/B with MRLEA/B negation microinstructions, the microcode can negate one MRLE while asserting the other.

Note

If the MRLE negation conditions continue after writing MatchA/B registers, the respective MRLE does not keep asserted. For instance, if MRL=1 and a match is programmed for a time value in the past during a thread with ME=1, then MRLE will be cleared soon after MatchA/B is written, even though a match does not occur (because MRL was already asserted, neither captures nor pin toggles occur).

When the match register is updated (with MRLE already asserted before) and field MRLA/B = 1 (no clear, see Clear Transition/Match Event Registers) and MRLA/B flag is zero, the eTPU behaves exactly as the TPU, that is: a match that comes concurrently with the rewrite of the match register, matching the old value, sets the MRL, as if the setting of

the MRLE due to match register write had precedence over its clear by the match at that moment. After this simultaneous operation, the MRLE value stays at 1, and the captured time base value, if any, reflects the match value.

When the match register is updated (with MRLE already asserted before) and field MRLA/B = 0 (clear MRL, see Clear Transition/Match Event Registers) and MRLA/B flag is zero, the match captures will occur, the MRLA/B flag will keep negated, and MRLE will stay asserted. Special care must be taken when a match is reprogrammed on a channel running at T2/T4 timing, please see the details in the “T2/T4 Channel Timing” section.

43.14.5.3 Transition Detection and Time Base Capture

Time Base Capture(s) occur when the value of a specified TCR is sampled into the CaptureA and/or CaptureB register. TBSA[1] and TBSB[1] select which TCR will be captured in CaptureA and CaptureB, respectively.

A capture event may occur due to either of the following events:

- the assertion condition of Match Recognition Latch (MRL), even if MRL is simultaneously negated by microcode
- the assertion condition of Transition Detection Latch (TDL), even if TDL is simultaneously negated by microcode.
- any Transition Event specified by IPACA if both the Transition Detection Latch TDLA and Transition Continuous Capture Enable TCCEA are asserted.

A capture event occurs together with the assertion of MRL or TDL either on T2 or T4 positive edges, and captures the time-base value that caused the match, even if TCR1/2 increments concurrently with the assertion (see Microcycle and I/O Timing)[24, 24] . MRLA/B and TDLA may, depending on the channel mode, inhibit the capture of the second event's TCR into CaptureA/B. As a general rule, values captured by signal transitions are not overwritten by values captured by match events.

When the enable bit TCCEA is asserted, captures due to Transition Events also occur after TDLA is asserted. Those captures happen on transition events specified by IPACA, and the TCR value is saved into CaptureA register only.

The capturing scheme is defined by the Channel Mode programmed at register PDCM, or at register UDCM when User Defined Channel Mode is selected . For more information on mode-dependent capture schemes refer to Channel Modes.

24. In TPU3, when TCR1 was counting at maximum rate of eTPU clock divided by 2, the next value was captured.

43.14.5.3.1 TDLA/B - Transition Detect Latches

TDLA/B indicate detection of specific transition occurrences on a channel input signal. TDLA and TDLB assertion causes service request in single and double transition predefined modes, respectively. TDLB assertion does not cause Service Request in single transition predefined modes, and TDLA assertion does not cause Service request in double transition predefined modes. In single transition channel predefined modes TDLB can be asserted on the second transition, but does not generate Service Request. Yet on predefined modes, TDLB assertion is enabled only if TDLA is asserted to detect an ordered input signal double transition. All the restrictions above, however, may be overridden by using the User Defined Channel Mode. The IPACA and IPACB registers indicate the programmed edges of the first and second detected transition, respectively.

The sampling of a determined value (0 or 1) on the input signal due the occurrence of a Match is also treated as a "transition", depending on IPACA/B programming (see IPACA,IPACB and OPACA,OPACB - Input and Output Pin Action Control Registers). When using a channel mode where the transition1 is initially blocked and IPACA is programmed to detect such "transitions", the occurrence of a Match A only unblocks the transition after the sampling. It means that the transition on the first Match (IPACA configurations 100 and 101) is not effective on predefined modes where Transition A is enabled by Match A (m2_st, m2_dt, m2_o_st and m2_o_dt) or user-defined modes with UDCM bit M1ET asserted . A Transition A can still happen after the Match A, however, if MatchA register is reprogrammed without clearing MRLA.

TDLA/B assertion conditions initiates a capture event of one or both selected TCR buses. TDLA or TDLB transition event generates a Service Request, depending on channel mode, previous events and the state of SRI. For more information on service request scheme refer to Entry Point Address Generation" and Channel Modes.

Assertion of TDLA/B occurs on either T2 or T4 positive edges. The capture event occurs on the same clock, and captures the time base value present when TDLA/B was asserted[1] . TDLA and TDLB are negated during reset and may also be negated independently by microcode. TDLA/B is reset by no way other than reset and microcode.

It is the transition from 0 to 1 in TDL that causes the Transition actions: even if TDL assert conditions are satisfied, no action due to a Transition occurs if TDL was already set to 1. However, if a Transition and a microoperation negating TDLs occur at the same time and TDL was already negated, TDL negation by microcode overrides its assertion, but any dependable captures and pin action occurs anyway.

43.14.5.3.2 TCCEA - Transition Continuous Capture Enable

TCCEA enables capture from transitions after the TDLA flag is set. TCCEA is negated on reset, so that a capture occurs only when TDLA asserts. TCCEA can be set and reset by microcode only, through the instruction field MTD (see Disable Match and Transition Service Requests). It can only be set together with inhibiting of the channel service requests (SRI = 1)[25].

When TCCEA is asserted, the transition events specified by IPACA that occur after TDLA is set also cause captures, into the CaptureA register only. However, output actions related to transition events are still blocked by TDLA.

43.14.5.4 Channel Modes

The Enhanced Channels support various modes of operation combining Match A/B recognition and transition detection events which set MRLA/B and TDLA/B. The channel mode is individually set for each channel by eTPU microcode, through the PDCM register (see PDCM - Predefined Channel Mode). The PDCM register selects among a set of 13 predefined channel modes, and also a user-defined channel mode.

The order in which events occur, combined with assigned channel mode, establish which following event detections are inhibited and/or enabled, as well as the actions taken: Time Base capture, flag setting (MRLA/B, TDLA/B), match disabling (MRLEA/B), output signal transition, and Service Request. Those channel mode characteristics are fixed in the predefined modes, but can be individually programmed in the user-defined channel mode.

A generic description of channel modes from the usage point of view can be found in Channel Modes Overview . Each mode is named with a mnemonic acronym for terse reference. The individual programmed attributes of the user-programmable channel mode are also described.

The modes are used differently for input and output signals, as explained in Predefined Channel Modes on Input Signal Processing and Channel Modes on Output Signal Generation. Modes also allow combining input processing and output generation in a single channel, as exemplified in Combining Input and Output Signals.

Channel Mode Logic and Event Flags shows a structural definition of channel logic and its relation to channel modes. A dynamic, event-oriented view of each channel mode can be found in Channel Mode Summary.

25. TCCEA provides compatibility with TPU when service request is inhibited.

43.14.5.4.1 Channel Mode Logic and Event Flags

Figure 43-306 shows a more detailed diagram of channel mode logic. The logic shown here is not necessarily identical to the actual channel logic implementation, but is equivalent with respect to conditions for event blocking, enabling, capture, and service requests.

Signals MSR, TSR, MCAP, TCAP, M1ET, M1EM2, M1BM2, M2BM1, M2BT, T1BM1, T2BM1, TBM2 and T1ET2 are decoded from programmed channel mode PDCM in predefined modes, and come directly from the UDCM register when user-defined mode is selected:

- TSR (1 bit) defines Service Requests issued by Transitions, as shown in Table 43-346.
- MSR (2 bits) defines Service Requests issued by Matches, as shown in Table 43-345.
- TCAP (1 bit) defines time base captures caused by Transitions, as shown in Table 43-346
- MCAP (1bit) defines time base captures caused by Matches, as shown in Table 43-347.
- M1ET, M1EM2, M1BM2, M2BM1, M2BT, T1BM1, T2BM1, TBM2, T1ET2 (1 bit each) define Match and Transition reciprocal blocking and enabling, as well as Transition ordering, as shown in Table 43-348 and Table 43-349.

Table 43-350 shows the decoded values of those control signals for each predefined channel mode. The first column shows the mnemonic reference for the predefined channel modes described in the following sections.

Changing PDCM or the UDCM when user mode is selected may set or reset any of the channel flags, or issue captures and service requests, so it is advisable to switch channel modes only in a quiescent channel state, with channel flags MRLEA/B, TDLA/B, MRLA/B cleared. Furthermore, an event flag asserted in one mode may keep asserted after the mode programming change, even if the flag is impossible to be set in the new mode.

Table 43-345. MSR signals - Match Service Requests

value	MSR
00	issue no Service Requests on Matches
01	issue Service Request on Match B only
10	issue Service Request on 2nd[1] Match
11	issue Service Request on both Matches

1. 2nd Match means the Match that happens after the 1st Match in time, either Match A or Match B.

Table 43-346. TCAP and TSR signals - Transition Captures and Service Requests

value	TCAP	TSR
0	1st[1, 1] Transition captures Time Bases corresponding to Transition A and Transition B[2] Transition A captures corresponding Time Base if it is the second transition; Transition B captures corresponding Time Base if it is the second transition.	issue Service Request on the 1st[1] Transition
1	Transition A captures corresponding Time Base. Transition B captures corresponding Time Base.	issue Service Request on the 2nd[3] Transition

1. 1st Transition means the Transition that happens first in time, either Transition A or Transition B.
2. Match capture(s) never overwrites a Transition capture. Transition captures can always overwrite a Match capture.
3. 2nd Transition means the Transition that happens second in time, either Transition A or Transition B

Table 43-347. MCAP signal - Match Capture

value	MCAP
0	Match A captures corresponding Time Base; Match B captures corresponding Time Base
1	either Match captures Time Bases corresponding to Match A and Match B[1]

1. Match capture(s) never overwrites a Transition capture. Transition captures can always overwrite a Match capture.

Table 43-348. TBM2 signal - Transition Blocks Match B

value	TBM2
0	Transition A Blocks Match B
1	Transition B Blocks Match B

Table 43-349. M1ET, M1EM2, M1BM2, M2BM1, M2BT signals

signal	active value meaning
M1ET (Match A Enable Transitions)	Transitions are initially blocked, and Match A enables Transitions
M1EM2 (Match A Enables Match B)	Match B is initially blocked[1, 1], and Match A enables Match B
M1BM2[2, 2] (Match A Blocks Match B)	Match B is initially enabled[1], and Match A blocks Match B[3, 3]
M2BM1[2] (Match B Blocks Match A)	Match A is initially enabled,and Match B blocks Match A[3]
M2BT (Match B Blocks Transitions)	Match B blocks Transitions

Table continues on the next page...

Table 43-349. M1ET, M1EM2, M1BM2, M2BM1, M2BT signals (continued)

signal	active value meaning
T1BM1 (Transition A Blocks Match A)	Transition A blocks Match A
T2BM1 (Transition B Blocks Match A)	Transition B blocks Match A
T1ET2 (Transition A Enables Transition B)	Transition B is initially blocked, and Transition A enables Transition B

1. The initial condition of M1EM2 prevails over M1BM2, while M1BM2 blocking prevails over M1EM2 enabling, so that Match B stays always blocked when both M1BM2 and M1EM2 are active. This combination is used in single-match modes (sm_*).
2. Blocking of one Match by the other is done through MRLEs.
3. Matches always block themselves by resetting their own MRLEs (Match A always blocks Match A, Match B always blocks Match B)

Table 43-350. Predefined Channel Mode control signals decoding

Predefined Mode	MSR	MCAP	M1ET	M1EM2	M1BM2	M2BM1	M2BT	T1BM1	T2BM1	TBM2[1]	T1ET2	TSR[1, 1]	TCAP[1]
em_nb_st	11	0	off	off	off	off	off	On	off	0	On	0	0
em_nb_dt	11	0	off	off	off	off	off	On	off	1	On	1	1
em_b_st	11	1	off	off	On	On	off	On	off	0	On	0	0
em_b_dt	11	1	off	off	On	On	off	On	off	1	On	1	1
bm_st	10	0	off	off	off	off	off	On	off	0	On	0	0
bm_dt[2, 2]	10	0	off	off	off	off	off	off	On	1	On	1	1
m2_st	01	0	On	off	off	On	off	On	off	0	On	0	0
m2_dt	01	0	On	off	off	On	off	On	off	1	On	1	1
m2_o_st	01	0	On	On	off	off	On	On	off	0	On	0	0
m2_o_dt	01	0	On	On	off	off	On	On	off	1	On	1	1
sm_st	11	1	off	On	On	On	off	On	off	0	On	0	0
sm_dt[2]	11	1	off	On	On	On	off	off	On	1	On	1	1
sm_st_e[3]	11	0	off	On	On	On	off	On	off	0	On	0	0

1. signals TSR, TCAP and TBM2 replace the signal DTM used in previous eTPU versions.
2. bm_dt and sm_dt are exceptions in the match blocking logic by transitions. See Both Match Request Modes (bm_st, bm_dt) and Single Match Modes (sm_st, sm_dt).
3. sm_st_e is an exception in the capture scheme. See Single Match Enhanced Mode (sm_st_e).

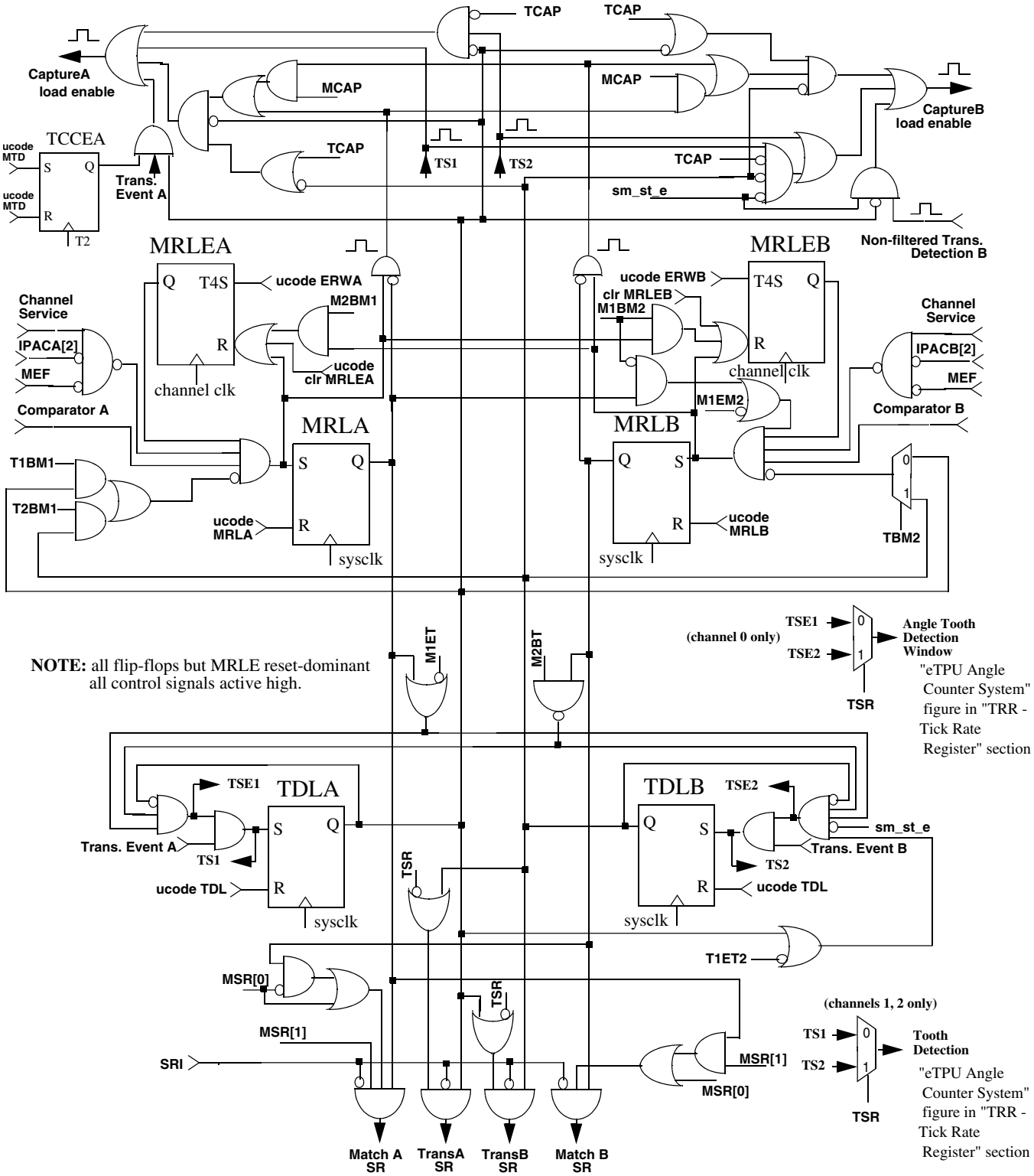


Figure 43-306. Channel Mode Logic and Event Flags

43.14.5.4.2 Channel Modes Overview

The predefined channel modes are divided, according to the way they treat transitions, in two basic modes:

- **Single Transition Modes** (mnemonic suffix **_st**): in these modes the first transition (flagged in TDLA) issues a service request, and captures both time bases (selected by TBSA[1] and TBSB[1]) except on sm_st_e. The second transition (flagged in TDLB) does not issue a service request, but it captures time base selected by TBSB[1], except on sm_st_e.
- **Double Transition Modes** (mnemonic suffix **_dt**): in these modes the second transition (flagged in TDLB) issues a service request, and each transition captures its own selected time base (Transition A and Transition B capture time bases selected by TBSA[1] and TBSB[1], respectively).

In predefined modes, Transition B is always (but not only) enabled by Transition A, so that transitions are always ordered: TDLA is set on the first transition and TDLB on the second. Unordered transitions are possible with user-defined mode, when UDCM bit T1ET2=0. Matches are generally not ordered, except on specific ordered match modes m2_o_st and m2_o_dt. Match capture(s) never overwrites a Transition capture, while Transition captures can always overwrite a Match capture, either in predefined or user-defined modes.

The following general rules apply to both predefined and user-defined modes:

- Blocking of one Match by the other, when it occurs, is done through MRLEs.
- Matches always block themselves by resetting their own MRLEs (Match A always blocks Match A, Match B always blocks Match B).

Predefined modes differ mostly by the way matches affects and are affected by other matches and transitions, as explained in next sections. However, some general rules on Match blocking apply:

- Match B is blocked by first transition (TDLA) in single transition modes, and by second transition (TDLB) in double transition modes.
- Both Matches are blocked by first transition in single transition modes.

Note

The rules above and in following sections may be overruled by the state of the channel latches if they are set/reset by microcode or if channel mode is changed. Care must be taken

to change channel modes, and is advisable to reset channel flags MRLA/B, TDLA/B and MRLEA/B before writing PDCM, or to UDCM when user-defined mode is selected.

43.14.5.4.2.1 Either Match, Blocking Modes (em_b_st, em_b_dt)

In these modes the first match recognition that occurs blocks the other match recognition and generates a service request. They end up with one service request for two programmed match recognitions where only the first match recognition has an actual effect. If both match recognitions occur at the same time, both MRLA and MRLB are set, before the mutual blocking takes effect. The associated logic is shown in the following figure.

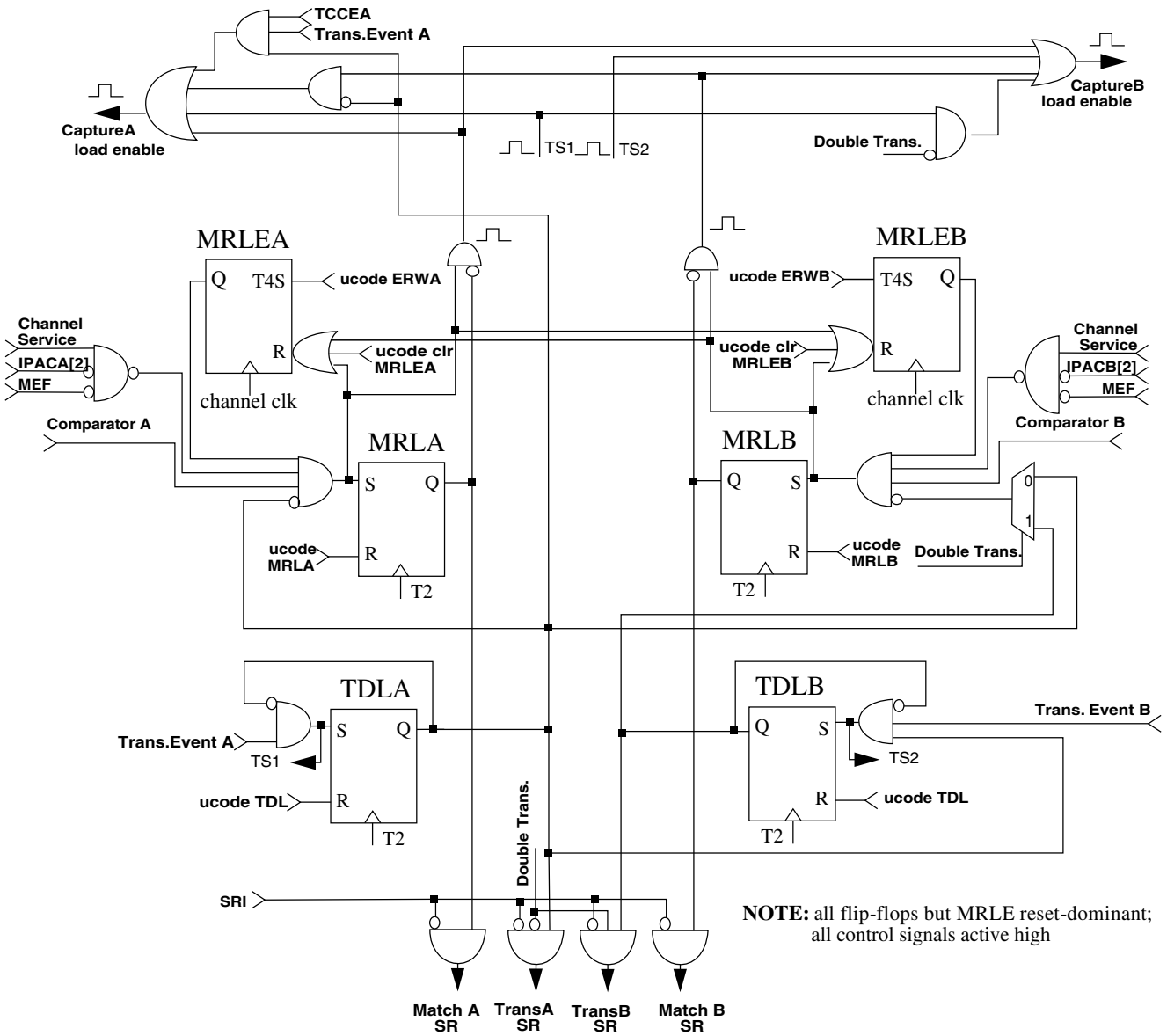


Figure 43-307. Either Match, Blocking Modes (em_b_st, em_b_dt)

43.14.5.4.2.2 Either Match, Non Blocking Modes (em_nb_st, em_nb_dt)

In these modes both match recognitions are independent and each of them generates service request. Each match recognition captures its related time base and does not block the other. The associated logic is shown in the following figure.

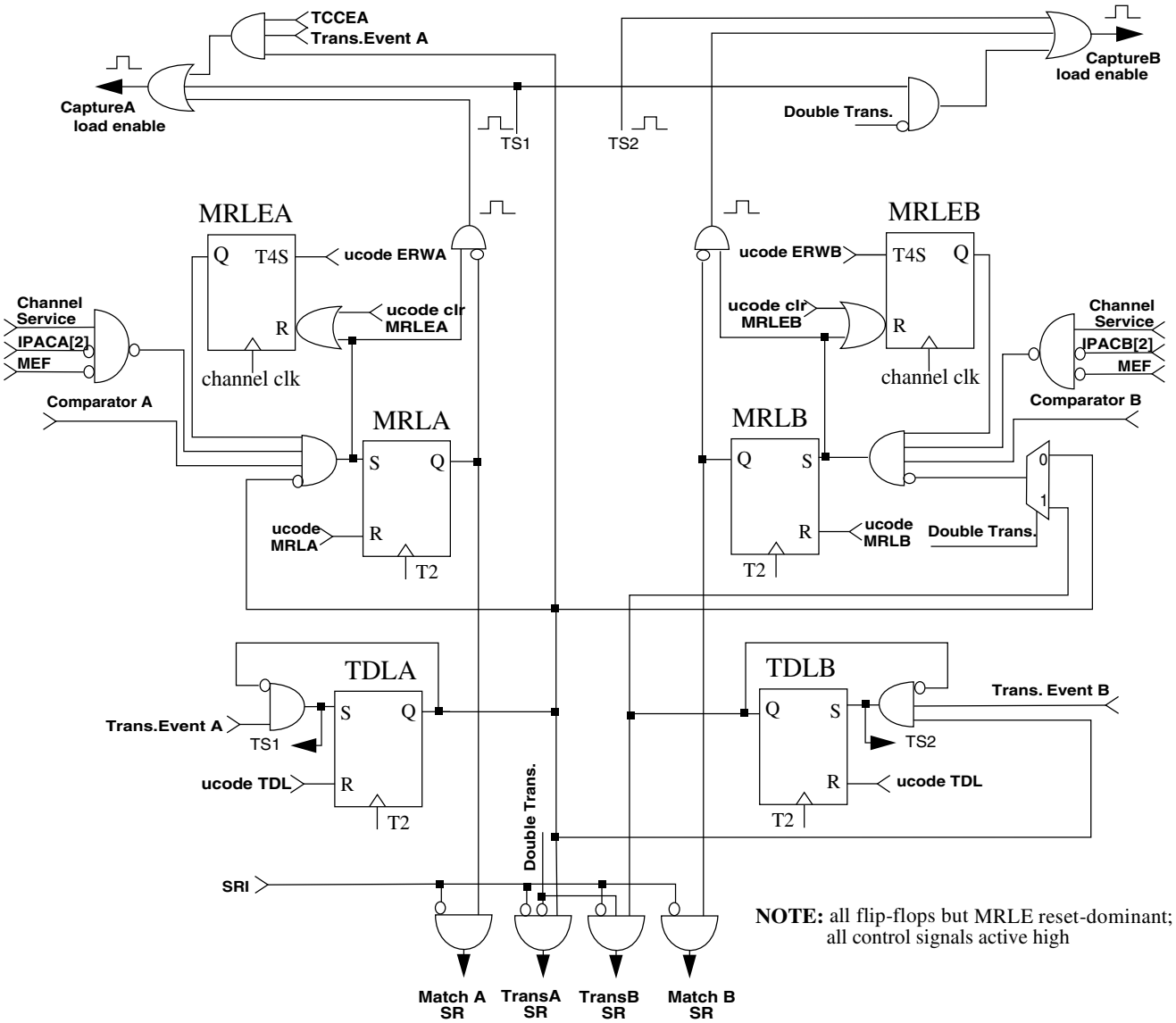


Figure 43-308. Either Match, Non Blocking Modes (em_nb_st, em_nb_dt)

43.14.5.4.2.3 Match B Request Modes (m2_st, m2_dt)

In these modes transitions are initially blocked, and are enabled by Match A. Match B recognition generates the match service request and disables Match A recognition. Each match recognition captures its own programmed timebase. In case of simultaneous match

recognition, both MRLA and MRLB are set, and OPACB register has priority over OPACA for selecting the pin action. The associated logic is shown in the following figure.

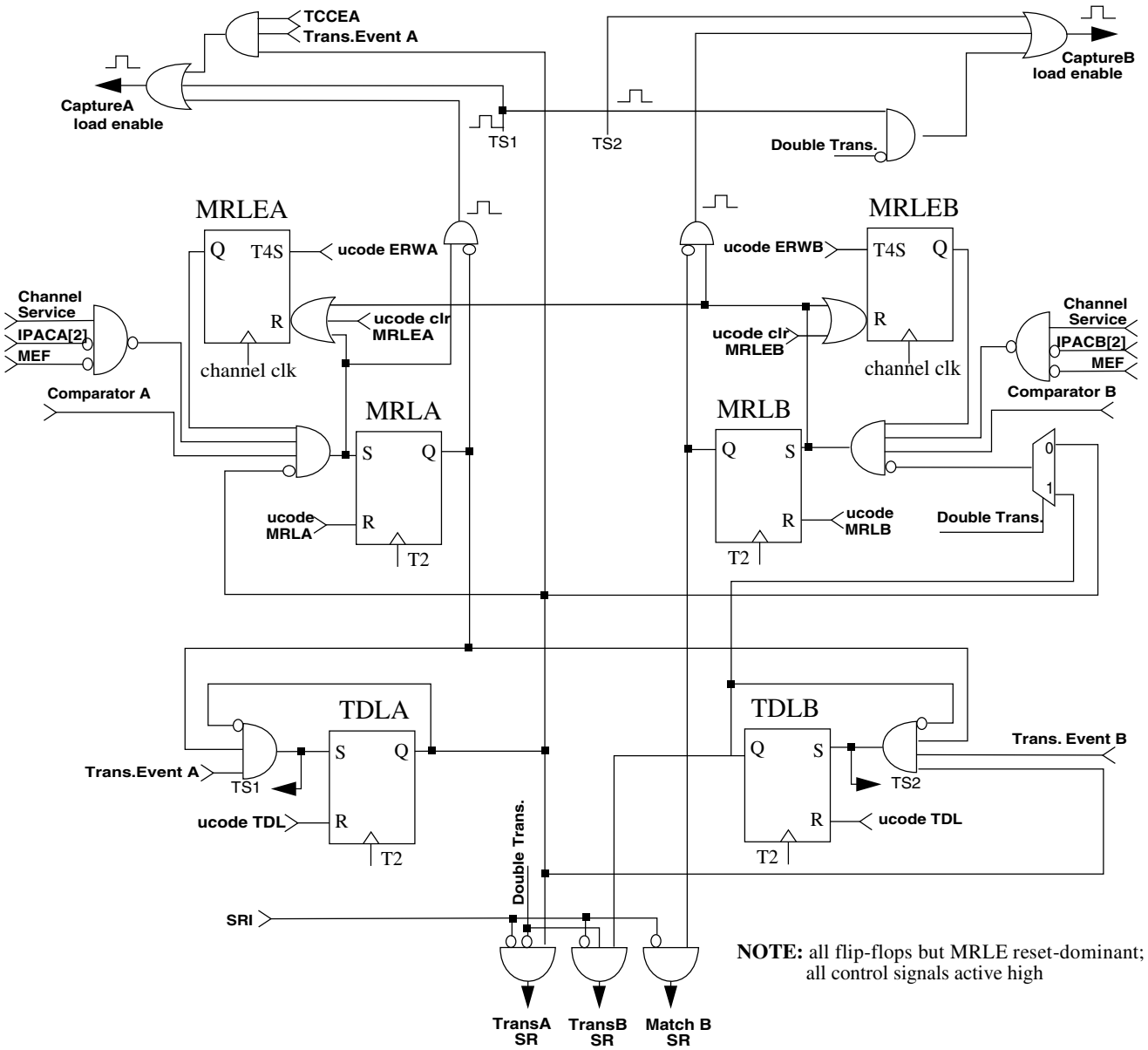


Figure 43-309. Match B Request Modes (m2_st, m2_dt)

43.14.5.4.2.4 Both Match Request Modes (bm_st, bm_dt)

In these modes, match service request is generated only after both match recognitions occurred. By definition this is a non-blocking match mode: match recognitions do not block each other, implementing a last-served scheme. Unlike other double transition

modes, bm_dt blocks Match A with Transition B (not with Transition A), so that the second transition blocks both matches. The associated logic is shown in the following figure.

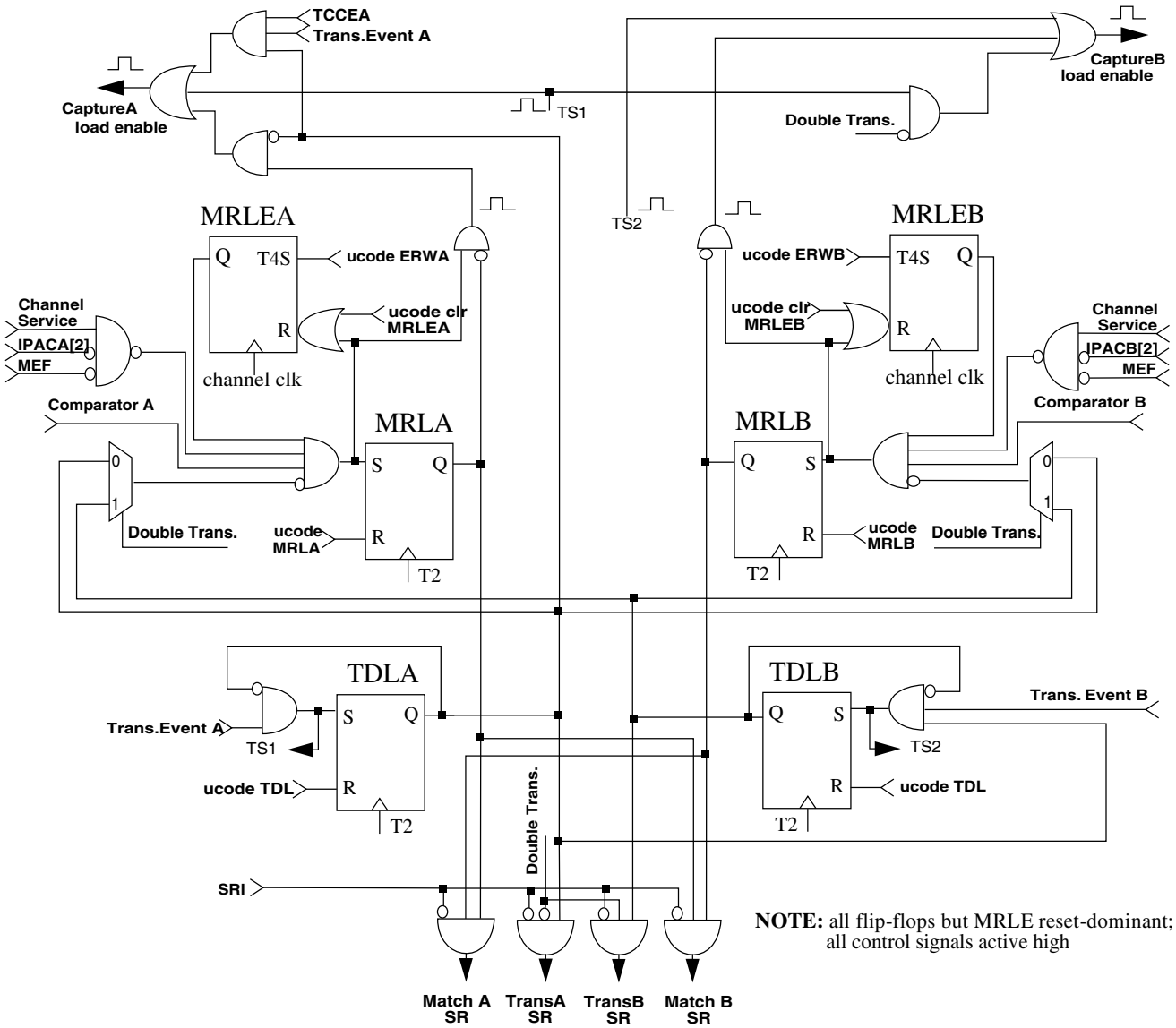


Figure 43-310. Both Match Request Modes (bm_st, bm_dt)

43.14.5.4.2.5 Ordered Modes with Match B Request (m2_o_st, m2_o_dt)

These are ordered match modes on which Match A recognition must precede Match B recognition (ordered 1->2). Match A asserts MRLA and enables Match B and transitions. Match B asserts MRLB, generates a match service request, and blocks both transitions. The associated logic is shown in the following figure.

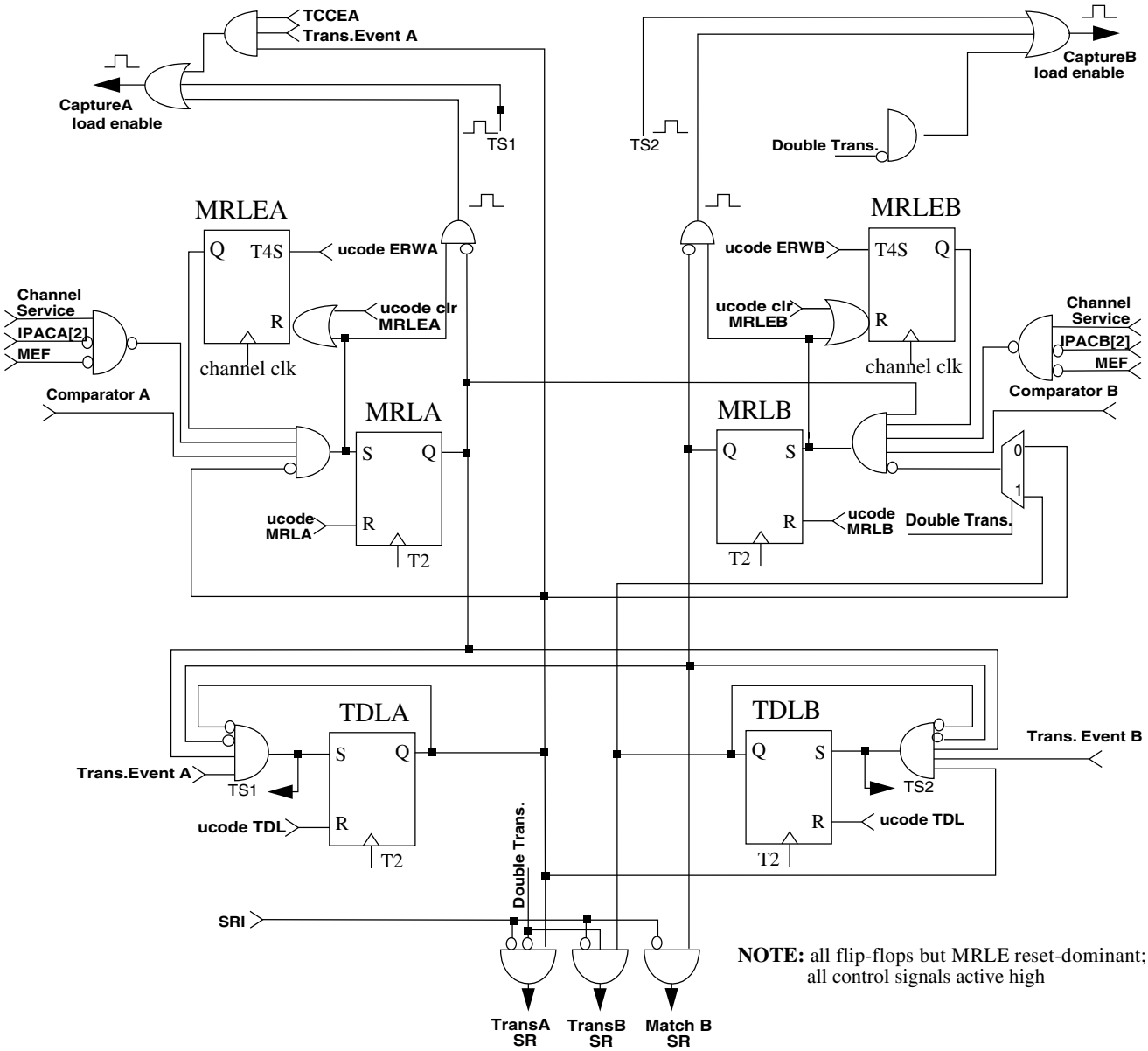


Figure 43-311. Ordered Modes with Match B Request (m2_o_st, m2_o_dt)

43.14.5.4.2.6 Single Match Modes (sm_st, sm_dt)

Single match modes support single or double transition with single match recognition. MRLB is never set, and MRLEB has no effect. The associated logic is shown in the following figure.

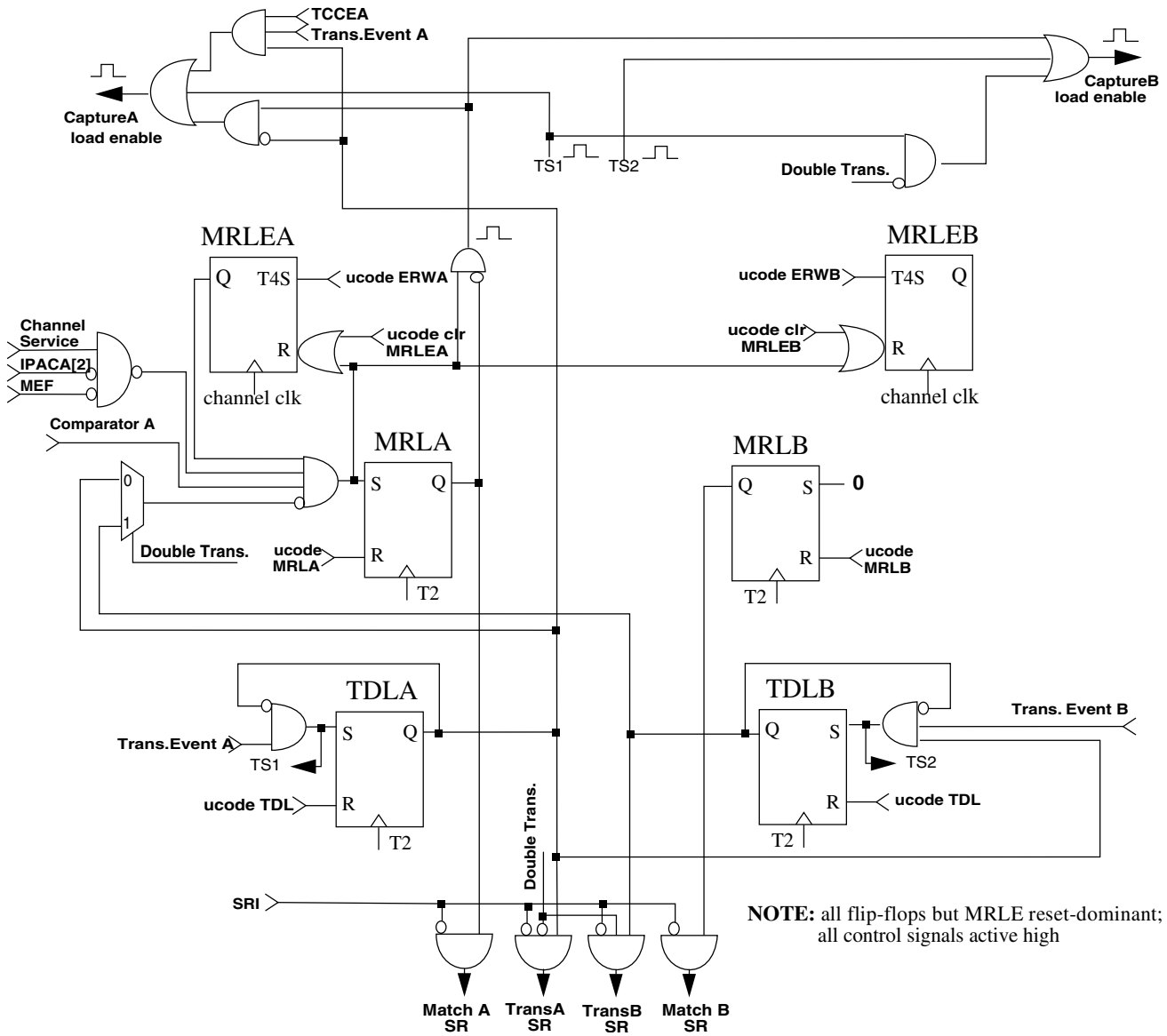


Figure 43-312. Single Match Modes (sm_st, sm_dt)

43.14.5.4.2.7 Single Match Enhanced Mode (sm_st_e)

This is an enhanced single transition and single capture mode, which provides non-filtered input captures in addition to the single capture, allowing one to measure the delay of the digital filter. In an output channel, it has the same functionality of sm_st (captures both time bases at once due to a match recognition). The associated logic is shown in the following figure.

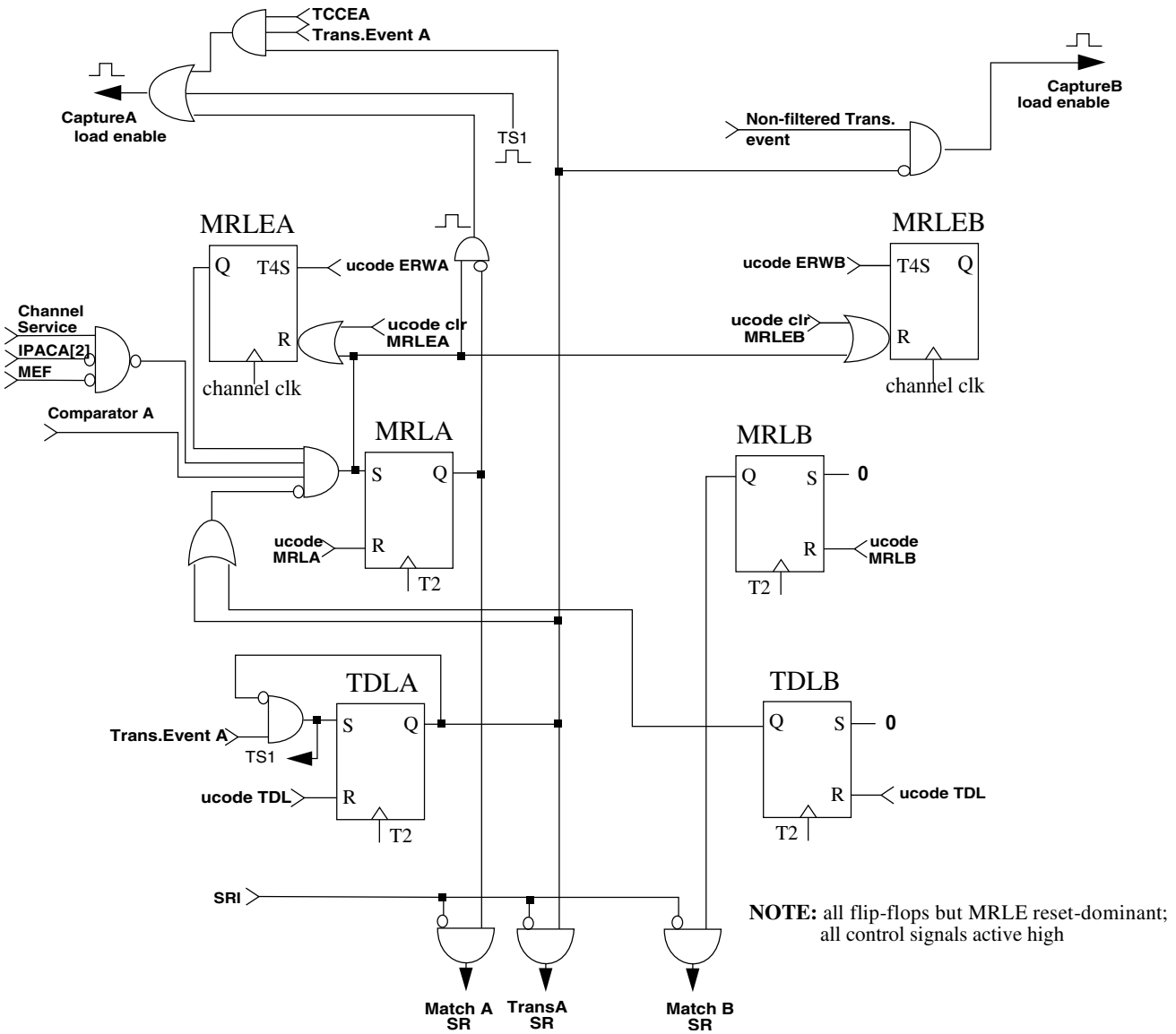


Figure 43-313. Single Match Enhanced Mode (sm_st_e)

43.14.5.4.3 Predefined Channel Modes on Input Signal Processing

When processing an input signal, the predefined channel modes can be classified in the following primary mode groups:

- Single Transition, Single Match: em_b_st, sm_st, sm_st_e
- Single Transition, Double Match: em_nb_st, bm_st, m2_st, m2_o_st
- Double Transition, Single Match: em_b_dt, sm_dt
- Double Transition, Double Match: em_nb_dt, bm_dt, m2_dt, m2_o_dt

In single transition modes, TDLA assertion may capture both time bases at once, while in double transition modes each transition captures its related time base in its related capture register.

Double transition is always ordered, i.e TDLB is enabled by TDLA and generates the service request.

The channel logic supports various input modes with combinations of single/double transition and single/double match, explained in the following subsections.

43.14.5.4.3.1 Either Match, Blocking, Single Transition (em_b_st)

On an input signal, this mode provides double timeout mechanism on a programmed transition edge with two timebases. The signal transition blocks both pending matches, indicating that no timeout condition occurred. The two match recognitions block each other, giving good separation in the entry table as to which match recognition caused the first timeout condition, and generating only one service request. Either match performs timebase captures which do not overwrite captures by transitions.

43.14.5.4.3.2 Either Match, Blocking, Double Transition (em_b_dt)

In double transition mode each transition is related to one match recognition. TDLA assertion captures its related timebase, blocks Match A and enables TDLB. TDLB assertion blocks Match B, captures its related timebase and generates a service request. Match recognitions block each other, so if there is a match timeout condition on TDLA, only one match service request is generated. This mode is good for qualifying two signal transitions by match timeout mechanisms, with one service request. Note that although TDLA assertion does not block Match B recognition, the value captured in CaptureA by TDLA assertion is not overwritten by this recognition. The second transition blocks Match B. Either match performs timebase captures which do not overwrite captures by transitions.

43.14.5.4.3.3 Either Match, Non Blocking, Single Transition (em_nb_st)

On an input signal, this is a double timeout mechanism of independent match recognitions of two different timebases. The match recognitions do not block each other, such that the microcode can check if one or two match recognitions occurred before their related signal transition. The signal transition detection (by IPACA) asserts TDLA, blocks both matches, captures both time bases and generates a transition service request, indicating that none of the two timeout conditions occurred. Any combination can be easily resolved by microcode (for example, signal transition after Match A and before Match B, or signal transition after both Match A and Match B).

Another possible use of this mode is allocating one match recognition for transition timeout and the other for another non-critical timed task, adding functionality to a single channel. Since the transition detection blocks both match recognitions, the match recognition of the other timed task is based on the fact that the comparator checks greater-equal conditions. It may be delayed if it occurs in the period between the signal transition detection (which blocks it) and the time TDLA is negated by microcode. If matches are enabled during the service, the same code can check if the match recognition of the timed task occurred in this period, by negating TDLA and writing to the CHAN register its own value (in order to update the MRLA flag in the branch logic).

43.14.5.4.3.4 Either Match, Non Blocking, Double Transition (em_nb_dt)

In this mode each transition is related to one match recognition, and the match recognitions are independent of each other. This mode can be used to give independent timeout conditions for the first and the second signal transition recognitions, and call service in any case of any timeout condition.

The first transition detection programmed in IPACA sets TDLA, captures its related timebase, blocks Match A recognition and enables TDLB assertion. The second transition detection programmed in IPACB sets TDLB, blocks Match B recognition, captures its related timebase and generates a service request. Any match recognition that occurs captures its related time base and generates a match service request, independent on the other match recognition.

43.14.5.4.3.5 Match B Request, Single Transition (m2_st)

On an input signal, this mode provides an open window filter for a single signal transition. MRLA assertion opens the window, and enables transition detection on TDLA from this time on. MRLB assertion blocks Match A (by negating MRLEA), providing conditional window opening, because transitions are indirectly blocked. It also generates service request, but if it happens after Match A it does not block transitions, providing a non-blocking timeout mechanism for the estimated signal transition time (typically it indicates a missing transition, or mis-prediction of the transition time).

Transitions can be detected from the microcycle following MRLA assertion. The Transition A detection asserts TDLA, blocks both matches, captures both timebases and generates service request.

Using this mode, the channel can replace software open window filtering of qualified transitions with the channel hardware window. The window opening and timeout can be scheduled for any of the two time bases or combination of them. Typically, Match A will be used to open a prediction window, and Match B will be used as a timeout condition

which does not close the prediction window. This configuration improves noise immunity from early signal transitions, and reduces the probability for blocking late signal transitions due to timeout mis-prediction.

Using these conditions, the microcode can easily resolve the state:

- If TDLA and MRLA are asserted and MRLB negated, signal transition is in the expected range.
- If MRLA and MRLB are both asserted, and TDLA is asserted, the signal transition had a timeout condition due to Match B mis-prediction.
- If MRLB is asserted and TDLA negated, a timeout condition occurred, and the expected signal transition had not occurred yet.
- If MRLA is negated and MRLB is asserted, the conditional window did not open at all (for example: a time window is open only after a specific angle, otherwise it is not opened).

43.14.5.4.3.6 Match B Request, Double Transition (m2_dt)

This mode is used as an open window filter for two signal transitions. In this case the Match A recognition opens the window (unless Match B recognition occurred first), and Match B recognition blocks Match A and generates a match service request. It is similar to m2_st, but in this case, it is the second transition that blocks Match B. MRLB assertion is a global timeout condition for the two pulses. Like m2_st, MRLB can conditionally eliminate the window from opening.

Using the TDLA, TDLB, MRLA and MRLB conditions, the microcode can easily resolve the state, in a similar manner as m2_st, with additional information on the second transition (TDLB).

43.14.5.4.3.7 Both Match Request, Single Transition (bm_st)

On an input signal, this is a double timeout mechanism on two different time bases. Both match recognitions must occur before the signal transition to generate a match timeout service request. Assertion of TDLA blocks both Match A and Match B recognitions, and captures both time bases, indicating there was no double timeout condition from both time bases.

Using the same timebase implements two timeout conditions, the first only sets its related MRL and the second generates a service request. Using these flags allows the microcode to check if one or both match recognitions precede the signal transition.

43.14.5.4.3.8 Both Match Request, Double Transition (bm_dt)

In this mode the first transition detection does not block matches, since both match recognitions are required to generate a match service request. The second transition detection asserts TDLB, blocks Match A and Match B, captures its related timebase and generates transition service request. In this mode, a Match A recognition which occurs after the assertion of TDLA does not capture a new value in CaptureA, to preserve the actual signal transition time. Assertion of TDLA, however, always captures its related timebase.

This mode allows putting a double match timeout condition on the second transition. Typically, a pulse trailing edge timing can be checked against two time bases, to indicate if the pulse has not ended when both MRLA and MRLB are asserted. When a transition service request is generated by TDLB assertion, the state of MRLA and MRLB indicates which timeout condition occurred, if any.

43.14.5.4.3.9 Ordered Mode with Match B Request, Single Transition (m2_o_st)

On an input channel, this mode provides a closing window filter for a single signal transition. Match A assertion captures its programmed time base in CaptureA, opens the filter window (enables assertion of TDLA), and enables assertion of MRLB. Match B recognition captures its related timebase, closes the window (disables assertion of TDLA) and generates a service request. Due to Match A and Match B ordering, the window is opened for at least one microcycle. Match B recognition indicates a window timeout condition which blocks late signal transitions, outside the prediction window. Transition detection blocks both matches, indicating the transition occurred inside the estimated window. Transitions can be detected from the microcycle following MRLA assertion until the microcycle on which MRLB is asserted. When TDLA is asserted inside the window range it disables both matches, captures both time bases and generates a transition service request.

Using this mode, the channel can replace software window filtering of qualified transitions with the channel hardware window. The window opening and closing time can be scheduled for any of the two time bases or a combination of them.

43.14.5.4.3.10 Ordered Mode with Match B Request, Double Transition (m2_o_dt)

In this mode the channel logic implements a window filter for two detected signal transitions. MRLA assertion captures its related timebase and enables the assertion of both TDLA and TDLB. MRLB assertion captures its related timebase and disables the assertion of both TDLA and TDLB. Transitions can be detected from the microcycle following MRLA assertion until the microcycle on which MRLB is asserted. The first

signal transition (following MRLA assertion) asserts TDLA, captures its related timebase and enables assertion of TDLB. The second signal transition detection asserts TDLB, blocks Match B, captures its related timebase and generates the service request.

If both signal transitions occur inside the scheduled window, Match B recognition is blocked. If one or both signal transitions do not occur inside the scheduled window, Match B recognition generates a match service request and blocks further transition detections. The microcode can resolve the state using MRLA, MRLB, TDLA and TDLB, which affect the entry point selection.

43.14.5.4.3.11 Single Match Enhanced Mode (sm_st_e)

This is an enhanced single transition and single match channel mode which provides timing information of the digital filter delay.

The CaptureA register captures the timebase selected by TBSA due to transition detection specified by IPACA or match recognition, as in sm_st mode. Initially, the CaptureB register continuously captures the unfiltered IPACB-selected signal transitions from the digital filter input, directly from the signal synchronizer. When an IPACA-qualified, filtered transition detection occurs, TDLA is set, MRLA assertion is blocked, and, in addition, captures into CaptureB are also blocked. On service, CaptureA and CaptureB (copied into ERTA and ERTB) holds the time of the qualified transition detection (ERTA), and the time of the last signal transition at the input of the digital filter (ERTB). Subtracting the time in ERTB from the time in ERTA provides the delay of the digital filter.

In a quiet environment, the two captures provide the accurate delay of the digital filter in granularity of two eTPU clocks. In a noisy environment, false transitions may be detected at the input of the digital filter due to the noise, and the delay measurement may be reduced, especially if IPACB selects both edge detection. The microcode can do sanity checks on this value to recognize noise effects (for example: calculated delay is less than the minimum delay of the digital filter).

Note

In Channel 0, if ETPUTBCR field AM=01 (Angle Mode), the unfiltered input comes from TCRCLK input and the filtered input comes from the TCRCLK filter output. The edge is selected by IPACA/B, and is independent of the edge selection by ETPUTBCR field TCR2CTL.

43.14.5.4.3.12 Single Match, Single Transition (sm_st)

In this mode the channel logic is functionally back-compatible to a TPU3 single action channel, but a match or transition detection captures at once both timebases. The mode recognizes a single transition with single match timeout. Either TDLA or MRLA generates service request and captures both timebases. Assertion of TDLA blocks future assertions of MRLA.

43.14.5.4.3.13 Single Match, Double Transition (sm_dt)

In this mode, the first transition detection asserts TDLA, captures a timebase in CaptureA and enables TDLB. The second signal transition asserts TDLB, blocks Match A, captures a timebase in CaptureB and generates a service request.

Match A (before TDLB) captures into CaptureB the timebase selected by TBSA, in order not to overwrite the captured value of TDLA.

This mode is used for scheduling one timeout condition on two input signal transitions (pulse timeout).

43.14.5.4.4 Channel Modes on Output Signal Generation

Since channel logic can generate output signal transitions based on Matches, the channel can be viewed as working in the following primary mode groups for signal generation:

- Single Match: em_b_st, sm_st, sm_st_e, em_b_dt, sm_dt
- Double Match: em_nb_st, bm_st, m2_st, m2_o_st, em_nb_dt, bm_dt, m2_dt, m2_o_dt

The channel logic supports various match channel modes with single/double match, as explained in the following subsections.

43.14.5.4.4.1 Either Match, Blocking Modes (em_b_st, em_b_dt)

On an output signal these modes are useful when using two different time bases to set a required signal transition. The first match condition which is met sets a required pin action, captures both time bases, blocks any effects of the other recognition, and generates a service request. Because the first match recognition blocks the other, the microcode can get good separation in the function entry table as to which match caused the timeout first, and both time bases are captured, enabling the microcode to compare one timebase to the other at the moment of the match recognition. These modes can be used for:

- Scheduling a required pin action to the first match recognition of two different time bases.
- Canceling a programmed pin action scheduled on one time base by match on another timebase (as a consequence of Table 43-351). Microcode has to set the OPAC register of the canceling match to no-action and the OPAC register of the other match to the required pin action which may be blocked. If Match A is the canceling match, it blocks the pin action also in case of two matches at the same time, since it has priority in this case. If Match B is the canceling match, it does not block the pin action in case of two matches at the same time.

43.14.5.4.4.2 Either Match, Non Blocking Modes (em_nb_st, em_nb_dt)

On an output signal these modes are useful in combination with the ME bit set on the entry point, to define an interlaced operation. For example, each match recognition can set a pin action, and the second pin action is not sensitive to microcode latency (ME bit asserted). Example for usage is PWM interlaced function on which the latency is determined by the period and not the duty cycle.

Another possibility is using one match for pin actions and the other match for an unrelated timed task without pin action (double the functionality of a single channel).

43.14.5.4.4.3 Match B Request Modes (m2_st, m2_dt)

On an output signal, these modes can generate narrow pulses or do conditional pin actions. A conditional pin action means that the pin state is changed only if the match recognitions occurred in the correct order, since the Match B recognition which generates the service request also has priority over the pin action and blocks future Match A recognitions.

Setting OPACA to a desired pin action and OPACB to no-action, and using different time bases for Match A and Match B defines a conditional OPACA pin action which can be blocked by Match B recognition. For example, setting Match A on time and Match B on angle can limit the pin action to a maximum angle value.

When pulses are generated, the service is requested at the trailing edge of the pulse, after MRLB is asserted.

43.14.5.4.4.4 Both Match Request Modes (bm_st, bm_dt)

On an output signal, each match recognition can affect the pin state, and capture its programmed time base. This way the pin action can be programmed separately for both match recognitions. For example, both match recognitions can negate the signal, and

service request is generated after both conditions are met. This mechanism can set two conditions to do a required pin action, and the first recognition changes the signal, but service is called only after both conditions occur.

When using the same time base, these modes can generate narrow pulses in any required order. For example, in a PWM function, when duty cycle is below 50% the function can get service on the low time and program the pulse to the required duty cycle of the high time. When duty cycle is equal or above 50%, the function can get service on the high time and program a negative pulse with the width of the required low time. To switch between the two states the function can program once the same transition time to MatchA and Match B with a required pin action, and on the next service program double match for the new state.

Another usage is generating a required pin action on one programmed time and service request later on another time, after the second match recognition occurs, or capturing some timebase on one time and generating a required signal transition and service request later.

43.14.5.4.4.5 Ordered Modes with Match B Request (m2_o_st, m2_o_dt)

The order of the match recognitions imply that OPACA register programmed pin action always precede the OPACB register pin action. Setting OPACA to no-action, based on the greater-equal comparator, enables using Match A on one time base to delay the signal effect of Match B on the other time base. This method implements a conditional pulse extension or conditional delay on signal transition.

These modes can also be used for deferred pulse generation with microcode service request after its trailing edge (if Match A condition comes after Match B condition). Another option is having Match A recognition associated with output pin actions and Match B recognition for a timed microcode task which has to be scheduled at a programmed time which may be delayed by the Match A pin action.

43.14.5.4.4.6 Single Match Modes (sm_st, sm_dt, sm_st_e)

There is no difference between plain and enhanced Single Match modes on an output signal.

In this mode the channel logic is functionally back-compatible to a TPU3 single match output channel. Match A recognition generates service request and sets the pin state according to OPACA register. It captures at once the timebase selected by TBSA in CaptureA and the timebase selected by TBSB in CaptureB.

43.14.5.4.4.7 Match/Transition Pin Action Conflict Resolution

In output signals, matches and/or transitions automatically cause pin actions defined by the OPACA/B and/or IPACA/B channel control registers (see Pin Control Registers). Simultaneous matches/transitions may be associated with different, possibly contradictory, pin actions. These conflicts are resolved according to the table below.

If an OPACA/B=000 (no action) prevails over non-zero OPAC according to the table below, then if Match A/Transition A and Match B/Transition B occur simultaneously, no output pin action occurs, that is: a match on the action logic with OPAC=000 inhibits simultaneous actions of the other OPAC, if prevailing according to the following table. That also applies when output actions are caused by inputs (OPAC=1xx).

Table 43-351. Simultaneous Match Pin Action Priority

Channel Mode	Priority
em_nb_st / em_nb_dt	OPACA
em_b_st / em_b_dt	OPACA
bm_st / bm_dt	OPACA
m2_st / m2_dt	OPACB
m2_o_st / m2_o_dt	in these modes there is no possibility of simultaneous matches
user-defined	OPACB if M2BM1 = 1 and M1BM2 = 0, OPACA otherwise

43.14.5.4.5 Combining Input and Output Signals

The processing of input signal can be combined with output signal generation. A detected input transition can trigger an output signal edge, even without microcode intervention, by using OPAC options 1xx.

The channel set-up examples below show these two capabilities combined (see the figure below).

The **first example** implements a fast (no microcode intervention) short-circuit protection feedback mechanism for driving high-current output devices. The signal after the high-current driver feeds back to the channel input. The input signal is normally delayed from the output signal by the device turn-on delay. After the channel output turns on, the channel logic must check if the driver output (connected to the channel input) follows the driven value after the maximum device turn-on delay. If it does not, the driver output is probably shorted, and the channel output must be turned off immediately to avoid damaging the device.

Match A causes the output to be driven high (for simplicity the output and input signals are shown as positive logic). It also causes a transition A, because IPACA=100 and the input is still low. Match B occurs after the expected driver delay, and causes a service request. If the output is shorted, a Transition B occurs on Match B because IPACB=100. This will cause the output to go low immediately, because OPACB=100.

In the **second example**, an output pulse is generated from an input transition without microcode intervention. Match A opens a window for transitions and also enables Match B. A rising edge on input sets output high. On Match B the window closes, and input signal is checked: if sampled high, the output resets; otherwise it stays high.

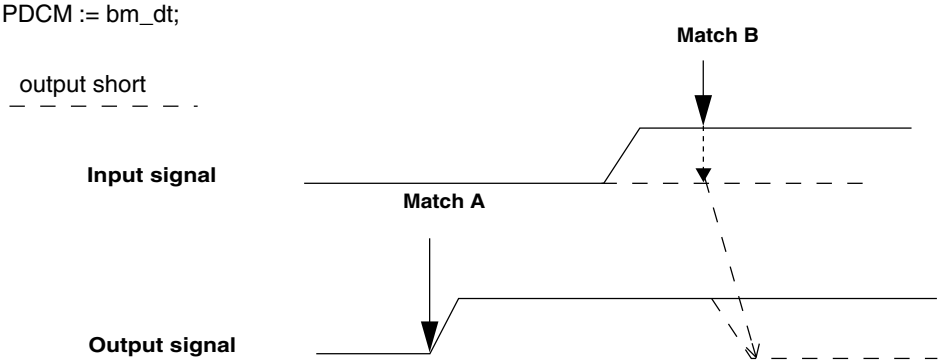
In the **third example**, a pulse is generated depending on the value sampled on the input signal at a predetermined time. Match A samples the input signal, causing a Transition. Low level on input sets output low, otherwise it stays high. Match B sets output high. In both cases a service request is issued (microcode intervention), at the beginning and at the end of the pulse (Match B), if required (SRI := 0).

Note

When IPAC = 1xx, a match event can cause simultaneously a Match recognition and a Transition detection. Depending on the Channel Mode, these Match and Transition may have conflicting effects on other Transition/Match blocking or enabling. In these cases, blocking always prevails over enabling, effective on the next microcycle.

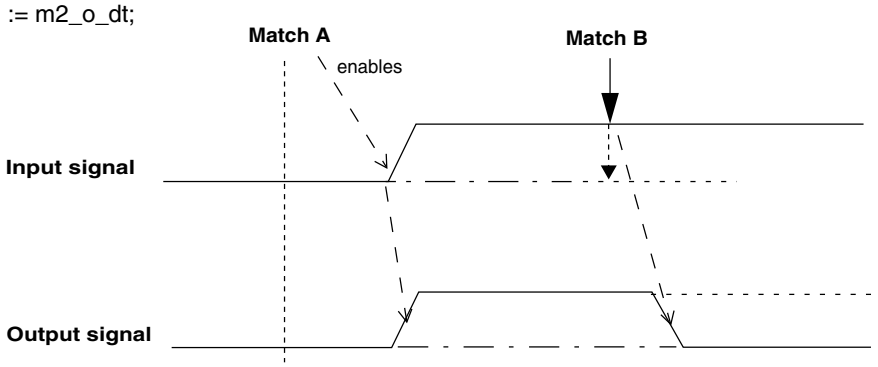
Example 1: Short-circuit protection feedback

IPACA := 100; OPACA := 001; MatchA := output activate time;
IPACB := 100; OPACB := 100; MatchB := MatchA + max. high-current driver turn-on delay;
PDCM := bm_dt;



Example 2: Pulse generation on windowed input transition

IPACA := 001; OPACA := 101; MatchA := window open time;
IPACB := 101; OPACB := 100; MatchB := window close time, input sampling;
PDCM := m2_o_dt;



Example 3: Pulse generation on input sampling

IPACA := 100; OPACA := 100; MatchA := window open time,input sampling;
IPACB := 000; OPACB := 001; MatchB := window close time = MatchA + pulse width;
PDCM := em_nb_dt;

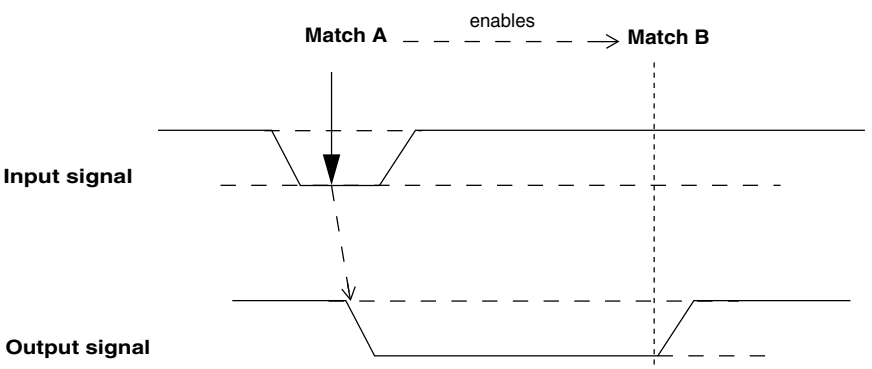


Figure 43-314. Input/Output combination

43.14.5.5 Channel Link

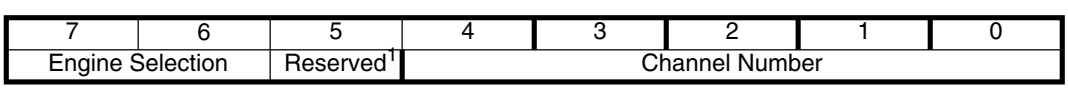
A channel can issue service requests to other channels through microcode, by assigning to the write-only microengine register LINK (refer to LINK Register) a value which specifies the target channel of the Link Service Request, as shown in the figure below.

Writing to the LINK register issues a link request to the target channel, setting its LSR flag. Each channel has its own LSR flag, which can be tested as a microcode branch condition (see Conditional/Unconditional Branch) and reset through the microcode field LSR (see Clear Link Service Request). The link branch condition samples, at the TST start, the value used to calculate the Entry Point.

Writing LINK with another channel target value in the same thread issues a Link Service Request to the new target, without negating the service request to the former one. This allows a channel to issue service requests to any number of channels, including itself. Neither LINK nor LSR microengine accesses are qualified by the CHAN register, i.e., they always access the serviced channel LINK and LSR, regardless of the value written in CHAN.

Channels can also receive link service requests from the Shared Subsystem Units (see Shared Subsystems Support). These links are set by the interrupt signals from the shared subsystem and are level-sensitive, that is, they keep on requesting link service as long as the interrupt is active. Therefore, it is not enough to clear the link service request through the LSR instruction field: one must clear beforehand the respective shared subsystem unit interrupt source.

If microcode executes an instruction with field LSR=0 (clear Link Service Request), the link branch condition is cleared. However, the link service request itself is cleared only if no link from other channel was received by the serviced channel during the same thread[26] . Note that this applies only to links that come from the other engine or the same channel: if only Shared Subsystem link was received during the thread, the link service is also cleared with field LSR=0. If microengine clears LSR of its channel and, simultaneously, Link Service Request is issued to the current serviced channel, the branch condition is cleared but the link service request remains pending.



NOTES:
1 reserved bit must be written 0.

Figure 43-315. Microengine LINK Register

26. That can only happen if the link service request came from the other Engine, or from the serviced channel itself.

A channel can issue Link Service Requests to channels in any of two Engines, determined by the LINK register field Engine Selection (2 bits), as shown in the following table. In a Single Engine eTPU, Link is ignored when sent to the other engine, or engine 2.

Table 43-352. LINK Engine Selection

Engine Selection	Description
00	this Engine
01	Engine 1
10[1, 1]	Engine 2
11[1]	the other Engine

1. ignored in Single Engine eTPU.

There is no explicit hardware mechanism to allow the thread to identify the link service request source, but it can be done by a software protocol using SDM parameters. All links are negated on reset.

43.14.5.6 Enhanced Digital Filter - EDF

The EDF eliminates passing of signal transitions which are caused by noise. Its purpose is to eliminate false transition service requests caused by noise pulses which are shorter than a programmed width.

See the chip-specific eTPU information for connections between eTPU and input glitch filters.

The EDF has three modes of operations, selected by the CDFC field in the ETPUECR register (see the ETPUECR register). These modes offer selections of trade-off between noise immunity and signal latency. CDFC also allows the filter to be bypassed. Table 43-353 gives an example of minimum detected signal pulse and maximum filtered noise pulse in the three EDF operation modes. In Angle Mode, if AM=01, the EDF in channel 0 is replaced with the digital filter and synchronizer of the TCRCLK signal. In this mode, channel 0 works in combination with the Angle Counter logic, and their operation is fully synchronized.

Following subsections provide the functional description of the eTPU channel digital filter.

43.14.5.6.1 Two-Sample Mode

In this mode the EDF works like the TPU2/3 digital filter. It uses the filter clock which is the eTPU clock divided by (2, 4, 8,.., 256) as a sampling clock. The filter clock is selected by the FPSCK field in the ETPUECR - Engine Configuration Register (see the ETPUECR register). The EDF compares two consecutive samples. If both samples have the same value, the input signal state is updated. Note that when the FPSCK field selects the eTPU clock divided by two, the EDF works like the TPU1 four-clock digital filter.

43.14.5.6.2 Three-Sample Mode

In this mode, like in the TPU2/3 mode, the EDF uses the filter clock as a sampling clock. The EDF compares three consecutive samples. If all three samples have the same value, the input signal state is updated.

The Three-Sample mode gives more signal latency than the Two-Sample mode, but also better noise immunity and better ratio between minimum detected signal pulse to maximum filtered noise pulse. When a certain filter clock frequency is selected for Two-sample mode, double filter clock frequency can be selected to get better latency in Three-sample mode.

43.14.5.6.3 Continuous Mode

In this mode the EDF compares all the values sampled at the rate of eTPU clock divided by two, between two consecutive filter clock pulses. If the signal is continuously stable for the entire digital filter clock period (i.e all the samples have the same signal value), the input signal state is updated.

This method gives the same latency and the same ratio between minimum detected signal pulse to maximum filtered noise pulse, as the Two-Sample mode, as long as there is no noise. Each sampled noise delays the signal transition detection by at least a whole digital filter clock period.

The Continuous mode gives the best noise immunity by comparing multiple samples of the noise. On the other hand, when a short noise pulse appears in the middle of the filter clock period at the same time of a real signal transition, the Continuous mode may reject a real signal transition and delay the response to the first filter clock period in which the signal is continuously stable. This may add to the latency and also to the minimum detected signal pulse in a noisy environment.

43.14.5.6.4 Bypass Mode

In bypass mode the signal that feeds the edge detection comes directly from the output of the synchronizer, not filtered. Bypass mode automatically makes the channel logic work in T2/T4 timing mode (see Channel Timing Modes).

43.14.5.6.5 Filter Clock Prescaler

Each channel configured as an input has an associated synchronizer followed by a digital filter connected to the signal that samples signal transitions. After reset, the digital filter filters out high and low pulse widths smaller than the period of two eTPU clocks with ETPUECR bit FCSS=0, or 1 eTPU clock with FCSS=1, preventing these transitions from being input to the transition detect logic. For FPSCK=0 and FCSS=0, the synchronizer and digital filter are guaranteed to pass pulses that are as wide as or wider than four eTPU clocks, meaning a minimum period of 8 eTPU clocks. These figures are halved by setting FCSS=1. By changing the FPSCK field in register ETPUECR the user can select a lower clock rate for the filter signal to define wider valid pulses and filter out wider noise pulses. The filter prescaler clock control is a division of the eTPU clock. To guarantee pulse detection by the digital filter, the pulse must cover at least the stated number of samples at the filter clock rate. For example, a two sample digital filter must sample two points in the pulse to detect it. The table below shows the minimum guaranteed detected pulse width and the maximum filtered noise pulse width. The table refers only to the digital filter operation. The external pulses may have to be wider (to ensure detection) or narrower (to ensure filtering) depending on the rise/fall delay differences in the MCU receivers and internal logic. Delays introduced by synchronizer, filter and edge detection logic are explained in Input/Output Signal Delays.

The TCRCLK signal also has a synchronizer and a filter. The FCSS bit has no influence on the clock prescaler of the TCRCLK filter, which works like the channel filter with FCSS=0.

Table 43-353. Pulse Widths and Delays

Filter Control (FPSCK)		Filter Clock Period[1]	Min. Width Guaranteed Detected / Max. Width filtered (Min. Filter Delay / Max. Filter Delay)[1, 1]	
FCSS = 0	FCSS = 1		Two-Sample or Continuous Mode	Three-Sample or Integrator[2] Mode
not avail.	000	1	2 / 1 (2 / 3)	3 / 2 (3 / 4)
0003	001	2	4 / 2 (3 / 3)	6 / 4 (5 / 5)
001	010	4	8 / 4 (5 / 7)	12 / 8 (9 / 11)
010	011	8	16 / 8 (9 / 15)	24 / 16 (17 / 23)
011	100	16	32 / 16 (17 / 31)	48 / 32 (33 / 47)
100	101	32	64 / 32 (33 / 63)	96 / 64 (65 / 95)

Table continues on the next page...

Table 43-353. Pulse Widths and Delays (continued)

Filter Control (FPSCK)		Filter Clock Period[1]	Min. Width Guaranteed Detected / Max. Width filtered (Min. Filter Delay / Max. Filter Delay)[1, 1]	
FCSS = 0	FCSS = 1		Two-Sample or Continuous Mode	Three-Sample or Integrator[2] Mode
101	110	64	128 / 64 (65 / 127)	192 / 128 (129 / 191)
110	111	128	256 / 128 (129 / 255)	384 / 256 (257 / 383)
111	not avail.	256	512 / 256 (257 / 511)	768 / 512 (513 / 767)

1. This table shows pulse widths and delays in number of periods of the eTPU clock.
2. Integrator mode is available for TCRCLK filtering only, see TCRCLK Digital Filter."
3. These delays also apply to the TCRCLK filter when TCRCF selects eTPU clock/2 as filter clock.

Note

The TCRCLK filter is clocked as if FCSS=0, always dividing eTPU clock /2 using FPSCK, regardless of the FCSS field.

43.14.5.7 Channel Timing Modes

Channels can work on two different timing schemes, defining the period of channel clocking, tied to T2 and T4 microengine phases, as explained in subsections below. Microengine T2 and T4 phases are shown in Microcycle and I/O Timing.

43.14.5.7.1 T2 Channel Timing

In T2 timing mode the channel event state can only be updated every two eTPU clocks (see Figure 43-343). Pin state, TDLs, MRLs and Capture registers are updated on the microengine's T2 clock phase. MRLE clears also happen on T2, but MRLE setting occurs on T4, together with the Match register write by microcode (see Write Channel Match and UDCM Registers).

Channels work in T2 timing mode when either ETPUCxCR bit T4DIS=1 or all the following conditions are true:

- ETPUTBCR bit TCR1CS is 0 (see the ETPUTBCR register.
- the Enhanced Digital Filter is not configured as bypass (see Enhanced Digital Filter - EDF).
- ETPUECR bit FCSS is 0 (see Filter Clock Prescaler).

43.14.5.7.2 T2/T4 Channel Timing

In T2/T4 timing mode the channel event state can be updated on any eTPU clock (see Figure 43-344). Pin state, TDLs, MRLs, MRLEs, and Capture registers are updated either on microengine's T2 or T4 clock phases. MRLE clears can happen on T2 or T4, but MRLE setting occurs on T4 only, together with the Match register write by microcode (see Write Channel Match and UDCM Registers).

Channels work in T2/T4 timing mode when ETPUCxCR bit T4DIS=0 and either one the following conditions are true:

- ETPUTBCR bit TCR1CS is 1 (see the ETPUTBCR register.
- the Enhanced Digital Filter is configured as bypass (see Enhanced Digital Filter - EDF).
- ETPUECR bit FCSS is 1 (see Filter Clock Prescaler).

T2/T4 timing mode demands special care on channel microcode that reprograms a match in a thread with matches enabled (entry point ME=1, see the “Entry Points” section). When the match register is written and the MRLA/B flag(s) is(are) cleared in the same instruction, a match can occur after MRLA/B is cleared, together with the set of respective MRLE. In this case, both MRL and MRLE will be set, and a match service request will occur if enabled. However, the match happened on the old match value, not on the new (reprogrammed) one. In order to prevent this ambiguity to the code that services the match, it is advisable to clear the MRLE (besides MRL) together with the match reprogramming, avoiding the match on the old value to occur while the new match value is being written. The set of the MRLE due to match reprogramming prevails over the MRLE clear, thus allowing the new programmed match to occur in the future.

A similar case occurs when a match is blocked by a transition, the blocking TDL is cleared and the blocked match is reprogrammed at the same instruction: in T2/T4 timing mode: a match can occur on the old match value. The solution is the same as the case above, that is, clear the respective MRLE in the same instruction.

43.14.6 Time Bases

Each eTPU engine has two Time Counter Registers, TCR1 and TCR2. They provide 24-bit time bases, shared by all 32 channels. Any channel can use both time bases to:

- Match channel's internal registers MatchA or MatchB;
- Capture time base value to channel's internal registers, CaptureA and/or CaptureB, when a match recognition or an Input transition detection occurs. For more information on channel events refer to Enhanced Channels.

The TCR1 and TCR2 counters are accessible by the microcode for read and write operations. Its current value is used for getting the current time, and the captured values are used for channel relative time calculations of future events. When they are written at the same time they are incremented from any clock source, the written value prevails.

TCR1 with ETPUTBCR[TCR1CS]=0 and TCR2 values are updated in T2 and read in T4 (see Microcycle and I/O Timing). TCR1 can also work at full-speed eTPU clock, and so be updated on both T2 and T4, when ETPUTBCR[TCR1CS]=1. Both TCR1 and TCR2 values can be imported from or exported to the STAC bus. When their values are imported (STAC Clients), these registers are written from the STAC bus and can only be read by microcode.

The TCR2 counters between the two Engines are out of phase by 1 eTPU clock, even when Time Bases are shared between them through STAC . It also applies to TCR1 counters if ETPUTBCR[TCR1CS]=0, but they can be in phase otherwise.

43.14.6.1 Timer Count Register 1 - TCR1

TCR1 can be used in the following modes:

- Internally Clocked Mode
- Externally Clocked Mode
- STAC Bus Client Mode

The host program can read TCR1 time base through the ETPUTB1R (see the ETPUTB1R register).

The TCR1 bus runs through all the local engine channels. In channels which select TCR1 as MatchA and/or MatchB source, when its value is greater or equal to the programmed match value, a Match A and/or Match B event occurs on that channel. A recognized match event sets its related Match Recognition Latch 1 or 2, and according to the Predefined Channel Modes (PDCM) it may generate a channel service request. For details on eTPU channels refer to Enhanced Channels.

43.14.6.1.1 Externally Clocked Mode

TCR1 can be driven externally by the TCRCLK input, after the digital filter. The TCR1 clock source is configured by the TCR1CTL bit, as shown in the following figure. For more information on clock source selection, please refer to the ETPUTBCR register.

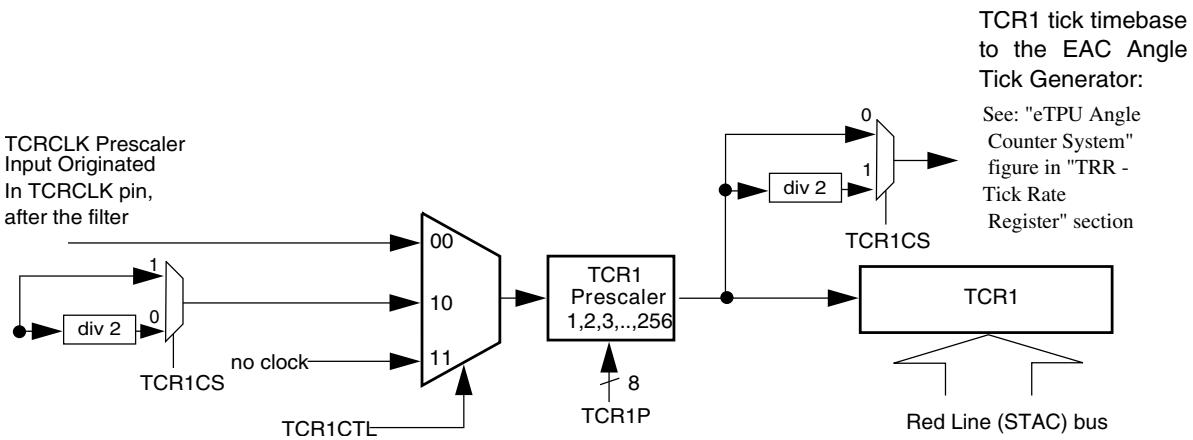


Figure 43-316. TCR1 Clock Selection

43.14.6.1.2 Internally Clocked Mode

TCR1 can be driven by the eTPU clock or eTPU clock divided by 2, before the prescaler. TCR1CTL can also be used to freeze TCR1 clock independently of TCR2 (unlike GTBE).

43.14.6.1.3 TCR1 Clock Prescaling

Any clock source selected by TCR1CTL is prescaled by a factor of 1 to 256, selected by ETPUTBCR field TCR1P. For more information on prescaler configuration refer to the ETPUTBCR register. The TCR1 Prescaler resets when the eTPU GTBE input is negated. After reset, it starts counting up to TCR1P when the eTPU GTBE input is asserted. When TCR1 increments (the eTPU GTBE input = 1), the prescaler starts a new count and the new TCR1P becomes effective. When TCR1 is written by microcode, the prescaler is reloaded with TCR1P and it becomes effective, if the eTPU GTBE input is asserted.

43.14.6.1.4 STAC Bus Client Mode

In this mode the TCR1 register is continuously updated from the STAC bus, and the clock selection and prescaling logic becomes ineffective. It is not writable by the microcode, and when read, it reflects the STAC bus imported value. The use of EAC is not allowed in client mode. This mode is configured through the ETPUREDCR register (see the ETPUREDCR register).

43.14.6.1.5 STAC Bus Server Mode

TCR1 bus can be exported to the STAC bus as a server, providing time information to other peripherals. This mode is configured through the register ETPUREDCR (see the ETPUREDCR register).

43.14.6.2 Timer Count Register 2 - TCR2

The TCR2 is a 24-bit counter which can be used in the following modes:

- Pin Transition Mode: Count the rise, fall or both transitions of TCRCLK signal.
- Angle Clock Mode: Count internal tooth angle in combination with the eTPU Angle Counter (EAC) hardware which implements an Angle PLL, and generates angle information to the channels. This mode is targeted for angle based applications.
- STAC (STAC) Bus Client Mode: TCR2 is driven by an external source (see STAC Bus Client Mode).
- Gated Mode: Count with rate derived from the eTPU clock divided by eight. The TCRCLK signal is used to gate this count, enabling pulse accumulator operations.
- Internally Clocked Modes: TCR2 is driven by internal clock, with count rate of eTPU clock divided by eight.

All clock sources pass through a prescaler. In addition, the TCR2 count can be originated from the EAC which is a hardware angle clock and angle counter. Figure 43-317 shows the diagram for TCR2 clock control. When TCR2 is not driven by the EAC or STAC, the ETPUTBCR field TCR2CTL selects the clock source, also allowing TCR2 to be frozen independently of TCR1 (see the ETPUTBCR register). When in Angle Mode, TCR2CTL selects the TCRCLK edge sensitivity.

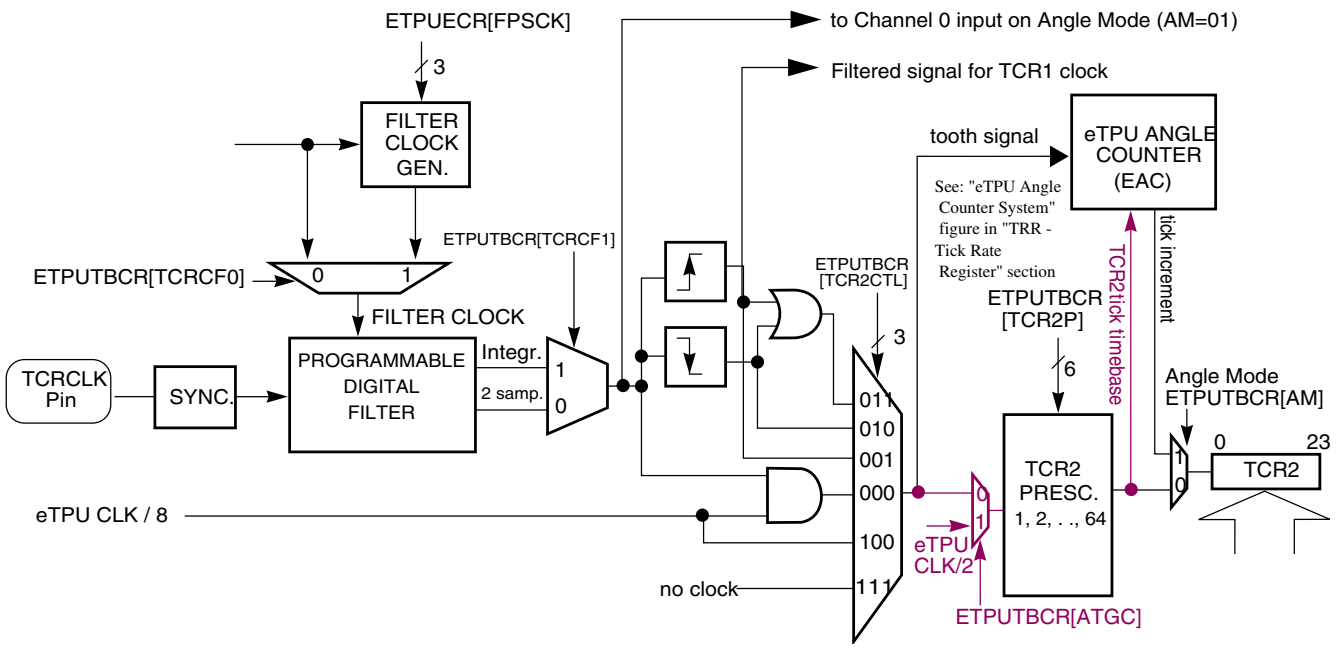


Figure 43-317. TCR2 Clock Control

The TCRCLK signal input is passed through a synchronizer and a programmable digital filter. In Angle Mode with AM=01, synchronizer and filter are also used in Channel 0, replacing its input synchronizer and filter, to get the same timing in the EAC and Channel 0. The TCRCLK synchronizer is an improved filter that provides best latency while maintaining proper noise filtering (see the field TCRCF in eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1)

The TCR2 bus runs through all the local engine channels. It transitions on clock T2 (see Microcycle and I/O Timing). In channels which select TCR2 as MatchA and/or MatchB source, TCR2 value is compared against MatchA and/or MatchB registers. A recognized match event sets its related Match Recognition Latch 1 or 2, and according to the Predefined Channel Modes (PDCM) it may generate a channel service request. For details on eTPU channels refer to Enhanced Channels.

The TCR2 counter is accessible by the microcode for read and write operations. Its current value is used for getting the current counter value (representing signal transitions, time or angle), and the captured values are used for channel relative count calculations of future events. The TCR2 value is readable to the host through the ETPUTB2R register (refer to the ETPUTB2R register). When the TCR2 bus value is imported from the STAC bus (STAC client mode), TCR2 is not writable by the microcode, and read access from the microcode or from the host reflect the imported TCR2 value.

43.14.6.2.1 TCR2 Clock Prescaling

Except in Angle Mode, any clock source selected by TCR2CTL is prescaled by a factor of 1 to 64, selected by ETPUTBCR field TCR2P. For more information on prescaler configuration refer to the ETPUTBCR register. The TCR2 Prescaler resets when the eTPU GTBE input is negated. After reset, it starts counting up to TCR2P when the eTPU GTBE input is asserted. When TCR2 increments (the eTPU GTBE input = 1), the prescaler starts a new count and the new TCR2P becomes effective. When TCR2 is written by microcode, the prescaler is reloaded with TCR2P and it becomes effective, if the eTPU GTBE input is asserted.

The counter that divides the eTPU clock by 8 before the prescaler also resets when the eTPU GTBE input is negated, or when TCR2 is written by microcode.

In Angle Mode the TCR2 prescaler is not used to directly increment TCR2, but can be used, selected by ETPUTBCR bit ATGC=1, as a clock source for the EAC generation of the angle ticks which increments TCR2. In this case, the TCR2 prescaler input is eTPU clock/2, independently of TCR2CTL. See Generating the Angle Ticks for details.

43.14.6.2.2 TCR2 Gated Mode

TCR2 Gated mode is selected in field TCR2CTL of register ETPUTBCR. In this mode the TCRCLK signal enables or disables transfer of the eTPU clock divided by 8 to the TCR2 prescaler. By programming the prescaler, TCR2 can run at rates from eTPU clock divided by eight down to eTPU clock divided by 512, in steps of eight eTPU clock divisions. For more information refer to the ETPUTBCR register.

43.14.6.2.3 TCR2 Signal Transition Modes

These modes are selected when the TCR2CTL field in ETPUTBCR is set to rise, fall or "rise-and-fall" . In these modes the TCRCLK signal is the TCR2 clock source, and its maximum transition rate depends on the TCRCLK digital filter mode of operation. The TCRCLK digital filter can be programmed to use the eTPU clock divided by two, or use the same filter clock of the channels, controlled by the TCRCF field in ETPUTBCR. It contains an up-down counter which operates as a digital integrator, optimizing signal latency in the selected mode and clock rate.

When eTPU clock divided by two is selected, the synchronizer and the digital filter are guaranteed to pass pulses that are wider than four eTPU clocks (two filter clocks). Otherwise the TCRCLK is filtered with the same filter clock as the channel input signals. For details on TCRCLK and channels digital filter control refer to the ETPUTBCR register and Enhanced Digital Filter - EDF .

43.14.6.2.4 STAC Bus Client Mode

In this mode the TCR2 register is continuously updated from the STAC bus, and the clock selection and prescaling logic becomes ineffective. It is not write accessible for the microcode, and when read, it reflects the STAC bus imported value. The use of EAC is not allowed in client mode. This mode is configured through the ETPUREDCR register (see the ETPUREDCR register).

43.14.6.2.5 STAC Bus Server Mode

When TCR2 bus is exported to the STAC bus as a server, it can provide either time or angle bus to other peripherals, according to its operation mode. This mode is configured through the ETPUREDCR register (see the ETPUREDCR register). To provide sequential update of the STAC clients, the Angle tick rate must not be faster than the STAC programmed update rate. This requirement puts a limitation on the angle clock count rate on high rate mode. In this case the Angle and Angle Fraction accumulator (see Angle Tick Generator" and Figure 43-323) are advanced at rate of eTPU clock divided by eight. Therefore, the STAC update rate for the Angle Bus must not be slower than eight eTPU clocks.

43.14.6.2.6 TCR2 Bus in Angle Clock Mode

In this mode the TCR2 counter operates as part of the eTPU Angle Counter (EAC). The TCR2 bus value reflects this angle representation in which it counts Angle Ticks. Angle Mode is selected when the AM bit is set in ETPUTBCR.

Note that when TCR2 works in Angle Mode, it does not count directly from the TCR2 clock input which indicates tooth signal transition. Its Angle counter is controlled by the Count Control and High Rate logic (see Count Control and High Rate Logic), which provides the interpolated pin position, and handle cases of missing tooth, acceleration, de-acceleration and mechanical corrections.

The EAC uses the TCRCLK signal to get the tooth transition indications. The TCR2CTL field in ETPUTBCR has to be set for the appropriate tooth edge detection rise, fall, "rise-and-fall" or none. TCR2 count clock comes from the EAC control and not directly from the physical tooth. This way the EAC control processes the signal transitions and handles missing teeth and flywheel mechanical corrections. Note that when TCR2CTL selects "none" for tooth edge selection, the TCR2 is not necessarily frozen, but can still be incremented by the EAC logic.

In Angle Mode, eTPU channel 0, 1 or 2 operation is combined with the EAC operation. When channel 0 is selected for EAC operation, the TCRCLK digital filter is used both by the EAC and by channel 0 to get full synchronization between the two logics.

The eTPU Angle Counter (EAC) logic runs continuously and updates the TCR2 Angle counter, eliminating the microcode latency in updating the TCR2 value.

43.14.6.3 STAC Interface

Both time bases TCR1 and TCR2 can be shared between the Engines and with other blocks in the same MCU. Each one of both eTPU engines can drive their time bases to the STAC (Shared Time and Count) bus, acting as a server, while any other block can capture the value into its resources and behave like a client. For further reference about the STAC bus operation refer to to the ETPUREDCR register.

The eTPU can export to the STAC bus or import from the STAC bus the following internal resources:

- TCR1: Can be exported to or imported from the STAC bus. TCR1 can only be imported from STAC bus when the engine is not in Angle Mode. When TCR1 is imported from the STAC bus, it becomes read-only for the microcode and reflects the imported values. For details refer to Timer Count Register 1 - TCR1.
- TCR2: Can be exported to or imported from the STAC bus. TCR2 can only be imported from the STAC bus when engine is not in Angle Mode. When TCR2 is imported from the STAC bus, it becomes read-only for the microcode, and reflects the imported values. When exported to the STAC bus, TCR2 can work in either Angle Mode or as a free running counter associated with the TCRCLK signal. For details refer to EAC - eTPU Angle Counter.

Proper configuration of the following bits is necessary to determine what can drive the STAC bus: ETPUTBCR[AM] and ETPUREDCR[REN2, RSC2], according to the following table.

Table 43-354. STAC Bus and Host Read Sources

AM (ETPUTBCR)	REN2, RSC2 (ETPUREDCR)	TCR2 Bus Source (Host read of ETPUTB2R)	STAC Bus Driver
00	0× (disabled)	TCR2/Time	×
01, 10 or 11	0× (disabled)	TCR2/Angle	×
00	11 (Server)	TCR2/Time	TCR2/Time
01, 10 or 11	11 (Server)	TCR2/Angle	TCR2/Angle
01, 10 or 11	10 (Client)	not allowed[1]	

1. STAC client configuration in Angle Mode is also not allowed for TCR1.

Note that Angle Mode is not available for STAC bus clients: configuring both at the same time brings unspecified results and must be avoided. When TCR2 is a stand-alone counter or a STAC Bus server, the same value that is driven to the internal TCR2 bus is also exported to the STAC bus (either Time Count or Angle).

STAC bus configuration is provided by the ETPUREDCR bits REN1/2 and RSC1/2. REN1/2 enable the STAC interface to interact with the resource (either TCR1 or TCR2 bus). RSC1/2 configure the resource (either TCR1 or TCR2 bus) as Server or Client.

Each time base / angle count resource from each engine receives a unique 4-bit hard-wired address that identifies it as a potential server. This address is used by the STAC Controller to coordinate which resource will drive the bus at a given STAC time-slot. For any time-slot there is a server driving the bus upon selection of the STAC Controller, and there may be a client linked to that server by the ETPUREDCR bits SRV1/2 on each Engine. When the server address on the Red Line STAC bus matches the value in SRV1/2, the client will load the STAC information into the appropriate resource. For information on eTPU STAC Bus configuration refer to the ETPUREDCR register.

The eTPU does not include a STAC Controller module, which is instantiated once in the system integration.

Note

Setting a timebase as client of itself is not allowed.

43.14.6.4 GTBE - Global Time Base Enable

GTBE bit in ETPUMCR register enables time bases in both engines, allowing them to be started synchronously. GTBE is divided in two block interface signals: the eTPU GTBE output and the eTPU GTBE input. GTBE bit sets the eTPU GTBE output , and the eTPU GTBE input enables time bases to start. The eTPU GTBE output signal can be used by MCU integration for synchronization between eTPU time bases and time bases from other modules. If the GTBE bit in ETPUMCR must enable only the eTPU time bases, the eTPU GTBE output is simply connected to the eTPU GTBE input. These two cases are shown in the figure below. Synchronization logic can be as simple as an OR logic gate.

Once the eTPU GTBE input transitions to 1, the Engine 1 Time Bases start 1 eTPU clock earlier than Time Bases in Engine 2, except when TCRCLK is selected as clock source or TCR1 when ETPUTBCR[TCR1CS]=1 . This happens independently of prescaler values as long as they are the same for both engines, because the prescalers also freeze when the eTPU GTBE input = 0. Microcode can always write to TCR1/2 registers, with either value of the eTPU GTBE input.

NOTE

The timebase prescalers are reset when the GTBE input is negated.

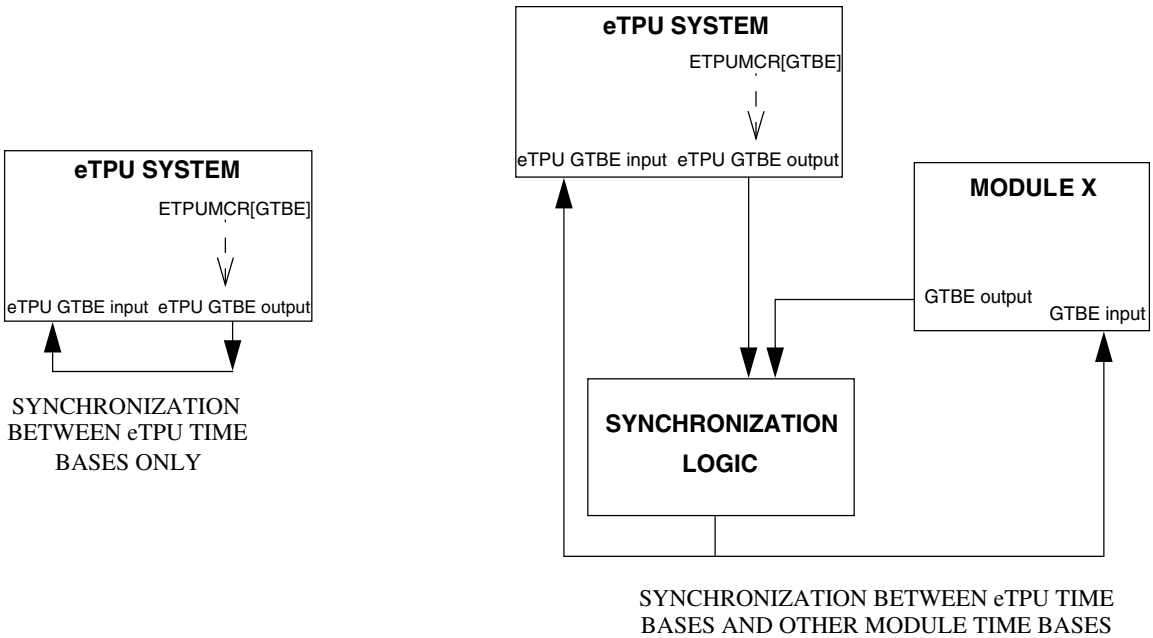


Figure 43-318. Time Base Synchronization

43.14.6.5 TCRCLK Digital Filter

The TCRCLK signal has an improved integrating digital filter with a 2-bit up-down counter. The counter counts up to 3 when a high signal level is detected, or down to 0 when a low level is detected. The signal state is updated to one when the counter stops at 3, or zero when the counter stops at 0. The field TCRCF in ETPUTBCR register (see the ETPUTBCR register) determines whether the TCRCLK signal input (after a synchronizer) is filtered with the same filter clock as the channel input signals (see Enhanced Digital Filter - EDF) or uses the eTPU clock divided by 2, and also whether the TCRCLK digital filter works in integrator mode or the same two sample mode as the channel filters (see Table 43-338).

The TCRCLK filter delay and prescaling determines the minimum detectable TCRCLK pulse widths and, therefore, its maximum frequency, as shown in Filter Clock Prescaler and Table 43-353 . The TCRCLK signal delay from the module input to TCR1/TCR2 incrementing or detection in the EAC logic is explained in Input/Output Signal Delays.

43.14.7 EAC - eTPU Angle Counter

43.14.7.1 General

The EAC logic contains a mechanism which follows the flywheel angle, based on the tooth rate. This hardware works in combination with the TCRCLK signal, the TCR2 counter and Channel 0, 1 or 2 (depending on the ETPUTBCR field AM) to generate angle information on the TCR2 bus which is passed to all the local engine channels. The EAC helps to implement a digital angle PLL (see Figure 43-322), which combines hardware with microcode processing at channel 0, 1 or 2 . The angle measurement is based on history knowledge of the tooth period, for predicting the period of the next tooth. The tooth period is partitioned into a programmable number of **Angle Ticks**. The eTPU application will use the divider in the MAC/Divide unit to calculate an integer and a fraction part of the angle tick such that the full tooth period gets the correct programmed number of angle ticks with no accumulated error.

Each single tooth can be divided in angle ticks, up to 1024. In a 60 teeth flywheel, 128 Angle Ticks per tooth provide resolution of ~0.05 degrees per tick, which meets the accuracy requirement of 0.1 degrees in current automotive applications.

The measurement of one tooth in angle ticks is independent on engine RPM; it is the tooth period itself (and the corresponding tick period) that is re-calculated for each new tooth, based on the difference between the estimated tooth and the actual detection.

For these applications, one of the eTPU channels 0, 1 or 2 is dedicated to service the physical tooth detection. Channel 0 shares the same filtered input as the TCRCLK signal to get the same timing as the EAC. The TCRCLK edge detection is selected by ETPUTBCR field TCR2CTL for the EAC, and by IPACA/B on channel 0, which must be set to detect the same edge(s). When channels 1 or 2 are selected to work with the EAC, IPACA/B is used to select the tooth signal edge detection for both the channel and the EAC, and the tooth signal that feeds the EAC is the same filtered input which feeds the channel.

Channel 0, 1 or 2 generates the signal transition service request, and can also be used for generation of a window filter on this transition, to qualify TCR2 clocks. For this purpose, the selected channel should be configured with double match window filtering mode (refer to Channel Modes). Depending on the channel mode set for the channel, Match A recognition opens the window, and Match B recognition may close it or leave it open. See Angle Logic and Channel Modes for details. Match B also generates a time-out service request. Its input signal transition comes from the tooth. The window can be defined by microcode to open at a predefined point inside the tooth period, and stay open for a desired percentage to the tooth period. The window can be measured in angle or

time This method improves the noise immunity by allowing transition detection only on an expected period, a feature which was software responsibility in previous TPU versions.

The EAC supports deceleration, acceleration, last tooth and missing tooth scenarios. The large range of angle ticks per tooth can be used to cover longer tick counts caused by one or more missing teeth, or to provide extra resolution for future application requirements. In case of a missing tooth, the EAC can be configured to insert a dummy tooth or to simply measure a longer tooth.

Figure 43-323 shows the block diagram of the Angle Counter system. TCR1 or TCR2 prescaler is used as a time base which measures the tooth period and is used for partitioning the period to angle ticks.

43.14.7.2 Angle Mode Registers

In Angle Mode, the registers described below control eTPU angle operations. They are accessible only by microengine as source and destination registers in microinstructions. When eTPU is not in angle mode (AM bit is negated in ETPUTBCR register), all angle mode registers can be used as general purpose registers.

43.14.7.2.1 TPR - Tooth Program Register

TPR provides configuration for the Angle Counter circuit. In this register, the microcode can properly adjust the tooth count (controlling last tooth, missing teeth, dummy tooth insertion, halt until tooth detection) and the number of angle ticks per tooth (field TICKS). Note that this register is sampled into a temporary register in the EAC logic when the High Rate Mode is detected (see High Rate Mode (Acceleration) and TPR Buffering), which means that changes to this register may take effect only for the next tooth.

Refer to Count Control and High Rate Logic and to Special Cases of Missing Teeth and Last Tooth until Handling False Tooth Detection for a detailed explanation about the use of this register. The figure below provides a detailed description of TPR register.

Several conflict issues on TPR writes are explained in Special TPR Write Cases.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LAST	MISSCNT [1:0]		IPH	HOLD	MISS CNT [2]	TICKS									
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-319. TPR Register

LAST —Last Tooth Indication

Asserted by microcode and negated when a tooth is detected or inserted via IPH.

1 = Last Tooth - reset TCR2 Counter at the end of the tooth tick count (after physical tooth or IPH=1) when MISSCNT=0.

0 = Not Last Tooth.

If the tooth is detected or inserted before the tooth tick count completes (going High Rate mode, see High Rate Mode (Acceleration)), LAST resets immediately, but TCR2 resets only when the tooth count completes and MISSCNT=0.

MISSCNT — Missing Tooth Counter

Decremented on each estimated tooth, stops at zero. Used for generation of up to 7 "Dummy Teeth" whenever it holds a non-zero value.

If the tooth is detected or inserted before the missing tooth tick count completes (going High Rate mode, see High Rate Mode (Acceleration)), MISSCNT resets immediately, but missing teeth count continues in High Rate mode (see TPR Buffering).

IPH — Insert Physical Tooth

This bit generates a dummy physical tooth which has the same effect as a real physical tooth, and resets itself subsequently. If EAC is in Halt mode, it switches back to Normal mode[1]. If EAC is in Normal Mode, it switches to High Rate Mode. If Angle Logic is frozen by HOLD=1 (see below), it returns to the state it was at the freezing moment.

1 = Insert dummy physical tooth.

0 = No Operation.

Note: IPH reads as 1 in the next microinstruction after it is asserted, negating subsequently. However, it can be set twice in two consecutive microinstructions to generate two teeth and make the EAC go from Halt to Normal to High Rate Mode.

HOLD — Force EAC Hold

This bit forces the EAC to halt its operation in a special EAC freeze mode until a new physical tooth (a real one or emulated with IPH=1) is detected. Assertion of this bit immediately freezes the EAC in the middle of the tooth period. When a new physical tooth is detected, the bit is automatically negated by the EAC. The HOLD bit can be used for synchronizing the EAC tooth count, in case that a false physical tooth is detected due to noise.

1 = Force EAC to halt until detection of a physical tooth.

0 = Normal Operation.

TICKS — Angle Ticks Number in the Current Tooth

This field defines the number of angle ticks in the current physical tooth. It partitions the tooth period to the required number of angle ticks. The actual number of angle ticks in a tooth is (TICKS+1).

Note: In High Rate mode (see High Rate Mode (Acceleration)), TPR writes are immediately effective only for bits IPH and HOLD. All other fields changes are "buffered" and become effective when EAC leaves High Rate mode. See also Special TPR Write Cases.

Note: Bits LAST, IPH and HOLD must not be asserted all at once

1. missing a physical tooth naturally causes EAC to get into Halt mode.

43.14.7.2.2 TCR2 - Timer Counter 2

In Angle Mode, TCR2 (see the following figure) counts angle ticks instead of time.

	23	22	21	20	19	18	17	16
R	Angle Tick Counter[23:16]							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Angle Tick Counter[15:0]															
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-320. TCR2 in Angle Mode

This 24-bit free-running counter is used to generate an accumulated Angle Fraction value. It is updated by the Angle Tick Generator - refer to Angle Tick Generator for more details. Refer to Count Control and High Rate Logic for a detailed explanation about the use of this register in Angle Mode.

TCR2 provides continuous count of the angle in units of angle ticks. The Angle Tick Counter in TCR2 can be reset due to "Last Tooth" microcode indication. TCR2 Prescaling is disregarded in Angle Mode: physical tooth detection is done by EAC regardless of the value set in TCR2P.

43.14.7.2.3 TRR - Tick Rate Register

The exact period of the Angle Tick is programmed in the Tick Rate Register (see the figure below) by microcode. The TRR time unit depends on the ETPUTBCR bit ATGC (see Figure 43-323):

- if ATGC=0, the TRR time unit is the period of TCR1 clocks taken as eTPU clocks divided by 2*(TCR1P + 1), even if TCR1CS=1 (see the ETPUTBCR register) .
- if ATGC=1, the TRR time unit is given by the eTPU clock divided by 2*(TCR2P+1), using the TCR2 prescaler fed by eTPU clock/2 (independently of TCR1CS).

Refer to Calculating the Angle Tick Period Integer and Fraction for a complete description about the mechanism to calculate the value to be written into this register.

	23	22	21	20	19	18	17	16
R	INTEGER[14:7]							
W								
RESET:	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INTEGER[6:0]							FRACTION								
W																
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-321. TRR Register

INTEGER — The integer part of TCR1/TCR2 clocks in one Angle Tick.

This number, decremented by one, works as a down-counter preload value. A value of INTEGER=0 represents an integer of 32768. A new value written is reloaded into the counter (becoming effective) when a new tick starts or a tooth is detected or inserted via IPH.

FRACTION — Nine-bit fractional part of TCR1/TCR2 clocks in one angle tick.

The FRACTION value is accumulated in the EAC Fraction Accumulator, and whenever the result overflows (i.e., the accumulated fraction added up to an integer), the Tick Prescaler is halted for one TRR time unit.

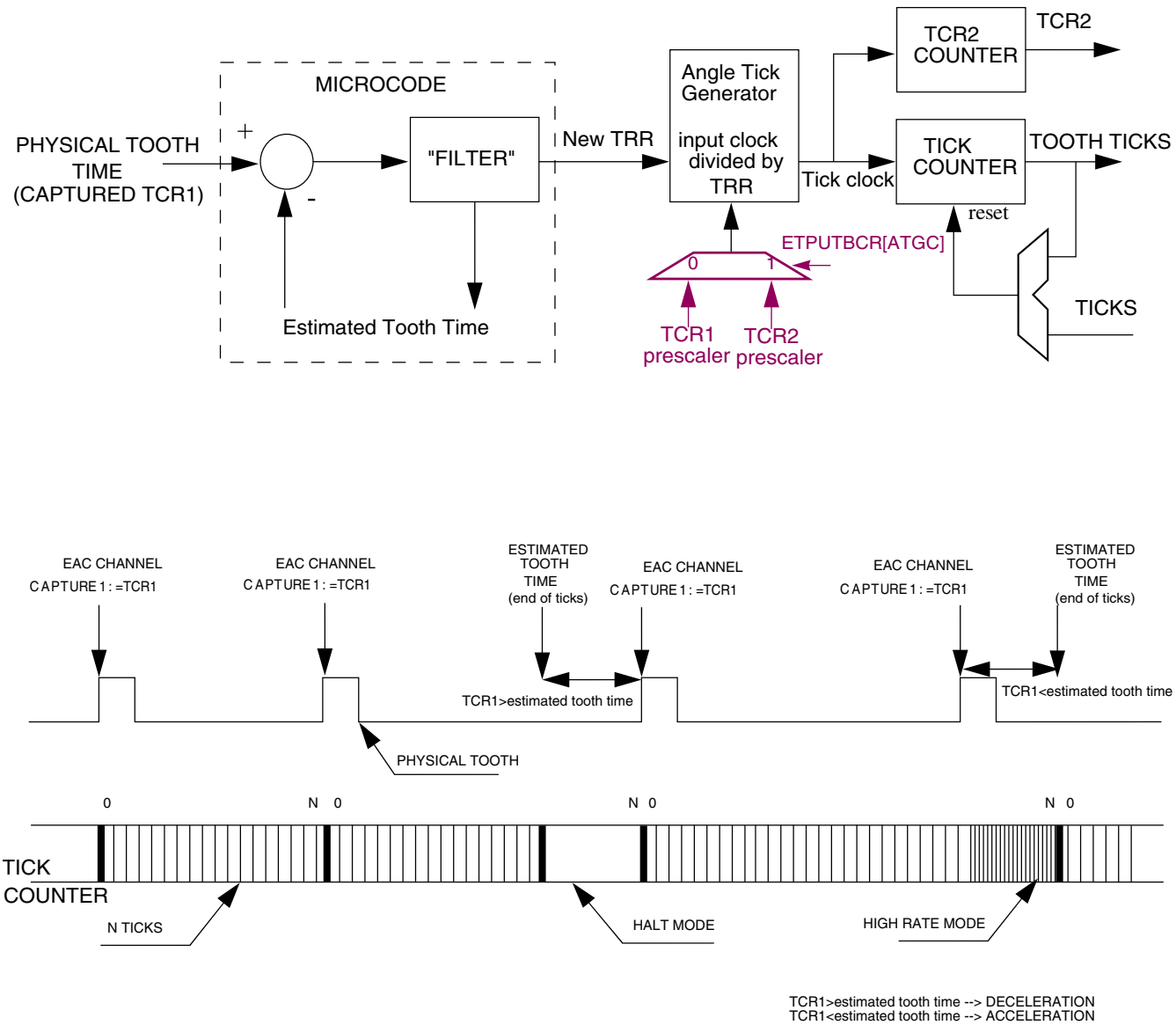


Figure 43-322. EAC "PLL"

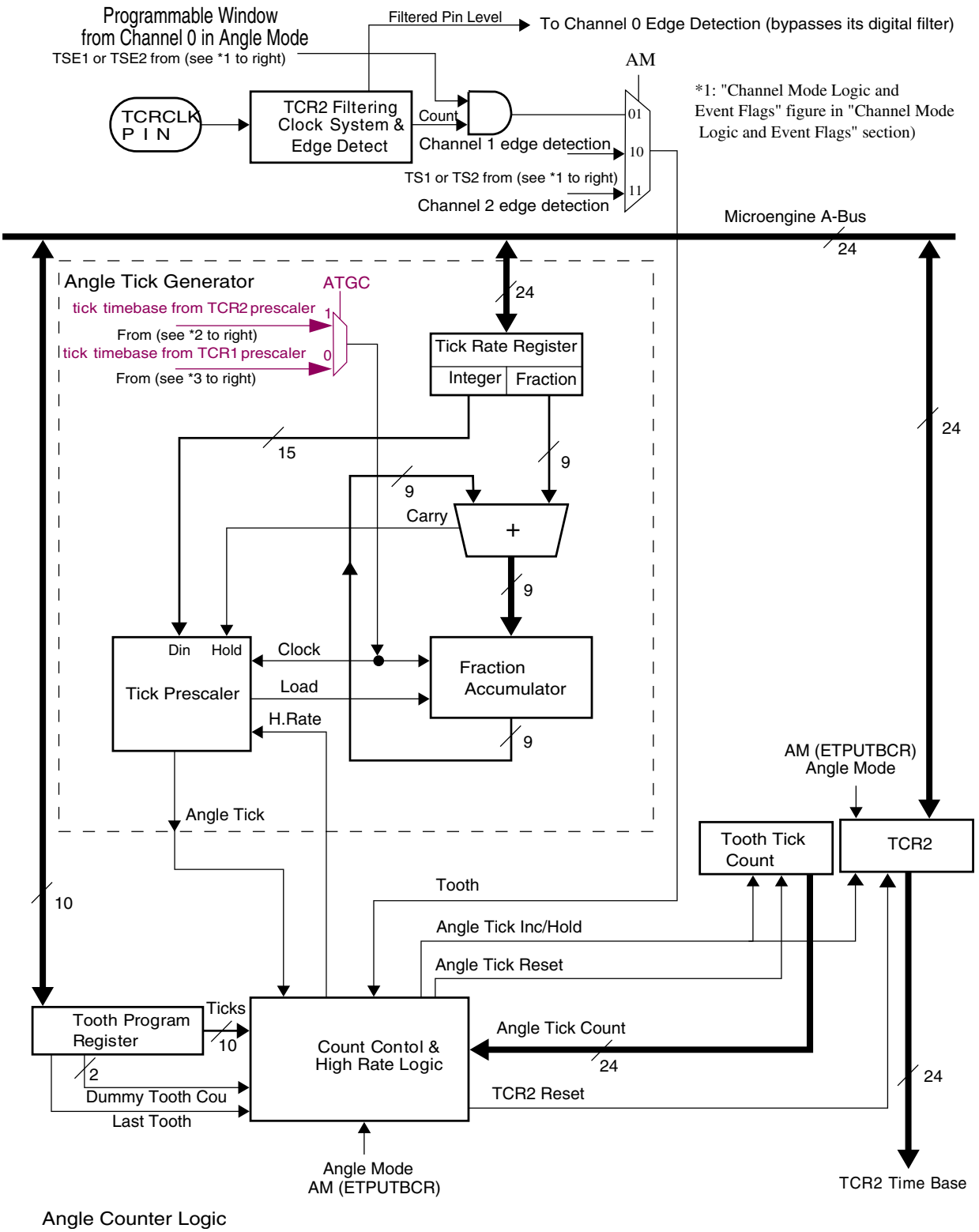


Figure 43-323. eTPU Angle Counter System

43.14.7.3 Acceleration and Deceleration

Acceleration and deceleration affect the new tooth period relative to the known period of the last tooth. Changes in tooth period may be extreme at very low engine RPM (such as cold start and warm start). The worst case of tooth period changes is caused during missing teeth, since there is more time for changes in angular velocity to be unnoticed by the EAC hardware. For example, on cold start (~20 RPM) there may be extreme acceleration: the ratio between a known tooth period before two missing teeth and the new tooth period after the missing teeth can be very high (up to a factor of 75). Acceleration and deceleration effects from tooth to tooth are less extreme as the engine climbs to high RPM.

In case of deceleration, the estimated tooth period ends before the actual tooth detection arrives. In this case, the EAC hardware waits at the end of the current tooth period, when it is said to be in **Halt mode**, until the real tooth indication is received, then continues with normal operation (**Normal mode**). See Figure 43-322.

In case of acceleration, the actual tooth period is shorter than the estimated tooth period. As a result, a new physical tooth indication arrives before the end of the estimated tooth period. In this case the EAC closes the gap on **High Rate mode** by counting on eTPU clock divided by eight to the end of the tooth, advances to the next tooth, and switches back to normal operation mode. See Figure 43-322.

The reason that the EAC does not jump directly to the next tooth is the need to provide sequential angle count throughout the whole tooth period, for channels or external STAC clients (if TCR2 is a STAC server) which compare angle in "equal" mode. These peripherals must get all the valid angle values in a sequential manner, to avoid missing angle matches.

TCR2 advancing from one tooth to another is a continuous count, and can be optionally reset at the end of the tooth. An estimated tooth is generated after the Tooth Tick Counter reaches the TICKS programmed value.

The EAC works continuously and switches automatically between Normal, Halt and High Rate modes. It relies on the microcode to calculate the estimated tooth period on every tooth, and to update the correct angle tick and tooth parameters in the EAC control registers. On high RPM, tooth period changes are reduced from tooth to tooth, and the EAC may follow the angle with good accuracy for several teeth without microcode intervention.

The EAC handles missing teeth by insertion of "dummy" teeth, or by enlarging the expected tooth period. It is a good practice to locate the flywheel missing teeth in non-critical angles, since a missing tooth may increase the angle measurement error (acceleration and deceleration is detected late).

43.14.7.4 Angle Tick Generator

The Angle Tick Generator is responsible for generating a programmed number of angle ticks in the tooth period. It generates the ticks in an average rate which ensures completion of the correct number of angle counts in the estimated period of the tooth, since the count of one tooth in angle ticks is independent on engine RPM. The main output of the Angle Tick Generator is the tick clock that feeds TCR2 in Angle Mode, as well as the internal Tooth Tick Counter (see Figure 43-323). The Tooth Tick Counter counts ticks within a tooth, from 0 up to TICKS, is controlled by the Angle Tick Generator logic and cannot be accessed by microcode. Refer to Figure 43-324 for a generic presentation of the angle tick count and the measurement of a single tooth period.

43.14.7.4.1 Calculating the Angle Tick Period Integer and Fraction

On each tooth the microcode has to update the exact period of a single angle tick used for counting the internal angle in the tooth. The period of an angle tick is determined by the EAC configuration, see Generating the Angle Ticks for details.

The microcode can use the eTPU MAC Divider unit (see MAC and Divide Unit (MDU)) to divide the tooth period by the number of angle ticks per tooth, which is stored in the TICKS field of TPR (refer to Angle Mode Registers). This division yields the integer part of the angle tick period and the remainder. Dividing again the remainder shifted left nine positions, by the number of angle ticks per tooth translates the remainder to a 9-bit fraction. The microcode concatenates the 15-bit integer and the 9-bit fraction to a 24-bit value and writes it to TRR. The new rate is effective immediately after the next Angle Tick is generated by the Angle Tick Generator[27] .

For high RPM, note that shifting the tooth period value nine positions to the left prior to the first divide operation would calculate, in one operation, the integer and the fraction. For example: On 60 teeth flywheel running at 1000 RPM, tooth period is 1 msec. If TCR1/TCR2 counts @25 MHz, it counts 25,000 times in a tooth, which can be represented by 15 bits. Therefore the tooth period can be shifted nine positions to the left prior to divide operation, and be represented with 24 bits.

Using shift left nine positions and one divide operation would get the result in MACL register (in MDU) which holds the integer and nine bits of the fraction:

```
Angle_Tick_Rate {Integer[14:0], Fraction[8:0]} = (ToothPeriod[28] <<9) / Ticks
TRR = Angle_Tick_Rate {Integer[14:0], Fraction[8:0]}
```

27. In High-rate mode, the tick keeps being updated at the rate of eTPU clock/8 until it goes back to Normal mode, when the new TRR value is used.

On low RPM the initial tooth period, measured in TCR1/TCR2 counts, may be too big to be shifted nine positions to the left. For lower RPM (for example 500 RPM) the tooth period cannot be represented by 15 bits, and shifting it nine positions to the left would lose the MSB. In this case, two divide operations are required as follows: first divide the Tooth Period by the number of TICKS - the integer is stored in MACL and remainder in MACH. MACL is saved in another register. MACH is shifted 9 positions to the left and divided again by TICKS. In parallel with the second divide, the register which saved the original MACL is shifted left 9 positions. After the divide MACL contains the 9 bits fraction and the other register contains the 15-bit integer, shifted left nine times. The logical OR of the two registers is written to the TRR:

```
Angle_Tick_Rate {Integer[14:0]} = (ToothPeriod) / Ticks
Remainder[9:0]                  = ToothPeriod) mod Ticks
Angle_Tick_Rate {Fraction[8:0]} = (Remainder[9:0] << 9) / Ticks
TRR = Angle_Tick_Rate {Integer[14:0], Fraction[8:0]} = (Integer[14:0] << 9) | Fraction[8:0]
```

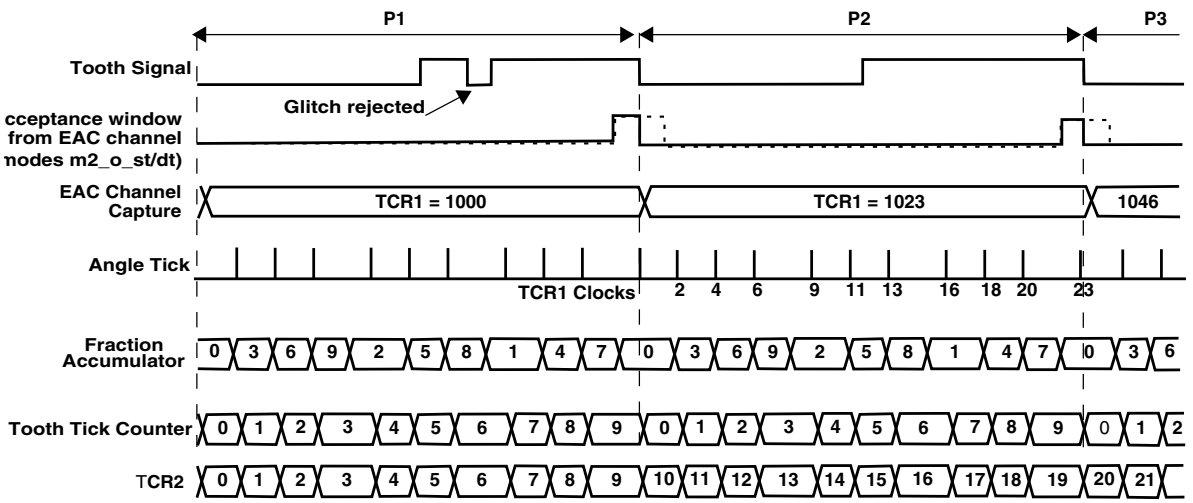


Figure 43-324. Angle Ticks Generation

43.14.7.4.2 Generating the Angle Ticks

The integer part of TRR is preloaded to a prescaler, which counts in time intervals as follows (see Figure 43-323):

28.

The tooth period (ToothPeriod) is not, in general, the value of estimated tooth time. It is obtained by microcode by subtracting TCR1 values between two teeth detections, converted to TCR2 periods if ATGC=1. Its comparison with the estimated tooth time indicates acceleration (if minor) or deceleration (if greater) to the microcode.

- if ETPUTBCR bit ATGC=0, one unit of TCR1 clock period (if TCR1CS=0) or twice the TCR1 clock period (if TCR1CS=1)[29], that is, eTPU clock period*2*(TCR1P+1) (independently of TCR1CS).
- if ATGC=1, two units of the eTPU clock period multiplied by the TCR2 prescaler factor, that is, eTPU clock period*2*(TCR2P+1).

When the down counter reaches zero, it generates an angle tick pulse to the Angle Counter Logic and a Load pulse to the Fraction Accumulator. It is then preloaded with most updated TRR integer part. Due to the Load pulse, the 9-bit fraction is accumulated in a 9-bit Fraction Accumulator. If a fraction overflow condition occurs (the 9-bit adder asserts carry out), the accumulator saves the lower 9 bits of the addition result, which is the remaining fractional part. The carry out bit indicates an accumulated integer "one" which means that the angle tick is early by one input clock. It halts the prescaler operation for one input clock to compensate the accumulated error generated by the integer prescaler. As a result, the average angle tick period takes into account both the integer and the fraction parts. The accuracy depends on the bit count of the fraction. Using 9-bit fraction part while the width of the field TICKS in register TPR is 10 bits provides accuracy of two LSB on a full scale (TICKS=1023) or one LSB on lower scale (TICKS<=511).

When the Tick Prescaler gets High Rate mode indication from the Angle Counter Logic, it generates angle ticks at a rate of eTPU clock divided by eight. In this case it does not generate Load pulses to the Fraction Accumulator, ignores its "hold" input and preloads internally to a fixed period of eight eTPU clocks. When High Rate mode is entered, the prescaler is preloaded to a period of eight eTPU clocks before its first angle tick generation, ensuring separation of at least eight eTPU clocks between the last Normal mode angle tick and the first High Rate mode angle tick. The fraction accumulator resets when the tick count advances to the next tooth, or when TRR is written by the microcode.

43.14.7.5 Count Control and High Rate Logic

The Count Control and High Rate Logic controls TCR2 operation in Angle Mode, using the angle ticks generated by the Angle Tick Generator. Count Control logic is responsible for advancing, holding and resetting the TCR2 and Tooth Tick Counter in the proper timing, such that the TCR2 time base will reflect the correct estimated angle. This logic also includes the Tooth Program Register (TPR, see Angle Mode Registers for more information).

29. The EAC logic overall runs at a rate of eTPU clock divided by 2, therefore the tick generator takes the halved TCR1 clock when TCR1CS=1 and ATGC=0.

The Count Control and High Rate Logic handles deceleration, acceleration, missing teeth and last tooth. On High Rate (acceleration) it ensures that the angle bus scans all valid angle values in a rate which can be traced by the STAC bus. This operation enables external STAC clients (if TCR2 is a STAC server) or channels working in "equal-only" comparator mode to match the TCR2 exported angle information in "equal" mode, in an exact match.

Because the eTPU channels are capable of capturing either TCR1 or TCR2 due to signal transition, the microcode can get either the angle or time of the physical pin transition. Since the EAC channel (0, 1 or 2) is connected to the physical tooth, the microcode can get the EAC error in angle domain (tooth appears at the wrong angle) or time domain (physical tooth captured time into the EAC channel, relative to the estimated tooth time). Note that in angle mode, the transition detect logic of the channel 0, if selected as the EAC channel, is fed from the digital filter of the TCRCLK signal, and not from the channel 0 internal digital filter. This ensures synchronous operation of channel 0 and the EAC hardware.

Another feature of the eTPU channel, when working in Single Match And Single Transition Enhanced Mode (refer to Single Match Enhanced Mode (sm_st_e)), is capturing a single time base due to signal transition before and after the digital filter. This option allows subtracting the digital filter delay to get accurate signal transition timing on the channel. This way, the TCRCLK signal may be programmed with a slow and reliable digital filter, and get accurate time measurement of the digital filter delay.

To assert the end of the estimated tooth period the Count Control and High Rate logic compares the TICKS field in TPR (refer to Angle Mode Registers) with the current value of the tooth tick counter. When the Tooth Tick value is greater or equal to TICKS, it determines the end of the estimated tooth period. On acceleration this event occurs during High Rate mode operation, after the arrival of a physical tooth. In deceleration, this event occurs during Normal Mode, before the arrival of a physical tooth. On constant angular velocity, this event appears together with the arrival of a physical tooth.

The following sections describe the operation of the Counter Control and High Rate logic.

43.14.7.5.1 Normal Mode

In Normal mode the Counter Control logic advances TCR2 and the Tooth Tick counter as if the engine has a constant speed during the tooth period. It receives the angle ticks from the Angle Tick Generator in an average rate which is determined by the Tooth Rate Register (TRR). This is the reset mode.

When the Tooth Tick Counter is about to reach the last value effectively stored in TPR field TICKS plus one, the hardware detects the end of the estimated tooth period. If the physical tooth and the estimated tooth arrive at the same time the EAC stays in Normal mode, the Tooth Tick Counter is reset, and TCR2 is incremented (depending on TPR bits LAST and MISSCNT). If the physical tooth and the estimated tooth do not arrive at the same time, either acceleration or deceleration is detected, and the EAC switches to the proper mode. See the figure below for a detailed diagram of Normal Mode behavior.

The microcode which services the EAC channel physical tooth transition may update TRR according to various conditions to give the best estimation of the current tooth period, according to the previous tooth period and other engine parameters.

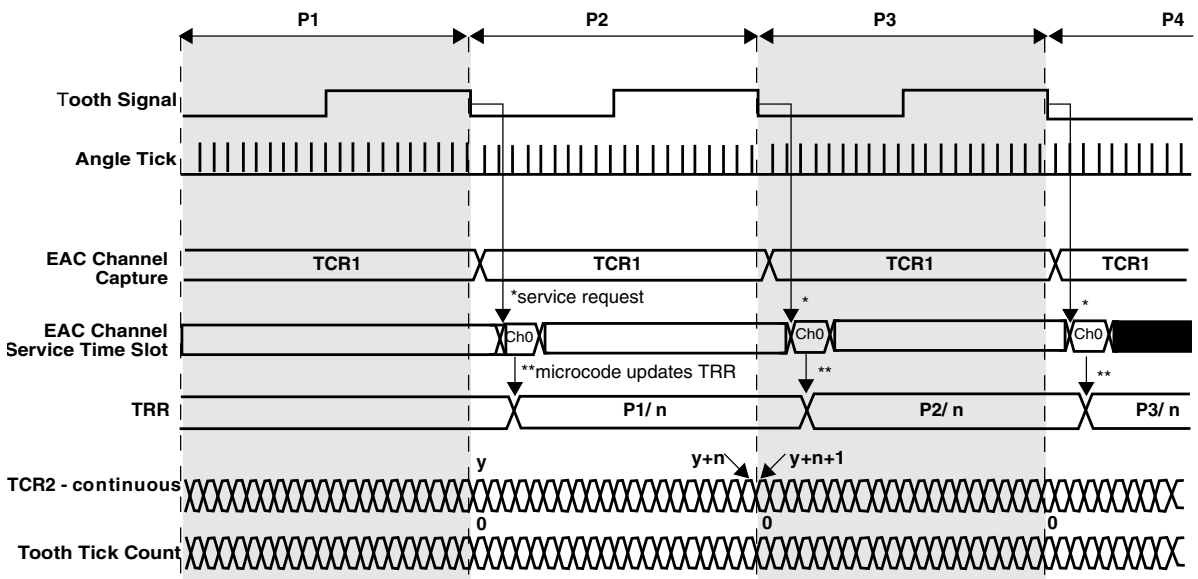


Figure 43-325. Normal Mode

43.14.7.5.2 Halt Mode (Deceleration)

In case of deceleration, the Tooth Tick Counter reaches the TICKS value before the arrival of the next tooth. The Count Control logic does not reset neither advances TCR2 and Tooth Tick Counters. The Count Control logic halts TCR2 and Tooth Tick Counters at the end of the tooth, waiting for the physical tooth to arrive.

When the physical tooth is detected the EAC switches back to Normal Mode and releases TCR2 to count the angle ticks of the new tooth, also resetting the Tooth Tick Counter. Only then TCR2 may wrap to 0, if TPR bit LAST is asserted. See the figure below for a detailed diagram of Halt Mode behavior.

The microcode service caused by the physical tooth determines the deceleration, calculates the new tooth period and Angle Tick period and updates TRR. This operation slows the angle tick rate generated by the Angle Tick Generator on-the-fly, to the rate required for the new tooth period.

Since the microcode service is initiated by the physical tooth edge, microcode latency may introduce a small angle error caused by using the TRR value of the previous tooth at the beginning of the current tooth. On high RPM, deceleration is relatively small but the microcode latency may take a significant percentage of the tooth period. On low RPM microcode service latency takes little percentage of the tooth period, but there may be cases of extreme acceleration and deceleration. The microcode latency can be calculated knowing TCR1 value during the service time, and TCR1 value captured in the EAC channel due to the physical tooth pin transition. The duration of the Halt mode is obtained using the estimated tooth time.

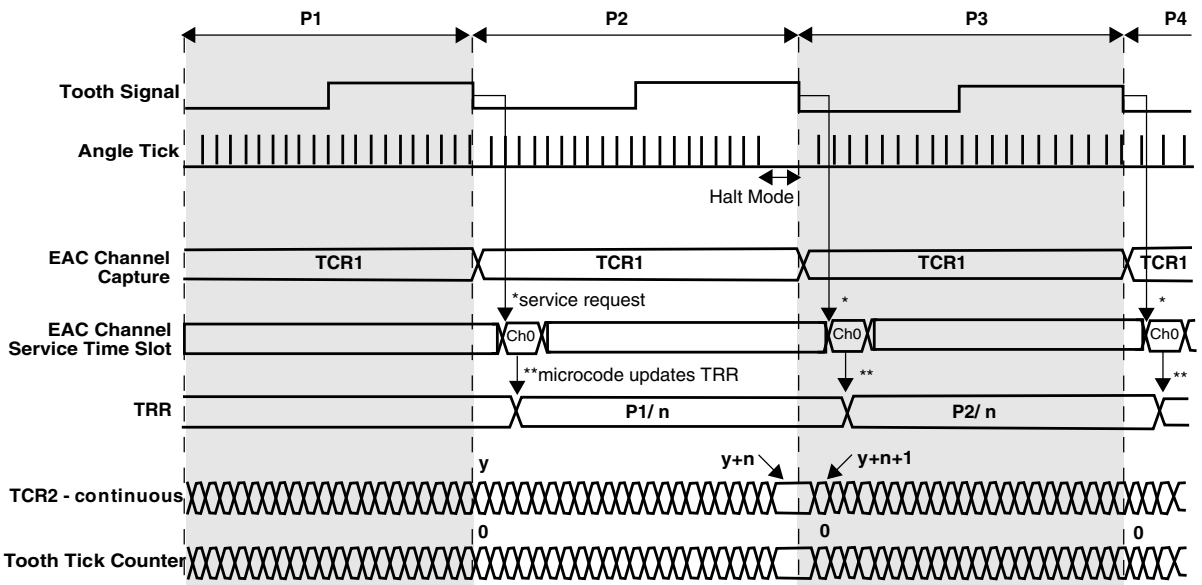


Figure 43-326. Halt Mode - Deceleration

43.14.7.5.3 High Rate Mode (Acceleration)

In case of acceleration, the next tooth arrives before the Tooth Tick Counter reaches the TICKS value. In this case the High Rate logic is responsible for closing the gap and advancing the tooth on the correct timing. The High Rate mode operates as follows:

- When the acceleration is detected (physical tooth arrives before the Tooth Tick Counter reaches the TICKS value), the Count Control and High Rate logic switches to High Rate mode in which both the Tooth Tick Counter and TCR2 count at rate of

eTPU clock divided by eight, until the Tooth Tick Counter reaches the current TICKS value. To ensure correct operation, the TICKS value is sampled in the logic at the beginning of the mode.

- At this point, which represents the estimated tooth edge, the logic resets the Tooth Tick Counter and advances TCR2 (or resets it if LAST is asserted and MISSCNT = 0).
- The control logic switches back to Normal Mode, using the most updated TRR value as input to the Angle Tick Generator. The logic samples the updated TICKS value for the tooth estimation, last tooth indication and number of missing teeth from TPR.

In High Rate mode the angle ticks are provided at high speed until the end of the current tooth. This operation is required to scan all the valid angle values of the current tooth, in a rate which is not too high for the STAC bus continuous update, but much higher than the rate dictated by TRR.

EAC channel microcode, which services the physical tooth transition detection, can start its service either before High Rate mode operation is complete (the Tooth Tick Counter has not reached the TICKS value) or after the EAC switched back to Normal mode. Any physical teeth received while the EAC is in High Rate Mode does not alter the immediate EAC state, but it is still detected by the EAC channel logic and can, therefore, alter future EAC behavior (for instance, closing the tooth detection window - see Angle Logic and Channel Modes).

At the beginning of High Rate mode operation, the TPR value is preloaded into a temporary register in the Counter Control Logic, used for scanning all the valid values to the end of the current tooth, with its appropriate LAST and MISSCNT attributes. While the EAC is in High Rate mode operation, the effect of microcode update of TPR fields LAST, MISSCNT and TICKS is delayed to the next estimated tooth, after the High Rate mode operation is complete (see Special TPR Write Cases). This is because the current physical tooth represents the next estimated tooth. If the microcode updates this field after High Rate mode operation is complete, the current physical tooth and estimated tooth are the same, and the effect is immediate. Either in High-Rate mode or not, the value read by microengine is the same written, even if not yet effective, until the EAC resets LAST and/or IPH, or decrements MISSCNT. Typically the microcode service may occur during the High Rate mode on extreme acceleration situation at low RPM. Therefore, the microcode operations are always related to the real physical tooth. From the above it can be seen that the microcode updates of the TICKS field in TPR affect the end time of the current physical tooth. For correct operation, this field should be updated before the Tooth Tick Counter has reached either the old or the new TICKS value.

During High Rate mode operation, TRR is ignored and the Angle Tick Generator uses eTPU clock divided by eight. Therefore, the TRR update by microcode will take effect only after the EAC switches back to Normal mode. If microcode service occurs after the Tooth Tick Counter has been reset, the EAC is already back in Normal mode, and some angle ticks may have been counted at the rate of the previous tooth. In this case the new TRR value will have immediate effect on the angle tick period, and the microcode should take into consideration the delay from the physical tooth to the estimated tooth in calculation of the next tooth period. See the figure below for a detailed diagram of High Rate Mode behavior.

An angle error may be introduced by the duration of the High Rate mode. Also, the scheduler latency may introduce a small accumulated error by using TRR value of the previous estimated tooth at the beginning of the current tooth. After the estimated tooth has advanced, the duration of the High Rate mode operation is the actual delay from the physical tooth edge to the estimated tooth edge. This delay can be obtained by comparing the estimated tooth time with the EAC channel capture register which captured TCR1 on the physical pin transition.

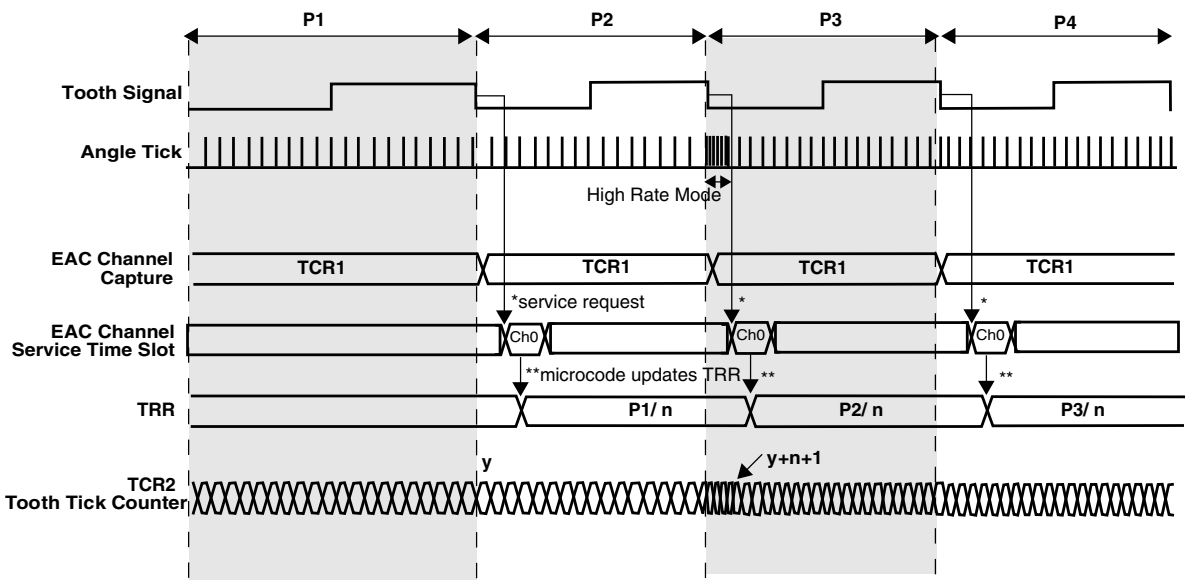


Figure 43-327. High Rate Mode - Acceleration

43.14.7.6 Special Cases of Missing Teeth and Last Tooth

The EAC handles cases of up to three missing teeth and the last tooth in the engine cycle. The following paragraphs describe these functions.

43.14.7.6.1 Handling the Last Tooth

The microcode can set the TCR2 counter to work in engine periods (wrap-around count) or continuous angle measurement.

For periodic operation, during the last engine cycle tooth the EAC microcode has to set the LAST flag in TPR. As a result, when the tooth period is ended, the Counter Control Logic generates a reset command to both the Tooth Tick Counter and TCR2 instead of an advance command. The operation resets the TCR2 based angle count, indicating a new period of the engine cycle. This implementation provides an engine cycle based periodic angle measurement.

43.14.7.6.2 Handling Missing Teeth

The EAC can handle up to three missing teeth in two ways:

- Count the angle ticks relative to the last physical tooth. The microcode should update the TPR TICKS field to the number of angle ticks included in two, three or four teeth, according to the flywheel type (one, two or three missing teeth). EAC hardware works in its regular manner.
- Insert a "dummy" tooth instead of the missing tooth, at the estimated point in time. After the "dummy" tooth, the Angle Tick Counter is incremented as if there was a physical tooth. A "dummy" tooth can be inserted only during Normal or High Rate operation modes. The microcode inserts "dummy" teeth by writing to the MISSCNT field in TPR.

In the first option the missing tooth is not counted on the angle measurement. For example, a flywheel with 59 physical teeth and one missing tooth can be considered as 58 identical teeth numbered (0-57) and tooth number 58 has a double number of angle TICKS. In this case a 720 degrees engine cycle has 118 teeth. TCR2 reflects the real angle, since it counts angle ticks continuously.

In the second option, the missing teeth are counted as "regular" teeth by automatic insertion of "dummy" teeth. The microcode has to write a non-zero value to the MISSCNT field in TPR. This field is a 3-bit down counter which affects the operation of the Counter Control logic.

For example, a flywheel with 59 physical teeth (0-58) and one missing tooth (59) can be considered as 60 teeth numbered (0-59), all having the same number of angle ticks. The microcode has to write "001" to the MISSCNT bits during the period of tooth number 58 to indicate that next tooth (59) is missing.

When the Tooth Tick Counter reaches the TICKS value, TCR2 is incremented as if a physical tooth has been detected. In addition, the MISSCNT value initializes a "dummy tooth counter" which is decremented to indicate the number of left "dummy teeth" which still need to be generated. Because a dummy tooth was counted, EAC does not enter Halt Mode and Tooth Tick Counter continues incrementing in the absence of a physical tooth detection.

In case of extreme acceleration on very low RPM (cold start) there can be a situation that the first physical tooth after one or two missing teeth appears even before the "dummy" tooth is generated. Due to the acceleration the EAC switches to High Rate mode in order to run through all the valid angle values, including the dummy teeth. When the Tooth Tick Counter reaches the TICKS value on High Rate mode, and the "dummy tooth" down counter is not zero, the generated "dummy tooth" advances to the next tooth and decrements the "dummy tooth" counter, but does not switch the EAC back to Normal mode. The last "dummy tooth" decrements the counter to zero, indicating that no more dummy teeth are to be inserted, and the next tooth is an estimated physical tooth. The EAC continues at High Rate mode until the Tooth Tick Counter reaches the TICKS value again, then advances to the next tooth while switching back to Normal mode. When in High Rate mode, the TPR register does not reflect the MISSCNT downcounting, see TPR Buffering for details.

MISSCNT can be rewritten before it reaches 0, allowing it to count more than 7 missing teeth, as long as no physical tooth arrives between the first MISSCNT write and the rewrite.

43.14.7.6.3 Combining Missing Teeth and Last Tooth

The Last Tooth indication takes effect when there are no more missing teeth to be generated, i.e the "dummy tooth" counter value is zero. If, for example, the microcode sets the missing teeth counter to "10" (two missing teeth) and sets the LAST flag, the first and the second dummy teeth will increment TCR2, and the third estimated tooth, which correlates with the physical tooth (the first of the next cycle), will reset TCR2, because LAST was set. This scheme enables the microcode to define one or more missing teeth to be replaced by "dummy tooth" insertion, and the end of the engine cycle in one service request. It is assumed that the two missing teeth must come together in the same engine cycle, and not split between two engine cycles (either the missing teeth are both last in an engine cycle or both not last, but not last in one engine cycle and first in the next).

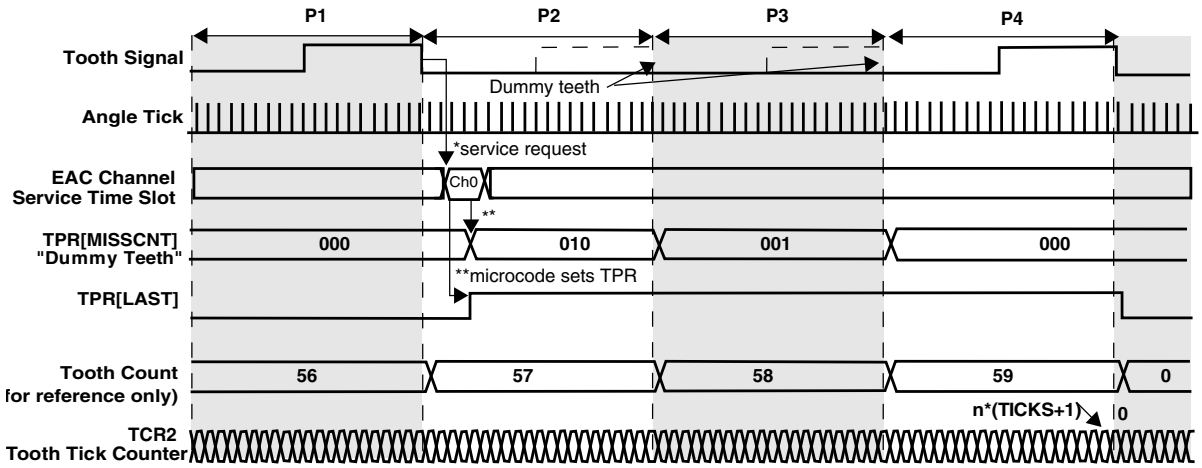


Figure 43-328. Missing Teeth and Last Tooth Combination

43.14.7.7 Handling Mechanical Tooth Correction

The EAC can handle tooth edge detection errors caused by flywheel mechanical errors. The eTPU application can hold a vector of tooth mechanical errors with one entry per tooth. This error can be measured in angle ticks which are independent of engine RPM. The TRR can be updated to the fixed period of any tooth, including its mechanical error.

Because TCR2 counts continuously, without being reset, the mechanical correction is transparent. Though the tooth has its own programmed TICKS value, TCR2 simply counts angle ticks, disregarding the boundary between two adjacent teeth.

43.14.7.8 Handling Mis-detected Tooth

When a physical tooth signal is missed by the engine sensor, the EAC may get into Halt mode at the end of the estimated tooth period, expecting the physical arrival. In this case, a Match timeout event of EAC channel will call service which detects extreme deceleration. The microcode can assert the IPH bit in TPR, to force the detection of the missed physical tooth. It can also calculate the accumulated angle bus error, and fix the next estimated tooth period, to close the gap.

43.14.7.9 Handling False Tooth Detection

Most of the false tooth detection, caused by noises on the engine tooth sensor, can be eliminated by the window blanking filtering, timed by EAC channel match recognitions. The EAC also provides means of fixing false detection of an additional tooth which passed the window filter. When such an event occurs, the EAC switched to High Rate

mode (advancing to the next tooth) and when the next physical tooth arrives, an extreme acceleration is detected: the EAC sees the remaining portion of the current tooth period as another tooth period. The microcode can detect the situation when the acceleration in not realistic, or when immediately after the detection of this extreme acceleration, the following tooth indicates extreme deceleration back to the original RPM.

When the microcode detects such a case, the Tooth Counter has been advanced by mistake to the next tooth. The microcode can set the HOLD bit in the TPR, forcing the EAC to freeze and wait for the next physical tooth to close the gap. When the next physical tooth arrives, HOLD is automatically negated and the EAC proceeds from that point to the remaining portion of the tooth period, in the same mode it was when HOLD bit was asserted.

43.14.7.10 Angle Logic and Channel Modes

The TCRCLK transition detection is qualified by a signal that comes from channel 0 (see Figure 43-323) and depends on the particular mode (PDCM) programmed for that channel. This configures a window for TCRCLK detection for the angle logic which is the same (except on High Rate mode, see High Rate Mode (Acceleration)) used to set the TDLA flag on single transition modes, and TDLB on double transition modes (see signals TSE1,TSE2 in Figure 43-306). The same applies when channels 1 or 2 are used to control EAC (see signals TS1, TS2 in Figure 43-306) . As a consequence, the window depends on the channel mode as follows:

- on all modes, the window closes upon a tooth edge detection: TDLA asserted on single transition modes, TDLB asserted on double transition modes.
- on mode m2_st: the window opens on Match A (which enables Transition A) and does not close with Match B. If Match B comes before Match A, it blocks Match A and, hence, Transition A.
- on mode m2_o_st: the window opens on Match A (which enables Transition A) and closes on Match B. Match B is enabled by Match A, so it cannot come before.
- on all other Single Transition modes, the window is "always open", independently of matches.
- on mode m2_dt: the window opens on Transition A, which is enabled by Match A. The window does not close with Match B, but if it comes before Match A the later gets blocked and, hence, blocks Transitions. Match A is also a condition for the window, so the microcode closes it by clearing MRLA.

- on mode m2_o_dt: the window opens on Transition A, which is enabled by Match A. The window closes on Match B, which is enabled by Match A. Match A is also a condition for the window, so the microcode closes it by clearing MRLA.
- on all other Double Transition modes, the window opens on Transition A.

43.14.7.11 Restarting Angle Logic

It is not recommended to change the ETPUTBCR field AM while GTBE=1. However, if the Angle Logic must be restarted without interfering with the timebase count running on TCR1, the procedure below must be followed:

1. write ETPUTBCR setting AM=00 and TCR2CTL=111 at once. That prevents TCR2 from incrementing while the Angle logic is disabled. The Angle Logic state-machine resets to Normal mode and the tick prescaler to the initial count with AM=00, but not the microengine registers TPR and TRR.
2. start a thread to reconfigure the EAC. The thread must set the EAC controlling channel (0, 1 or 2) flags in a state, depending on the channel mode, that lets the channel tooth detection window open (see Angle Logic and Channel Modes). It can optionally write TCR2 with an angle preset value equivalent to the first tooth expected after restart. The thread must also set TPR bit HOLD=1. The TPR bit IPH must be 0.
3. after the thread is finished, write ETPUTBCR setting AM=01, 10 or 11, and TCR2CTL according to the desired tooth edge selection if AM=01 .

The first tooth detected after this procedure restarts the TCR2 counting, unfreezing the Angle Mode logic into normal mode.

43.14.7.12 Special TPR Write Cases

This section describes how simultaneous modification of TPR fields are resolved, and how the effect of TPR writes depend on the EAC mode.

43.14.7.12.1 TPR Buffering

In High Rate mode (see High Rate Mode (Acceleration)), TPR writes are immediately effective only for bits IPH and HOLD. Writes to all other fields are "buffered" and become effective when EAC leaves High Rate mode. However, if TPR is written a

second time right after IPH is asserted in Normal mode, this second write behaves as if EAC is still in Normal mode. Only in the next microcycle (after execution of a nop, for instance) the TPR writes are buffered, acknowledging High Rate mode.

MISSCNT and LAST can be written any value during High Rate mode, and the value that prevails for the next tooth is the one sampled when EAC goes back to Normal mode (or the value written in Normal or Halt mode thereafter). If MISSCNT and/or LAST are not zero when High Rate mode begins, they are sampled into the internal EAC logic and are effective while High Rate lasts (missing teeth count continues, and TCR2 is reset at the end of High Rate if LAST=1). However, their values in TPR are reset when High Rate mode starts. After that and until the end of High Rate mode, the value read by microcode is the same written. This behavior prevents read-modify-writes to TPR from unwillingly rewriting LAST or MISSCNT.

43.14.7.12.2 IPH and LAST

If both IPH and LAST are asserted in the same microinstruction with HOLD=0, the EAC acts as if LAST was set first and then IPH right after, so that:

- in Normal mode, it goes to High Rate with LAST=1;
- in Halt mode, it goes to Normal Mode resetting LAST (and TCR2);
- in High Rate mode, IPH is ignored and LAST becomes effective in the next tooth (physical or inserted) after it goes back to Normal mode.

43.14.7.12.3 IPH and TICKS

Because of different results depending on the EAC mode at the time of TPR write, it is not advisable to write 1 to IPH and change TICKS at the same microinstruction. A consistent behavior is obtained if IPH is written first and TICKS on the second microinstruction after (for instance with a nop between them), making the new TICKS value valid for the next tooth regardless of the mode.

The mode-dependent behavior is (assuming HOLD=0):

- in Normal mode, the new TICKS value becomes valid before EAC goes High Rate due to the IPH;
- in Halt mode, the EAC goes to Normal mode, and new TICKS is valid for the next tooth;
- in High Rate mode, the new TICKS value is effective when EAC leaves High Rate mode, and IPH is ignored;

43.14.7.12.4 IPH and MISSCNT

If both IPH and MISSCNT are written non-zero values with HOLD=0:

- in Normal mode, at the next microcycle the EAC goes to High Rate mode, the MISSCNT field in TPR goes to 0, and the missing teeth are counted in High Rate mode.
- in Halt mode, the EAC goes to Normal mode for one microcycle and then, yet another microcycle later, goes to High Rate mode, counting the missing teeth. The TPR fields IPH and MISSCNT are zeroed on the transition from Normal to High Rate mode.
- in High Rate mode, IPH is ignored (resetting at the next microcycle) and MISSCNT is buffered (see TPR Buffering).

43.14.7.12.5 IPH and HOLD

If IPH and HOLD are asserted at once, IPH cancels the HOLD and both reset. The EAC is not frozen, regardless of the EAC mode (High Rate, Normal or Halt).

43.14.7.12.6 LAST and HOLD

If LAST and HOLD are written 1 at once, LAST asserts and EAC is frozen. Afterwards, when a physical tooth is detected or IPH is asserted, the EAC is unfrozen in the same state it was before, and LAST is kept asserted.

43.14.8 Microengine

Each eTPU engine has a microengine that fetches, decodes and executes microinstructions. The Microengine only works when there are service requests to be attended, otherwise it turns to idle state, controlled by Hardware Scheduler (see Scheduler).

Microcode is stored in Shared Code Memory (SCM) that is 32-bit wide. Microengine can access SDM using a different bus from the one used to accesses code memory, so that code and data can be accessed at the same time (Harvard Architecture).

Some of eTPU functionality can only be made through the microengine, like configuring channels and interrupting host. Microengine gives eTPU a high degree of flexibility, since any desired treatment for channel's events can be implemented; however, that

flexibility comes at the cost of channel service's latency. Latency is worsened when channels from a same eTPU engine contend for microengine service. In the figure below, a block diagram of microengine architecture is shown.

Follows a summary of Microengine features:

- P, DIOB, A, B, C, D, SR, RAR, LINK, CHAN, MACL, MACH, ERTA, ERTB, TCR1, TCR2, TPR, TRR registers are accessible by microcode.
- 24-bit ALU and Post-ALU shifter performs basic arithmetic and logical operations described in ALU and Post-ALU Shifter.
- MDU (MAC/Divide Unit) performs integer MAC, multiply and divide operations.
- Fixed Microinstruction Size of 32bits.
- Fixed-length instruction execution (2 eTPU clocks)
- Static superscalar operation

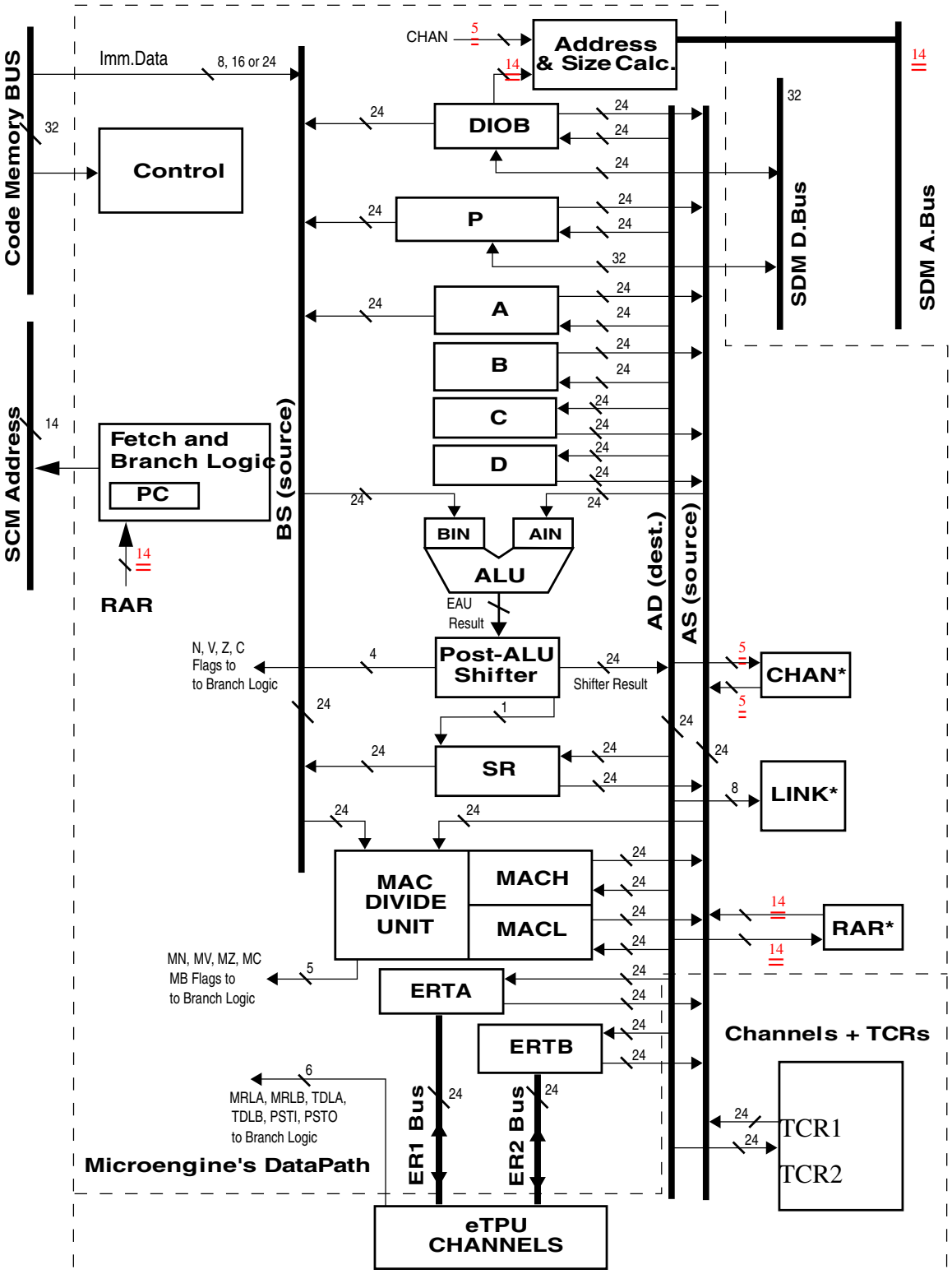


Figure 43-329. Microengine Block Diagram

43.14.8.1 Registers

eTPU microengine accesses a total of 18 registers. Fourteen of them are special purpose (registers A, B, C and D are for general use). Special purpose registers except CHAN and LINK can also be used as general use if the operation that use their contents are not performed. Register description is intended to just introduce their functionalities and not to provide detailed explanation of it since it will be described in Microinstruction Set . Registers less than 24 bits in size are right-justified.

None of the registers have guaranteed reset values. However, some are initialized just before the thread starts (see Time Slot Transition).

43.14.8.1.1 P Register

P register is the only one that is 32-bit wide. It can be used as source and destination for arithmetic/logical operation, and as source and destination for SDM read/write operations.

For P source/destination possibilities in ALU/MDU microoperations, see Selecting Sources and Destination.

When P is used as SDM read/write operations source or destination there are only 3 possibilities of access: all 32 bits, lower 24 bits and upper 8 bits. SDM operations are explained in detail in SDM Microoperations.

P is automatically loaded with one parameter before the thread starts (parameter preload). For more information see Entry Point Format and Time Slot Transition.

Upper 8 bits of P register can be used as application state, since these bits can be tested as branch conditions. P[31:24] is also used in dispatch microoperation (see Dispatch Microoperation), and bit pairs P[29:28], P[27:26], P[25:24] can be directly copied into Channel flags 1 and 0 using field FLC. Together with Entry Table Condition Encoding, it provides fast state resolution without code execution.

43.14.8.1.2 DIOB Register

DIOB is an abbreviation for Data Input/Output Buffer. DIOB is 24-bit wide and can be used as source and destination for arithmetic/logical operations as well as SDM data source and destination. DIOB only can be accessed as 24 bits, both in arithmetic/logical and SDM read/write operations. When using DIOB to perform an SDM access, only lower 24 bits of SDM will be accessible (SDM upper 8 bits always remains unchanged).

DIOB can also be used as SDM addressing register, when DIOB contents is used as absolute SDM address (14-bit wide). In this case DIOB can also be pre-decremented or post-incremented (see Indirect Addressing Mode).

DIOB is automatically loaded with one parameter before the thread starts (parameter preload). For more information see Entry Point Format and Time Slot Transition.

43.14.8.1.3 ERTA and ERTB Registers

ERTA/B registers are 24-bit wide and can be used as source or destination in arithmetic/logical operations. ERTA/B are the only source for channel's match registers write (see Write Channel Match and UDCM Registers). ERTA can also be the source for UDCM write.

When a thread starts to be executed, ERTA and ERTB are loaded with a copy of CaptureA and CaptureB registers respectively. ERTA/B can be used to receive a copy of MatchA and MatchB registers. ERTA/B are the only destination of MatchA/B read operation (see Special T4ABS Source Operation: Read Match Registers).

ERTA and ERTB also receive a copy of CaptureA and CaptureB registers when CHAN register is written (see CHAN Register). For more information about Capture and Match registers see MatchA and MatchB Registers and CaptureA and CaptureB Registers.

43.14.8.1.4 SR register - Shift Register

SR is a 24-bit wide register that can be used as source and destination register for arithmetic/logical operations. SR can shift right its contents by 1 bit at time and, at the same time, receive in its bit 23 the lost bit of a shift-right operation in post-ALU shifter (ALU and Post-ALU Shifter), allowing SR to be used to perform 48-bit shift right (see Shift Operations).

43.14.8.1.5 MACH and MACL Registers

Both MACH and MACL are 24-bit registers, part of MAC/Divide unit (see MAC and Divide Unit (MDU)). They can be used as source and destination in most arithmetic/logic operations. When multiply or divide operations are used (multiply-accumulate included), MACH and MACL have special purpose and some restrictions apply, see MAC and Divide Unit (MDU) for more information.

43.14.8.1.6 LINK Register

Link Register is an 8-bit wide register and can be used only as destination in arithmetic operations. LINK is a write-only command register, which precludes its use as a source register for ALU operations. When LINK register is written, it issues a service request for the channel number and eTPU engine equal to the number written in LINK register (see Functions and Threads and Channel Link for information about Link Service Request).

43.14.8.1.7 RAR Register

RAR is a 14-bit register and can be used as source and destination in arithmetic operations. RAR also receives the contents of PC register when a subroutine call is executed. The contents of RAR register is loaded into PC when a return from subroutine is executed. RAR is loaded with value 0x3FFF during TST. For more information about subroutine call and return see Branch Operations and Return From Subroutine respectively.

43.14.8.1.8 CHAN Register

CHAN is a 5-bit register that can be used as source and destination in arithmetic operations. The contents of CHAN register affects the execution of many channel-related microinstructions, because its number indicates the **selected channel** . CHAN register must not be used to store temporary values in arithmetic operations. For more details, refer to Channel Selection Register - CHAN.

43.14.8.1.9 Counter Registers: TCR1, TCR2, TPR and TRR

All these registers are 24-bit wide except TPR, which is a 16-bit register. They can be read or written in arithmetic/logical operations, and have special-purpose uses for time base and angle mode operations. For more information about those registers see Time Bases and EAC - eTPU Angle Counter.

43.14.8.1.10 General Purpose Registers: A, B, C and D

A, B, C and D are 24-bit general purpose registers, which can be used to store intermediate values and do not have other specific uses with any eTPU feature.

43.14.8.2 ALU and Post-ALU Shifter

The ALU executes 24-bit arithmetic and logical operations. ALU's output goes directly to a 1-bit shifter, called post-ALU shifter, so it is possible, for example, to add and shift using only one microinstruction.

In some microinstruction formats, it is not possible to specify the operation executed by ALU. In these cases ALU will always perform addition.

In formats which have the field ALUOP for ALU operation selection, all of them can be performed, including add/subtract using C (carry) flag as ALU's carry-in, bitwise AND/OR/NOT/XOR, and shift/rotate of 2, 4, 8 and 16 bits. See ALU/MDU Operation Selection.

Subtraction, inversion, increment and decrement can be performed by combinations of source inversion and setting ALU's carry-in to 1.

ALU always performs 24-bit operations on its inputs, called **A-source** and **B-source**, and outputs a 24-bit result. 8-, and 16-bit inputs are zero padded to 24 bits. Likewise, ALU 24-bit output is always truncated to the destination register size.

43.14.8.2.1 ALU Flags

Four flags - Carry, Negative, Overflow, Zero - described below, are related to ALU and post-ALU shift operations. Operation size and shifting affect flags generation logic. Operation size determines the result boundary to be used for flags generation. Operation size is determined by size of sources and destination (see Flags Sampling Control). For more information about flag generation, see Flags Sampling Control . ALU flags can be used as branch condition (see Conditional/Unconditional Branch) or conditional ALU/MDU operation (see Conditional ALU/MDU Operation Execution).

Field CCS/CCSV in microinstructions can force **no update** of all flags. Not all flags are updated in all ALU operations: Overflow is updated only on addition and absolute value operations, Carry flag is updated in most ALU operations, and only Zero and Negative are updated in all ALU operations.

ALU flags are never updated when microinstruction starts an MDU operation, regardless of CCS/CCSV, but are updated normally afterwards, on ALU operations that are executed in parallel with an ongoing MDU operation (MDU has its own flags).

Note

Operation size can be smaller than destination register. For example: 0xFFFF + 0x0001 (both 16-bit sources) stores 0x10000 in a 24-bit register and sets Zero and Carry flags because operation size is 16 bits.

43.14.8.2.1.1 Carry Flag (C)

In an unsigned addition without shifting, Carry Flag is the ALU carry from bit 7 to 8, 15 to 16, or 23 to 24 on 8-, 16- and 24-bit operation sizes respectively. In an unsigned subtraction without shifting, Carry Flag represent the sign of ALU's result considering operation size (Carry Flag equal to 1 means a negative result).

Carry Flag definition is operation-dependent. The Carry flag in add/subtraction with Post-ALU shift is defined in Table 43-360 . Find the definitions for other operations in the following sections.

43.14.8.2.1.2 Negative flag (N)

Negative flag indicates the sign of result based on the operation size, regardless of the operation performed, as shown in the following table.

Table 43-356. Negative (N) flag behavior

Operation Size	Value
8 bits	N = result[7]
16 bits	N = result[15]
24 bits	N = result[23]

Note

The N flag may not reflect the sign of the value actually written into the destination register, if it does not have the same size of the operation (see Flags Sampling Control). This is always the case for registers RAR (14 bits) and CHAN (5 bits).

43.14.8.2.1.3 Overflow (V)

Overflow is updated only on addition (with or without carry) and absolute value operations. In signed operations, overflow flag indicates that the result of arithmetic operation (add or subtraction) can not be represented by a word of the size of the operation. Overflow flag behavior for addition is defined in the following table. Overflow flag for Absolute operation is explained in Absolute Value Operation . V Flag is calculated using ALU adder output (i.e., it is not affected by 1-bit shift/rotate operations).

Table 43-357. Overflow Flag on Addition - V

Op. Size	Value[1]
8 bits	(AS[7] & BS[7] & !alu_adder_output[7]) | (!AS[7] & !BS[7] & alu_adder_output[7])
16 bits	(AS[15] & BS[15] & !alu_adder_output[15]) | (!AS[15] & !BS[15] & alu_adder_output[15])
24 bits	(AS[23] & BS[23] & !alu_adder_output[23]) | (!AS[23] & !BS[23] & alu_adder_output[23])

1. BS is taken after any inversion by the BINV field, but not added to the carry bit (CIN field)

NOTE

For V-flag definition on the absolute operation, see “Absolute Value Operation".

43.14.8.2.1.4 Zero Flag (Z)

Zero flag equal to 1 indicates that the result from the ALU, limited to the operation size, is zero, regardless of the operation performed, whether the result is written, or where it is written. It depends on the operation size, as shown in the following table.

Table 43-358. Zero Flag - Z

Operation Size	Value
8 bits	Z = (result[7:0] == 0x00)
16 bits	Z = (result[15:0] == 0x0000)
24 bits	Z = (result[23:0] == 0x000000)

43.14.8.2.2 ALU ADD Operation with and without shifting

ADD operation is selected by ALUOP or ALUOPI fields and when none of them is available in a microinstruction format[30] . Optionally, result can be shifted or rotated right by 1 bit, which is selected by SHF, ALUOP or ALUOPI fields. See Microinstruction Set for more details. The following table describes how CIN and BINV fields change ADD operation behavior.

Table 43-359. Types of ADD operations

BINV	CIN	Operation (adder output)
1	1	AS + BS
1	0	AS + BS + 1
0	0	AS - BS
0	1	AS - BS - 1

ALU adder output can be 1-bit shifted or 1-bit rotated right as follows:

Shift right:

```
if BINV==1
  result [23:0] = adder_output[24:1]
else
  result [23:0] = adder_output [24:1] xor 0x800000
endif
```

Shift left:

30. ALU operations only occur on formats where a destination field is found (T2ABD/T2D).

```
result[23:1] = adder_output[22:0]
result[0]    = 0
```

Rotate right:

```
case(opsize/CCSV)
8-bit:
  result[6:0]   = adder_output[7:1]
  result[7]     = adder_output[0]
  result[23:8]  = adder_output[23:8]
16-bit:
  result[14:0]  = adder_output[15:1]
  result[15]    = adder_output[0]
  result[23:16] = adder_output[23:16]
24-bit:
  result[22:0]  = adder_output[23:1]
  result[23]    = adder_output[0]
```

Note

Only for the Post-ALU rotate right, the operation size is determined by the field CCSV (see Flags Sampling Control). For example: if CCSV=00, T4ABS=P (24-bits), T4BBS=A (24 bits), T2ABD=B (24 bits), and ALUOP="Add ROR", then B gets A+P with bits 7:0 rotated, even though the operation size is 24 bits.

The following table describes Carry flag behavior.

Table 43-360. Carry flag update on ADD operation

BINV[1]	Op. Size	shift/rotate	Value
1	8 bits	none	adder carry from bit 7 to bit 8
1	16 bits	none	adder carry from bit 15 to bit 16
1	24 bits	none	alu_adder_output[24]
0	8 bits	none	!adder carry from bit 7 to bit 8
0	16 bits	none	!adder carry from bit 15 to bit 16
0	24 bits	none	!alu_adder_output[24]
0 or 1	8 bits	shift left	alu_adder_output[7]
0 or 1	16 bits	shift left	alu_adder_output[15]
0 or 1	24 bits	shift left	alu_adder_output[23]
0 or 1	x	shift right	alu_adder_output[0]
1	8 bits	rotate right	adder carry from bit 7 to bit 8
1	16 bits	rotate right	adder carry from bit 15 to bit 16
1	24 bits	rotate right	alu_adder_output[24]
0	8 bits	rotate right	!adder carry from bit 7 to bit 8
0	16 bits	rotate right	!adder carry from bit 15 to bit 16
0	24 bits	rotate right	!alu_adder_output[24]
		except on max-constant generation (see Max Const Generation With T4BBS=111)	

1. BINV has no effect on carry-out when used to code max constant (see Max Const Generation With T4BBS=111).

Flags N and Z on shift are updated according to the result after shift. Flag V with shift is updated according to the ADD operation only, the same way as without shift.

43.14.8.2.3 ADC Operation

ADC operation is selected by the ALUOP field. CIN field is ignored when this operation is selected. The following table describes how BINV change ADC operation behavior.

Table 43-361. Types of ADC operations

BINV	CIN	Operation
1	x	AS + BS + C flag
0	x	AS - BS - C flag

Flags behave exactly the same way as for ADD operation without shift/rotate.

43.14.8.2.4 Bitwise Operations

Bitwise AND, OR and XOR are selected by ALUOP field. On these operations CIN field is ignored and BINV field inverts (bitwise NOT) BS. C and V Flags are never updated on these operations. The following table describes AND, OR and XOR bitwise operations.

Table 43-362. Types of Bitwise Operations

ALUOP	BINV	Operation
10000	1	AS | BS
10000	0	AS | (~BS)
10001	1	AS ^ BS
10001	0	AS ^ (~BS)
10010	1	AS & BS
10010	0	AS & (~BS)

43.14.8.2.5 Set Bit / Clear Bit Operations

These operations set or clear the AS bit determined by BS[4:0]. If the bit number resolves to a value greater than 23, no bit is set or cleared (i.e., result is equal to AS). On these operations CIN field is ignored and BINV field inverts (bitwise NOT) BS. C and V flags are never updated for set/clear bit operations. These operations **override** B-Source size to 8 bits.

set bit (BINV = 1):

```
result = AS | (1 << BS)
```

clear bit (BINV = 1):

```
result = AS & ~(1 << BS)
```

set bit (BINV = 0):

```
result = AS | (1 << (31 - BS))
```

clear bit (BINV = 0):

```
result = AS & ~(1 << (31 - BS))
```

43.14.8.2.6 Exchange Bit

Exchange the AS bit determined by BS with C flag. If the bit number resolves to a value greater than 23, no exchange is performed (i.e., result is equal to AS and C flag is not updated). This operation **overrides** BS size to 8 bits. On this operation, CIN field is ignored and BINV field inverts (bitwise NOT) BS. V flag is never updated on exchange bit operation. C flag is always updated, regardless of CCSV, unless BS > 23.

Exchange Bit (BINV=1):

```
if BS <= 23
begin
  temp_C_flag = AS[BS]
  if C_flag == 1
    result = AS | (1 << BS)
  else
    result = AS & ~(1 << BS)
  C_flag = temp_C_flag
end
```

Exchange Bit (BINV = 0):

```
if (31 - BS) <= 23
begin
  temp_C_flag = AS[31 - BS]
  if C_flag == 1
    result = AS | (1 << (31 - BS))
  else
    result = AS & ~(1 << (31 - BS))
  C_flag = temp_C_flag
end
```

43.14.8.2.7 Multibit shift/rotate Operations

These operations shift or rotate AS by 2, 4, 8 or 16 bits. Size of shift/rotate is determined by BS[1:0]. The following table describes the number of shifted/rotated bits depending on BS[1:0] value.

Table 43-363. Number of shifted/rotated bits for each BS[1:0] value

BS[1:0]	Bits Shifted/Rotated
0	2
1	4
2	8
3	16

Shift right is a logical operation (i.e., zeros are inserted on left). Multibit shift and rotate operations **overrides** BS size to 8 bits. The shifts and rotate operate on 24 bits, independently of the operation size.

V flag is never updated for multibit shift or rotate operations. Carry flag behavior is described in the following table. CIN is ignored in these operations, but BINV is effective.

Table 43-364. Carry flag value on multibit shift/rotate operations

ALUOP	BS[1:0]	C flag value[1]
11001 (shift left)	0	AS[22]
11001 (shift left)	1	AS[20]
11001 (shift left)	2	AS[16]
11001 (shift left)	3	AS[8]
11010 (shift right)	0	AS[1]
11010 (shift right)	1	AS[3]
11010 (shift right)	2	AS[7]
11010 (shift right)	3	AS[15]
11011 (rotate right)	0	AS[2]
11011 (rotate right)	1	AS[4]
11011 (rotate right)	2	AS[8]
11011 (rotate right)	3	AS[16]

1. CCS/CCSV can disable flag update on multibit shift/rotate, but the specified flag size in CCSV is ignored for the C flag.

43.14.8.2.8 Absolute Value Operation

Absolute Value operation is selected by ALUOP field. On this operation, AS is interpreted as a signed number and its absolute value is the result. V and N flags are updated with the result signal determined by the operation size. AS bit 23 after size override and sign extension (if any, see A-Source Size Override), regardless of A-source register size, is used to check the operand signal and is copied to C-flag. Note that if AS is 8-bit or 16-bit, its sign is taken into account and copied to C only if sign-extension is performed. The following table summarizes flag updating for Absolute Value operation.

This operation is independent of B-source. Instruction fields T4BBS, BINV and CINV are ignored in this operation. The Absolute Value operation size is the minor between A-source size and Destination size.

Table 43-365. ALU Flags in Absolute Value operation

operation size	V, N[1]	C	Z
8	alu_output[[7]	AS[23]	alu_output[7:0] == 0
16	alu_output[15]		alu_output[15:0] == 0
23	alu_output[23]		alu_output[23:0] == 0

1. V, N can be 1 on 8- and 16-bit Absolute Value, because the operand sign is always taken from bit 23. V, N can also be 1 in 23-bit Absolute Value (or 8-bit and 16-bit with sign extension), if the operand is 0×800000 (0×80, 0×8000).

43.14.8.3 MAC and Divide Unit (MDU)

MDU is an autonomous resource in the microengine which can carry out sequential multiply, multiply-accumulate, fractional multiplication and divide operations, selected through the microinstruction fields ALUOP or ALUOPI. The unit supports signed and unsigned multiply and fractional multiplication of any combination of 8, 16 or 24 bit operands [31], and also signed and unsigned 24-bit multiply-accumulate. Divide operation is unsigned, and both operands are always 24-bit wide.

Depending on the size of operands and the type of operation, MDU can take more than one microcycle to execute the operation, but microengine continues to execute microinstructions in parallel. When the microcode issues an END command, any MDU executing operation terminate immediately and is left incomplete. When selecting an operation that uses MDU, the result is always placed in MACH and MACL registers, and the register selected as destination does not have its value changed (Selecting Sources and Destination). During calculations, MACH and MACL holds temporary values and should not be written, otherwise the result is unspecified. One must not start an MDU operation while MDU is already busy: the result is unspecified for both the ongoing operation and the started one.

MDU Operations update its own set of 5 flags, described in MDU Flags. MDU operations never update C, N, V and Z flags. CIN and BINV microinstruction fields affect MDU operations according to the follwoing table.

31. There is no distinct selection of 24-bit fractional multiplication, for it works exactly as a 24-bit ordinary multiplication.

Table 43-366. CIN and BINV with MDU Operations

B-source operand	BINV	CIN	Op. Performed
signed	1	1	AS mdu_op BS
	0	0	AS mdu_op (-BS)
	1	0	reserved
	0	1	reserved
unsigned[1]	1	1	AS mdu_op BS
	1	0	AS mdu_op (BS+1)
	0	x	reserved

1. includes the B-source (unsigned) in fmults (signed) operations.

43.14.8.3.1 Multiply and Multiply-Accumulate Operation Length

MDU needs 2 sources, A source and B source, to perform an operation. The time needed to perform a multiply or multiply-accumulate is:

- on 24-bit x 8 bit multiplies: 2 microcycles (one start-MDU plus one execution microcycle)
- on 24-bit x 16 bit multiplies: 3 microcycles (one start-MDU plus two execution microcycles)
- on 24 bit x 24 bit multiplies/macs: 4 microcycles (one start-MDU plus three execution microcycles)

An internal pipeline in MDU allows multiply-accumulate (or even non-fractional multiply) operations to start one microinstruction before a multiplication or multiply-accumulate (signed or unsigned) has been completed (e.g., one can start one multiply or multiply-accumulate once every three microinstructions). However, by doing that it is not possible to read the result in MACH and MACL (although the MDU flags can be tested), so this is intended to be used in a multiply-accumulate sequence. It is also allowed to mix different sizes in multiply/mac sequences.

Multiply-accumulate operations are similar to multiply operations, except that the contents of MACH and MACL registers are added to the multiplication result.

When multiply or multiply accumulate operations finish, MACL and MACH hold the least and the most significant 24-bit words, respectively.

43.14.8.3.2 Divide Operation Length

The divide operation is always unsigned. The division completes in 13 microcycles, meaning that after the start divide microinstruction, one has to wait for 12 microcycles and then read the result and the remainder in MACH and MACL registers. During the 12 execution microcycles, microengine can execute microinstructions unrelated to the MDU.

43.14.8.3.3 Signed Multiplication (mults)

MDU signed multiplication is defined as follows:

```
(signed) MACH,MACL = (signed) AS * (signed) BS
```

MC and MV flags are reset. MZ is set if result is 0, resets otherwise. MN is set if result is negative.

43.14.8.3.4 Unsigned Multiplication (multu)

MDU unsigned multiplication is defined as follows:

```
(unsigned) MACH|MACL = (unsigned) AS * (unsigned) BS
```

MC and MV flags are reset. MZ is set if result is 0, resets otherwise. MN is a copy of the most significant bit of result.

43.14.8.3.5 Signed Multiply-Accumulate (macs)

MDU signed multiply-accumulate is defined as follows:

```
(signed/unsigned) {MACH,MACL} += (signed) AS * (signed) BS
```

MC is not altered.

MV is set if result can not be represented by a 48-bit signed number. MACS never resets MV flag: it is left as is if no overflow occurs, or set it otherwise. This allows checking the overflow flag only once at the end of a series of multiply-accumulate operations in a scalar product calculation.

```
if (({MACH,MACL} += AS * BS < -2^47) || ({MACH,MACL} += AS * BS > 2^47 - 1))
  MV = 1
```

MZ is set if result is 0, resets otherwise. MN is a copy of the most significant bit of result.

Note that only 24-bit multiply-accumulate is available.

43.14.8.3.6 Unsigned Multiply-Accumulate (macu)

MDU Unsigned Multiply-Accumulate is defined as follows:

```
(signed/unsigned) {MACH,MACL} += (unsigned) AS * (unsigned) BS
```

MC is set if result can not be represented by a 48-bit unsigned non-negative number. MACU never resets MC flag: MC flag is left as is if no carry occurs, or set otherwise. This allows checking the carry flag only once at the end of a series of multiply-accumulate operations in a scalar product calculation.

```
if (({MACH,MACL} += AS * BS < 0) || ({MACH,MACL} += AS * BS > 2^48 - 1))
  MC = 1
```

MV is not altered.

MZ is set if result is 0, resets otherwise. MN is a copy of the most significant bit of result.

Note that only 24-bit multiply-accumulate is available.

43.14.8.3.7 Signed Fractional Multiplication (fmults)

MDU Signed Fractional Multiplication takes the B-Source as an **unsigned** 8- or 16-bit[32, 32] fraction between 0 and $(2^8 - 1)/2^8$ (inclusive) for the 8-bit operation, or between 0 and $(2^{16}-1)/2^{16}$ (inclusive) for the 16-bit operation. Only A-Source is taken as a signed number. The value of B-Source is considered the unsigned numerator of a fraction with denominator 2^8 or 2^{16} for the 8- and 16-bit operations, respectively.

The integer part of the result is stored in MACH, and the fractional part in MACL. The result is signed, so that the concatenation of MACH and MACL form a 48-bit fixed point number with a 24-bit integer part, both for 8- and 16-bit operations. To calculate the unsigned numerator of the fractional part (with denominator 2^{24}) of the result, one must take the absolute value of MACL considering the signal of the result (not MACL alone), i.e.: if flag MN=1, invert MACL and add 1.

MDU flags are updated in the same way as in the Signed Multiplication.

43.14.8.3.8 Unsigned Fractional Multiplication (fmultu)

MDU Unsigned Fractional Multiplication takes both A-Source and B-Source as unsigned operands. B-Source is taken as an 8- or 16-bit[1] fraction between 0 and $(2^8 - 1)/2^8$ (inclusive) for the 8-bit operation, or between 0 and $(2^{16}-1)/2^{16}$ (inclusive) for the 16-bit operation. The value of B-Source is considered the numerator of a fraction with denominator 2^8 or 2^{16} for the 8- and 16-bit operations, respectively.

32. 24-bit fractional signed/unsigned multiplication is equivalent to mults/multu.

The integer part of the result is stored in MACH, and the fractional part in MACL. The fractional part in MACL is the numerator of a fraction with denominator 2^{24} . The concatenation of MACH and MACL form a 48-bit fixed point number with a 24-bit integer part, both for 8- and 16-bit operations.

MDU flags are updated in the same way as in the Unsigned Multiplication.

43.14.8.3.9 Unsigned Divide (div)

At the end of a divide operation MACL holds the result of the division, taking A-source as numerator and B-source as denominator, while MACH holds the remainder. If a divide by 0 is executed, MACL holds the maximum unsigned number (0xFFFFFF) as result and flag MV is set to indicate division by 0 (otherwise reset). The contents of MACH become indetermined.

MC flag is always reset.

MZ flag is set if MACL equals 0, and reset otherwise.

MN receives a copy of MACH bit 23 (msb from the remainder).

Note that signed division is not available.

43.14.8.3.10 MDU Flags

MDU has its own flags to indicate the result and status of an MDU operation. They are: MC, MZ, MV, MN and MB. All MDU flags are updated with the final result at the end of the operation, and do not change until the next operation finishes. Therefore it is possible to start a new MDU operation and test the flags of the previous one in parallel, except for mult/mac with 8-bit operand (takes only 1 microcycle).

43.14.8.3.10.1 MDU Negative Flag - MN

MN flag is always a copy of MACH bit 23 at the end of the operation, either in signed or unsigned ones. Note that MACH holds the rest of a division operation, which is always unsigned.

43.14.8.3.10.2 MDU Carry Flag - MC

MDU carry flag indicates if the result cannot be represented by a 48-bit number, in Signed and Unsigned Multiply Accumulates. It is reset in the other operations.

43.14.8.3.10.3 MDU Zero Flag - MZ

In multiply and multiply-accumulate operations, MDU zero flag is asserted if MACH and MACL are equal to zero at the end of an operation. In divide operations, zero flag is asserted if MACL (result) is equal to 0.

43.14.8.3.10.4 MDU Overflow Flag - MV

In multiply operations, MV flag is negated and keeps negated in the end, because the result of a multiplication can always fit in a 48-bit result (MACH and MACL concatenated). In a multiply-accumulate operation, MV is asserted if the result size is wider than 48 bits. MV flag work in both signed and unsigned operations.

In divide operations it is only asserted if a divide-by-zero operation was executed.

43.14.8.3.10.5 MDU Busy Flag (MB)

MB tests as true at the next microinstruction after the MDU start operation, and as false at the last microcycle of any MDU operation execution.

43.14.8.4 Branch Conditions

Microengine allows conditional branch. There are five sets of flags that can be tested in a conditional branch: ALU flags, MDU flags, P flags, Channel flags, and Semaphore flag (flag SMLCK).

When a thread starts to be executed, the values in MDU and ALU flags are not initialized. ALU flags are described in ALU Flags, MDU flags are described in MDU Flags. MDU and ALU flags are updated during execution of microinstructions.

P flags are actually the upper byte of P register, which optionally can work as user defined flags (see P Register).

Channel flags Flags0, Flag1, MRLA, MRLB, TDLA, TDLB, PSS, PRSS and PSTO are obtained from the selected channel (value in CHAN register), while channel flags, LSR, FM[0] and FM[1] are selected by the serviced channel, regardless of the CHAN value[33] .

Flags TDLA/B, MRLA/B, LSR, FM and PSS, are sampled at the beginning of a thread. Flag PSS does not change during its execution while CHAN register is not written. When a write in CHAN register is performed, all flags except LSR and FM are updated according to the channel specified by CHAN value. Flags MRLA/B and TDLA/B are reset when their respective latches in channel are cleared by microcode.

33. Serviced channel does not change during execution of a thread, and it is the channel that requested a service (initial value of CHAN register when a thread starts).

Table 43-367. Channel Flags as Branch Condition

Flag	Description	Service or Selected Channel
Flag0	State Resolution flag	reflects the selected channel (CHAN)
Flag1	State Resolution flag	reflects the selected channel (CHAN)
MRLA	Match A Recognition Latch	these flags reflect the selected channel (CHAN)see MRLA/B - Match Recognition Latches and TDLA/B - Transition Detect Latches for more information.
MRLB	Match B Recognition Latch	
TDLA	Transition A Detection Latch	
TDLB	Transition B Detection Latch	
LSR	Link Service Request	reflects the serviced channel.
PRSS	Input Pin state sampled at service request	reflects the selected channel (CHAN). Does not change if CHAN is not changed.
PSS	Sampled Input Pin State	reflects the selected channel (CHAN). Does not change if CHAN is not changed (see Pin Control Registers).
PSTI	Current Input Pin State.	reflects the selected channel (CHAN). Changes any time.
PSTO	Current Output Pin State	reflects the selected channel (CHAN). Changes any time.
FM	Function Mode Bits	reflects the Function Mode for serviced channel (see the ETPUCnSCR register)

Semaphore condition SMLCK always indicates if a semaphore is locked for the Engine, resolving as false before any lock attempt. For each trial, the SMLCK flag is updated. The SMLCK value set in one thread is not meaningful to the other. After a free, the SMLCK condition tests as false until a new lock attempt on the same thread.

Branch conditions are selected through instruction fields BCC and BCF (see Conditional/ Unconditional Branch).

43.14.9 Microinstruction Set

Each microinstruction can execute up to 3 microoperations in parallel. Microinstructions are grouped into formats, and there are four types of microoperations:

- ALU/MDU Operations
- SDM Operations
- Channel Configuration/Control Operations
- Flow Control Operations

Each microinstruction format is defined by a set of microinstruction fields, which determine the operations, each belonging to one of the groups above (there may be several in one group). Complete microinstruction formats are shown in Microinstruction formats.

Parallelism conflicts may arise when two operations are executed in the same microinstruction. These situations are explained in Microinstruction Parallelism Issues.

43.14.9.1 SDM Microoperations

The access to SDM is made by providing an address and a register to perform a data transfer, except semaphore operations, which are also classified in the SDM group. Only P and DIOB registers can exchange data with SDM. Microengine always addresses SDM in 32-bit boundaries, for 8, 24, or 32 bit wide data.

Direction is determined by the field RW in all addressing modes: RW=0 selects read and RW=1 selects write.

43.14.9.1.1 SDM Addressing Modes

There are 4 addressing modes in eTPU:

- Absolute
- Selected Channel Relative
- Indirect
- Engine Relative

The addressing modes Absolute and Selected Channel Relative use immediate bits to form the physical address of SDM, which is identified in microinstruction as a field called AID. AID field can be 3-, 7-, or 8-bit wide depending on the addressing mode.

43.14.9.1.1.1 Absolute Addressing Mode

In Absolute addressing mode, the address range is 256 parameters, addressed by field AID, which in this mode is 8-bit wide. These parameters are located in SDM addresses from 0 to 255.

```
physical_address = AID[7:0]
```

43.14.9.1.1.2 Selected Channel Relative Addressing Mode

In Selected Channel Relative addressing mode, only the first 8 (with 3-bit AID) or 128 (with 7-bit AID) parameters of the selected channel are accessible, depending on the microinstruction format. Physical address is calculated using the channel parameter base

address that is specified in field CPBA of ETPUCnCR register (see the ETPUCnCR register). AID field is added to channel parameter base address to compose the physical address. The address equation is:

```
physical_address = selected_channel_parameter_base_address + AID[6:0], or
physical_address = selected_channel_parameter_base_address + AID[2:0]
```

43.14.9.1.1.3 Indirect Addressing Mode

In Indirect Addressing mode the physical address is taken from DIOB register. Only DIOB bits 14 to 2 are relevant. Since the SDM word address is shifted two bits up in DIOB, its contents hold the same parameter address value used by Host. The address equation for SDM access is:

```
physical_address = DIOB[13:2], or
physical_address = (truncated to 12 bits) (DIOB / 4)
```

Indirect addressing mode is used to access the Shared Subsystem Area when ETPUCxCR bit SSMAE=1 and DIOB addresses the 16 Kbytes just above the SDM range (DIOB bit 14 = 1). See Shared Subsystems Support for more details. If SSMAE=0, accesses with DIOB bit 14 = 1 hit the same SDM addresses as with bit 14 = 0, providing full backwards compatibility with respect to DIOB contents.

Indirect addressing mode can have post-increment or pre-decrement on DIOB, allowing stack operations. See DIOB Stack Operation for more information.

43.14.9.1.1.4 Engine Relative Addressing Mode

In Engine Relative Addressing mode the physical address is the concatenation of the ETPUECR field ERBA (see the ETPUECR register) with the 7-bit AID instruction field. This allows the same function microcode, when running on distinct engines, to access different address spaces, global to the engine only.

The SDM physical word address in Engine Relative Addressing mode is calculated as follows:

```
physical address = (ETPUECR[ERBA]<< 7) + AID[6:0]
```

43.14.9.1.2 SDM Source/Destination Registers

When performing an SDM operation, only DIOB or P can be used as data source or destination. P is 32-bit wide, and DIOB is 24-bit wide. Microinstruction field P/D (1 bit) is used to choose between P and DIOB as data the source or destination. When the P/D field is not available in microinstructions that support SDM access, the source/destination is P.

Table 43-368. SDM source/destination Register Selection

P/D	meaning
0	P access
1	DIOB access

43.14.9.1.3 SDM Operation Size

When using DIOB register to perform SDM data transfers, the operation size is always 24-bit wide (lower 24 bits of SDM). When using P register, the operation size can be 8-, 24- or 32-bit wide, which is controlled by microcode RSIZ field (2 bits). RSIZ meaning is shown in the following table.

Table 43-369. SDM P access Size

RSIZ	P access	DIOB access
00	full 32-bit access (i.e. P[31:0]=SDM[addr] [31:0])	RESERVED
01	only upper 8 bits are transferred (i.e. P[31:24] = SDM[addr] [31:24])	RESERVED
10	only lower 24 bits are transferred (i.e. P[23:0] = SDM[addr] [23:0])	DIOB = SDM[addr] [23:0
11	RESERVED	RESERVED

RSIZ is not available in all microinstructions that support SDM access. In microinstructions where RSIZ field is not available, SDM access will be 24 bits by default.

When performing a Zero SDM write operation (see Zero SDM Operation), RSIZ defines the size of operation regardless of the P/D field (SDM Source/Destination Registers).

43.14.9.1.4 SDM access direction

RW field defines the direction of the access in the SDM. The access direction is summarized in the following table.

Table 43-370. SDM access direction

R/W	Meaning
0	read SDM parameter into P or DIOB registers
1	write SDM parameter from P or DIOB registers

43.14.9.1.5 Zero SDM Operation

Zero SDM operation is controlled by microcode field ZRO (1bit). When ZRO field is 0, the data portion written in SDM or in P/DIOB (SDM read) registers will always be 0x0. When performing a Zero SDM write operation, the RSIZ is relevant regardless of the P/D field (usually RSIZ is meaningful only for P/D = 0), which means that zero SDM write operation can be performed with 32, 24 or 8 bits according to SDM operation size. These conditions are summarized in the following table.

Table 43-371. Zero SDM Operation

ZRO	RW	P/D	Meaning
0	0	0	Clear P register. Size is determined by RSIZ field. See SDM Operation Size
0	0	1	Clear DIOB (all 24 bits), independently of RSIZ
0	1	x	Clear SDM parameter. Size is determined by RSIZ field. See SDM Operation Size
1	RW	P/D	Regular SDM operation

Note

When field STC is present, STC=11 also disables Zero SDM operation (see Table 43-372). The conflicts with DIOB operations (see DIOB Stack Operation) and ALU operations are resolved like a normal SDM operation (see Table 43-413).

Zero SDM read operations do check memory errors on the read data, however discarded. Zero SDM write operations normally update the ECC of the written data.

43.14.9.1.6 DIOB Stack Operation

SDM Indirect Addressing Mode (see Indirect Addressing Mode) is used if STC field (2 bits) exists in the microinstruction, controlling automatic increment/decrement of DIOB register, as shown in the following table, thus allowing stack operations. DIOB is incremented and decremented in word addresses, only from bits 15 downto 2, i.e.: the bits 23 to 16 and 1 to 0 are left untouched by STC pre-decrement and post-increment.

Table 43-372. DIOB Post-Increment / Pre-Decrement - STC

STC	Meaning
00	Post-Increment of DIOB
01	Pre-Decrement of DIOB
10	No Increment/Decrement (normal access)
11	No SDM Access[1]

1. also disables Zero SDM operation.

43.14.9.1.7 Semaphore Operations

Semaphore lock and free operations are available through eTPU microcode. For more information about semaphores see Hardware Semaphores. There are 2 microinstruction fields that control semaphore operations: FL (1 bit) and SMPR (2 bits). Serviced channel sees 4 semaphores, selected by field SMPR.

Table 43-373. Semaphore Operations Fields

Field	Meaning
FL	0 = free semaphore, 1 = lock semaphore
SMPR	semaphore number selector

When freeing a semaphore, the field SMPR has no meaning. This is because only one semaphore can be locked at a time by each engine, so when freeing a semaphore it is not necessary to specify its number.

Note

If microcode tries to lock a semaphore already locked for the same Engine, the semaphore continues locked for the Engine and the SMLCK branch condition resolves as true.

43.14.9.2 ALU/MDU Operations

ALU/MDU microoperations mostly comprises two sources, one destination and one operation. The operation is generally selected through fields ALUOP, ALUOPI or SHF. In formats where there is no operation selection field (ALUOP, ALUOPI or SHF), the operation performed is always addition; however, it is possible to perform subtraction, increment or decrement using fields BINV (see B-Source Inversion) and CIN (see Carry-in Control).

43.14.9.2.1 Source and Destination Register Set Selection

Microcode field T4ABS allows selection of a source from either one of two register sets, shown in Table 43-377. The same applies to T2ABD, used for ALU destination selection with other two register sets, as shown in Table 43-378. Microinstruction fields ABSE and ABDE control the register set selection for source and destination, respectively, when available at the format. In formats without ABSE/ABDE, the field T4BBS determines the register sets used by T2ABD and T4ABS, as shown in Microinstructions Without Fields ABSE and ABDE.

43.14.9.2.1.1 Microinstructions With Fields ABSE and ABDE

In microinstructions where ABSE and ABDE fields are available (1 bit each), ABSE controls register set selection for T4ABS and ABDE controls register set selection for T2ABD. The following table shows the meaning of values for ABSE and ABDE fields.

Table 43-374. Register Set Selection by ABSE or ABDE

ABSE or ABDE	Register Set Selected
0	second
1	first

43.14.9.2.1.2 Microinstructions Without Fields ABSE and ABDE

When ABSE and ABDE are not available in a microinstruction format, register set is selected by T4BBS field. The following table explains how to select a register set either for T4ABS and T2ABD.

Table 43-375. Register Set Selection by T4BBS w/o ABSE,ABDE

T4BBS	Register Set For T2ABD	Register Set For T4ABS
0xx	first	first
100	second	second
101	second	first
110	first	second
111	first	first
none[1]	first	first

1. refers to operations with immediate data as B-source, without ABSE,ABDE.

43.14.9.2.2 Selecting Sources and Destination

All ALU/MDU operations need 2 sources (called AS and BS) and 1 destination (called AD), except for some of those that use immediate data (see Operations With Immediate Data). Fields T4ABS (4 bits), ABSE (1 bit), T4BBS (3 bits) select sources, while T2ABD (4 bits) and ABDE (1 bit) select the destination. When MDU is used (multiply/divide), T2ABD destination selection is ignored and results are stored in MACH and MACL (see MAC and Divide Unit (MDU)). ABSE and ABDE are not available in some microinstruction formats that support ALU/MDU operations. However, in all formats where ABSE is available, ABDE is also available and vice-versa. The existence of

ABSE/ABDE fields changes the meaning of T4BBS field, as shown in the table below. On instructions with immediate data, it is used as B-source (see Operations With Immediate Data).

All sources and destinations have a size associated to it, and these sizes are used to select flag sample position (see Flags Sampling Control). Sizes can be 8, 16 or 24 bits. Registers that are not exactly of one of these sizes are treated as the immediately upper size (e.g., CHAN[4:0] is an 8-bit source). See Flags Sampling Control for more information.

Some parallelism issues arise when selecting P, DIOB, ERTA or ERTB as destination registers, since they can be modified by other microoperations in the same microinstruction (see Microinstruction Parallelism Issues for details).

Table 43-376. B Source Selection - T4BBS

T4BBS	Meaning in microinstruction formats with ABSE/ABDE	Meaning in microinstruction formats without ABSE/ABDE[1]
000	BS[23:0] = P[23:0]	
001	BS[23:0] = A[23:0]	
010	BS[23:0] = SR[23:0]	
011	BS[23:0] = DIOB[23:0]	
100	reserved	BS = 0
101	reserved	BS = 0
110	reserved	BS = 0
111	BS=0, or Max const., if CIN=0 and BINV=0 (see Generating "max" constant).	

1. T4BBS also selects A-source and destination register set in this case, according to Table 43-375.

T4ABS selects one source from 2 register sets, shown in the following table. ABSE and T4BBS control which set T4ABS field uses to select the source. For more information about how to select a register set for T4ABS and T2ABD see Source and Destination Register Set Selection . All sources are zero-filled to 24 bits, unless sign-extension is specified (see A-Source Size Override).

Table 43-377. A Source Selection - T4ABS

T4ABS	First Register Set		Second Register Set	
	Selected Register	Size	Selected Register	Size
0000	AS[7:0]=P[7:0]	8	AS[7:0]=0	8
0001	AS[7:0]=P[15:8]	8	AS[23:0]=C[23:0]	24
0010	AS[7:0]=P[31:24]	8	AS[15:0] = TPR[15:0]	16
0011	AS[23:0] = ERTB[23:0]	24	AS[23:0] = B[23:0]	24
0100	AS[23:0] = D[23:0]	24	AS[23:0] = TRR[23:0]	24

Table continues on the next page...

Table 43-377. A Source Selection - T4ABS (continued)

T4ABS	First Register Set		Second Register Set	
	Selected Register	Size	Selected Register	Size
0101	AS[15:0] = P[15:0]	16	AS[7:0] = 0, read_match[1]	8
0110	AS[15:0] = P[31:16]	16	AS[13:0] = RAR[13:0]	16
0111	AS[7:0] = P[23:16]	8	AS[23:0] = MACH[23:0]	24
1000	AS[23:0] = P[23:0]	24	AS[23:0] = MACL[23:0]	24
1001	AS[23:0] = A[23:0]	24	AS[4:0]=CHAN[4:0]	8
1010	AS[23:0] = SR[23:0]	24	AS[14 :2] = CHAN_BASE[2]	16
1011	AS[23:0] = DIOB[23:0]	24	AS[13:0] = ENGINE_BASE[3]	16
1100	AS[23:0] = TCR1[23:0]	24	Reserved	-
1101	AS[23:0] = TCR2[23:0]	24	Reserved	-
1110	AS[23:0] = ERTA[23:0]	24	Reserved	-
1111	AS[23:0] = 0	24	Reserved	-

1. T4ABS=0101 with second register set also reads MatchA/B registers into ERTA/B (see Special T4ABS Source Operation: Read Match Registers).
2. CHAN_BASE is the selected channel's base SDM address in channel relative address mode (see CHAN_BASE as a Source).
3. ENGINE_BASE is the ETPUECR field ERBA shifted left 9 positions.

T2ABD selects the destination from 1 of 2 register sets, shown in the following table. ABDE and T4BBS control which set T2ABD field uses to select the destination.

Table 43-378. Destination Selection - T2ABD

T2ABD	First Register Set		Second Register Set	
	Selected Register	Size	Selected Register	Size
0000	A[23:0] = AD[23:0]	24	C[23:0] = AD[23:0]	24
0001	SR[23:0] = AD[23:0]	24	LINK[7:0] = AD[7:0]	8
0010	ERTA[23:0] = AD[23:0][1]	24	TPR[15:0] = AD[15:0]	16
0011	ERTB[23:0] = AD[23:0][2]	24	B[23:0] = AD[23:0]	24
0100	DIOB[23:0] = AD[23:0]	24	CHAN[4:0] = AD[4:0]	8
0101	P[15:0] = AD[15:0]	16	D[23:0] = AD[23:0]	24
0110	P[31:16] = AD[15:0]	16	RAR[12:0] = AD[12:0]	16
0111	P[23:0] = AD[23:0]	24	MACH[23:0] = AD[23:0]	24
1000	TCR1[23:0] = AD[23:0]	24	MACL[23:0] = AD[23:0]	24
1001	TCR2[23:0] = AD[23:0]	24	Reserved	-
1010	P[31:24] = AD[7:0]	8	Reserved	-
1011	P[23:16] = AD[7:0]	8	Reserved	-
1100	P[15:8] = AD[7:0]	8	Reserved	-
1101	P[7:0] = AD[7:0]	8	Reserved	-
1110	TRR[23:0] = AD[23:0]	24	Reserved	-
1111	no destination selected[3]	24	Reserved	-

1. T2ABD=0010 with first register set also writes to MatchA or UDCM registers of the selected channel if field ERWA=0 (see Write Channel Match and UDCM Registers).
2. T2ABD=0011 with first register set also writes to MatchB register of the selected channel if field ERWB=0.
3. if no destination is selected, ALU flags are updated, although the result is lost.

43.14.9.2.2.1 Max Const Generation With T4BBS=111

When T4BBS = 111, BINV = 0, and CIN = 0, the value assigned to BS will be 0x800000, and not 0x0 as expected. See Generating "max" constant for detailed explanation about this.

43.14.9.2.2.2 Special T4ABS Source Operation: Read Match Registers

When T4ABS = 0101 and the source for T4ABS is selected from the second register set, the constant 0x00 is used as AS (8-bit size) and the following register transfer is performed in parallel as well: match registers of the selected channel (value in CHAN register) are copied to ERTA/ERTB registers, where ERTA receives the value of MatchA register and ERTB receives the value of MatchB register (see ER - Event Registers). Note that ALU destination can still be chosen by T2ABD in parallel. When ERTA or ERTB is selected by T2ABD, a parallelism issue arises (see ALU Operations and Read Match Registers).

43.14.9.2.2.3 CHAN_BASE as a Source

Each channel has a parameter base address in SDM, which is configured in ETPUCxCR registers, CPBA field (see the ETPUCnCR register). CHAN_BASE, which represents a parameter address (CPBA*2), can be used as A-source using T4ABS=1010 when T4ABS selects a source from the second register set. In this case, CHAN_BASE is loaded into AS[13:2] to form the byte address (AS[23:14]=0, AS[1:0]=0). For example, in Indirect addressing mode, where the destination register is DIOB, CHAN_BASE is loaded into DIOB[13:2], which is the parameter address, and DIOB[13:0] represents the byte address. CHAN_BASE is the base address of the selected channel (given by CHAN register).

43.14.9.2.3 Flags Sampling Control

This section explains how the flags Z (zero), C (carry), N (negative) and V (overflow) are updated in an ALU operation. When there are post-ALU shift operations, the ALU Carry Out is not directly sampled in Carry flag, but passed to the post-ALU shifter (see Shift Operations). Since the size of source operands in ALU operations is variable, flags can be sampled as an operation of 8, 16 or 24 bits wide. The operation size selection is automatic, based on defined sizes of sources and destination, using the equation:

```
operation_size = minor(size_of(destination), greater(size_of(A-Source),
size_of(B-Source)))
```

Operation size can also be shown with the following table:

Table 43-379. Operation Size Determination

A Source	B Source	Destination	Operation Size
x	x	8 bits	8 bits
8 bits	8 bits	x	8 bits
16 or 24 bits	x	16 bits	16 bits
16 bits	8 or 16 bits	24 bits	16 bits
8 or 16 bits	16 bits	24 bits	16 bits
24 bits	x	24 bits	24 bits
x	24 bits	24 bits	24 bits

Note

Whenever BS = (constant) 0, its size is considered 8 bits, and all 24 bits in B-bus are set to 0. Therefore, all operations with BS = (constant) 0 have their size determined by AS and Destination only.

CCS field (1 bit) controls whether flags will be updated or not (Table 43-381). When CCS bit exists in a microinstruction, the operation size will be used to sample flags. In some microinstructions CCS field is replaced by CCSV (2 bits, Table 43-380). Flag sampling according to CCSV can be set as defined by the operation size, or fixed as 8 or 16 bit operations.

Note

For the post-ALU rotate right operation, CCSV also determines the rotate size: whether 8-bit, 16-bit, or determined by the operation size.

Table 43-380. Flag Sampling Using CCSV field

CCSV	Meaning
00	sample flags as an 8 bit operation
01	sample flags as a 16 bit operation
10	sample flags as defined by operation size
11	do not sample flags

Table 43-381. Flag Sampling Using CCS field

CCS	Meaning
0	sample flags as defined by operation size
1	do not sample flags

When neither CCS nor CCSV are present in the microinstruction, flags are not sampled. CCS and CCSV do not affect the Carry update on Exchange Bit operation (see Exchange Bit), but does control the N and Z flags.

43.14.9.2.4 B-Source Inversion

The data selected as second source (T4BBS) can be inverted (bitwise boolean NOT) before operation. This is controlled by microinstruction field BINV (1 bit, in the following table). A zero value for BINV activates B-source inversion.

Table 43-382. B-Source Inversion - BINV

BINV	Meaning
0	invert B-source[1]
1	keep B-source bus unchanged

1. Except on max-constant selection, see Generating "max" constant .

BINV also selects between *adc* or *sbc* enhanced ALU operation, using inverted C flag as carry-in besides BS inversion for *sbc* . Note that BINV does not invert carry in fixed-carry operations (see Table 43-383).

When BINV = 0, T4BBS = 111 and CIN = 0, the value assigned to BS is 0x800000, instead of 0x0. See Generating "max" constant for more details.

43.14.9.2.5 Carry-in Control

CIN field (1 bit, Table 43-383) controls the carry-in for addition/subtraction operations. Functionality of CIN field depends on the arithmetic operation selected by ALUOP. When ALUOP is not available in microinstruction, the operation selected is *add* . For carry-in control in MDU operations, see the following table.

Table 43-383. ALU Carry-In Control

operation	CIN=0	CIN=1
add (addition)	carry-in used is 1[1]	carry-in used is 0
adc (addition with carry)[2]	carry-in used is C flag	
sbc (subtraction with borrow)[3]	carry-in used is inverted C flag	

1. Except on max-constant selection, see Generating "max" constant .

2. Selected by ALUOP=11000 and BINV=1
3. Selected by ALUOP=11000 and BINV=0

43.14.9.2.5.1 Generating "max" constant

When T4BBS = 111, CIN = 0 and BINV = 0, BS is assigned to 0x800000 (called "max constant") instead of 0x000000. This is an exception for CIN and BINV fields: when "max constant" is selected, the carry in is 0 and B-source ("max constant" itself) is not inverted, neither the carry out.

"Max constant" is the value which, added to a time base value minus 1, gives the farthest wrapped time base value that satisfies a channel greater-equal comparison. See Enhanced Channels for more info.

43.14.9.2.6 Shift Operations

There are three types of shift operations: ALU, post-ALU and Shift Register. ALU shift operations are covered in ALU/MDU Operation Selection . Post-ALU and Shift Register are covered in the following sections.

43.14.9.2.6.1 Shift Register Operations

SR can be used as a general purpose register and it can easily shift-right its contents, combined or not to post-ALU shift operations. If field SRC (1 bit) in microcode is 0, SR will shift its contents 1 bit to the right according to the algorithmic description below. SR shifting operation depends also on SHF or ALUOP fields. ALUOP and SHF never exist both on the same microinstruction format.

Table 43-384. Shift Register Control - SRC

SRC	Meaning
0	shift right SR 1 bit
1	no shift

SR Operation:

```
SR[22:0] = SR[23:1];
if SHF == "01" or ALUOP == "10110" then
 SR[23] = ALU_OUT[0];
else
  SR[23] = 0;
endif;
```

43.14.9.2.6.2 Post-ALU Shift Operations

Post-ALU shift can be selected by SHF field (2 bits) or by some specific ALUOP field values. SHF and ALUOP fields are never both available in the same microinstruction format. When selecting post-ALU shift operation using ALUOP field, ALU will always add the sources before shifting the result.

Table 43-385. Post-ALU Shift Operation

Post ALU Operation	SHF[1]	ALUOP
shift left (1 bit)	00	10101
shift right (1 bit)	01	10110
rotate right (1 bit)	10	10111
no shift/rotate	11	any other[2]

1. ALU performs AS+BS before shift/rotate for all SHF values.
2. some ALUOP combinations perform shift/rotate, but not using the Post-ALU Shifter (see Table 43-390)

Carry flag is only updated when CCS or CCSV fields allow it (see Flags Sampling Control). Algorithmic descriptions of post-ALU shift operations are presented in ALU ADD Operation with and without shifting .

43.14.9.2.7 Conditional ALU/MDU Operation Execution

The 3-bit field AS/CE allows conditional execution of arithmetic operation, as shown in the following table . The same field can also be used for overriding the size of A-Source (see A-Source Size Override).

Table 43-386. ALU/MDU Conditional Execution

AS/CE	Meaning
000	used for A-Source size override (see A-Source Size Override)
001	
010	execute if C = 1
011	execute if C = 0
100	execute if Z = 1
101	execute if Z = 0
110	execute if N = 1
111	execute unconditionally/no size override

Other operations not related to ALU/MDU in the same microinstruction are not affected by the AS/CE field.

If a conditional operation is selected, there is no A-Source size override; similarly, when size override for A-Source is selected, the ALU/MDU operation executes unconditionally.

When a conditional ALU/MDU operation is not executed:

- the destination register is not updated. If the destination is CHAN, no actions associated with CHAN assignment occur (see Channel Selection Register - CHAN).
- the ALU and MDU flags are not updated.
- MDU does not start any operation, i.e., MACH and MACL are not updated.
- SR does not shift.
- T4ABS-selected read-match does not occur.

43.14.9.2.8 A-Source Size Override

Some values if the AS/CE field are used for A-Source Size Override, as shown in the following table.

Table 43-387. A-Source Size Override

AS/CE	Meaning
000	A-source size override to 8 bits
001	A-source size override to 16 bits
010	used for conditional execution (see Conditional ALU/MDU Operation Execution)
011	
100	
101	
110	
111	execute unconditionally/no size override

Register size override zero-pads an overridden source to 24-bits (if no sign extension is performed, see A-source Sign Extension) and affects operation size calculation. When register source is wider than size override, most significant bits of selected register are not used as A source (zeros are used instead). When size override is wider than selected register, register value is padded to zeros.

When size override is used with MDU operations, it affects only the operand values, but not the operation size: MDU operation size is fully determined by the operation definition (fields ALUOP, ALUOPI).

Table 43-388. AS/CE field A source size override funcionality

Size Override	Size of Selected Register	AS Value[1]
8 bits	8 bits	reg[7:0]

Table continues on the next page...

Table 43-388. AS/CE field A source size override funcionality (continued)

Size Override	Size of Selected Register	AS Value[1]
8-bits	16 bits	reg[7:0]
8bits	24 bits	reg[7:0]
16 bits	8 bits	reg[15:0]
16 bits	16 bits	reg[15:0]
16 bits	24 bits	reg[15:0]

1. All values are zero-padded to 24 bits

43.14.9.2.9 A-source Sign Extension

The SEXT microinstruction field forces sign extension of A source according to the size of A operand, either overridden or not by AS/CE field, according to the following table. The sign is taken from the size-overridden value, not the original one.

Table 43-389. A source Sign Extension

SEXT	Meaning
0	extends sign of A source from its size to 24-bits
1	does not extend sign of A source

A-source sign is not extended in microinstructions without SEXT field, even if AS/CE field is present.

43.14.9.2.10 ALU/MDU Operation Selection

When field ALUOP is available in microinstruction, enhanced ALU operations shown in the following table can be performed, otherwise addition is performed. The ALU operations are defined in ALU and Post-ALU Shifter. The MDU operations are defined in MAC and Divide Unit (MDU).

Table 43-390. ALU Operation Selection - ALUOP

ALUOP	Operation	Comment
00000	AS mults BS[7:0]	signed multiplication
00001	AS multu BS[7:0]	unsigned multiplication
00010	AS fmults BS[7:0]	signed fractional multiplication
00011	AS fmultu BS[7:0]	unsigned fractional multiplication
00100	AS mults BS[15:0]	signed multiplication
00101	AS multu BS[15:0]	unsigned multiplication
00110	AS fmults BS[15:0]	signed fractional multiplication

Table continues on the next page...

Table 43-390. ALU Operation Selection - ALUOP (continued)

ALUOP	Operation	Comment
00111	AS fmultu BS[15:0]	unsigned fractional multiplication
01000	AS mults BS[23:0]	signed multiplication
01001	AS multu BS[23:0]	unsigned multiplication
01010	AS macs BS[23:0]	signed multiply-accumulate
01011	AS macu BS[23:0]	unsigned multiply-accumulate
01100	AS div BS[7:0]	unsigned division by 8-bit value
01101	AS div BS[15:0]	unsigned division by 16-bit value
01110	AS div BS [23:0]	unsigned division by 24-bit value
01111	n.a.	RESERVED
10000	AS[23:0] | BS[23:0]	24 bit bitwise OR
10001	AS[23:0] ^ BS[23:0]	24 bit bitwise XOR
10010	AS[23:0] & BS[23:0]	24 bit bitwise AND
10011	abs(AS)	absolute value of AS
10100	AS + BS	arithmetic addition
10101	(AS + BS) shl 1	arithmetic addition with 1-bit post-ALU shift left. (Post-ALU Shift Operations)
10110	(AS + BS) shr 1	arithmetic addition with 1-bit post-ALU shift right (Post-ALU Shift Operations)
10111	(AS + BS) ror 1	arithmetic addition with 1-bit post-ALU rotate right (Post-ALU Shift Operations)
11000	AS adc/sbc BS[1]	addition/subtraction with C flag (Carry-in Control)
11001	AS shl (2^(BS[1:0]+1))	AS is shifted left: 2 bits for BS=0; 4 for BS=1; 8 for BS=2; 16 for BS=3
11010	AS shr (2^(BS[1:0]+1))	AS is shifted right: 2 bits for BS=0; 4 for BS=1; 8 for BS=2; 16 for BS=3
11011	AS ror (2^(BS[1:0]+1))	AS is rotated right: 2 bits for BS=0; 4 for BS=1; 8 for BS=2; 16 for BS=3
11100	AS exch BS[4:0]	exchange C flag and AS bit determined by BS (Exchange Bit)
11101	AS setb BS[4:0]	set bit in AS determined by BS [2, 2]
11110	AS clrb BS[4:0]	clear bit in AS determined by BS[3]
11111	n.a.	RESERVED

1. Addition/Subtraction is selected by field BINV (see B-Source Inversion)
2. In setb and clrb operations, the register that drives A source is not changed, unless selected as destination of the operation.
3. In setb and clrb operations, the register that drives A source is not changed, unless selected as destination of the operation.

43.14.9.2.11 Operations With Immediate Data

Immediate data can be used with some specific microinstruction formats. eTPU microcode allows 8-, 16- or 24-bit immediate data. Immediate data is loaded as B-source, so T4BBS field is not available. When using 24-bit immediate, an add is performed with A-source = 0 and flags are not updated. ALU operations are available with 8-bit immediate, although the field that selects ALU operation in this case is ALUOPI.

43.14.9.2.11.1 24-bit Immediate Destination

When using 24-bit immediate data, the destination register is selected by T2D field (2 bits), according to the following table.

Table 43-391. 24-bit Immediate Destination - T2D

T2D	Target Register
00	P[23:0]
01	A
10	SR
11	DIOB

43.14.9.2.11.2 Enhanced ALU Operations With Immediate Data

Enhanced operations with immediate data, selected by ALUOPI (5 bits) are allowed only with an 8 bit immediate operand (see the following table).

Table 43-392. ALU Operation Selection With Immediate Data - ALUOPI

ALUOPI	Operation	Comment
00000	AS mults #imm8	signed multiplication
00001	AS multu #imm8	unsigned multiplication
00010	AS fmults #imm8	signed fractional multiplication
00011	AS fmultu #imm8	unsigned fractional multiplication
00100	AS div #imm8	unsigned division
00101	n.a.	reserved
00110	n.a.	reserved
00111	n.a.	reserved
01000	AD[7:0] = AS[7:0] | #imm8, AD[23:8] = AS[23:8]	bitwise OR
01001	AD[7:0] = AS[7:0] ^ #imm8, AD[23:8] = AS[23:8]	bitwise XOR
01010	AD[7:0] = AS[7:0] & #imm8, AD[23:8] = AS[23:8]	bitwise AND
01011	AD[7:0] = AS[7:0] & #imm8, AD[23:8] = 0×0	bitwise AND with clear
01100	AD[15:8] = AS[15:8] | #imm8, AD[23:16] = AS[23:16], AD[7:0] = AS[7:0]	bitwise OR
01101	AD[15:8] = AS[15:8] ^ #imm8, AD[23:16] = AS[23:16], AD[7:0] = AS[7:0]	bitwise XOR

Table continues on the next page...

Table 43-392. ALU Operation Selection With Immediate Data - ALUOPI (continued)

ALUOPI	Operation	Comment
01110	AD[15:8] = AS[15:8] & #imm8, AD[23:16] = AS[23:16], AD[7:0] = AS[7:0]	bitwise AND
01111	AD[15:8] = AS[15:8] & #imm8, AD[23:16] = 0×0, AD[7:0] = 0×0	bitwise AND with clear
10000	AD[23:16] = AS[23:16] | #imm8, AD[15:0] = AS[15:0]	bitwise OR
10001	AD[23:16] = AS[23:16] ^ #imm8, AD[15:0] = AS[15:0]	bitwise XOR
10010	AD[23:16] = AS[23:16] & #imm8, AD[15:0] = AS[15:0]	bitwise AND
10011	AD[23:16] = AS[23:16] & #imm8, AD[15:0] = 0x0	bitwise AND with clear
10100	AS + #imm8	arithmetic addition
10101	(AS + #imm8) shl 1	arithmetic addition with 1-bit shift left.
10110	(AS + #imm8) shr 1	arithmetic addition with 1-bit shift right
10111	(AS + #imm8) ror 1	arithmetic addition with 1-bit rotate right
11000	n.a.	reserved
11001	AS shl (2^(#imm8[1:0]+1))	AS is shifted left: 2 bits for #imm8=0; 4 for #imm8=1; 8 for #imm8=2; 16 for #imm8=3
11010	AS shr (2^(#imm8[1:0]+1))	AS is shifted right: 2 bits for #imm8=0; 4 for #imm8=1; 8 for #imm8=2; 16 for #imm8=3
11011	AS ror (2^(#imm8[1:0]+1))	AS is rotated right: 2 bits for #imm8=0; 4 for #imm8=1; 8 for #imm8=2; 16 for #imm8=3
11100	AS exch #imm8[4:0]	exchange C flag and AS bit determined by #imm8[4:0] (see Exchange Bit)
11101	n.a.	reserved
11110	n.a.	reserved
11111	n.a.	reserved

43.14.9.3 Channel Control and Configuration Microoperations

Channel Control and Configuration fields set configuration values in the channel logic of the channel selected by the CHAN register, except fields LSR and CIRC.

43.14.9.3.1 Channel Flags Operations

Each channel has two associated hardware flags, called Channel Flag 0 and Channel Flag 1. Microcode field FLC (3 bits) allows them to be set or cleared, as shown in the following table . These flags can be tested by microcode as a branch condition, and may also influence in the Entry Point taken, allowing fast state decoding. For more details, see Entry Points.

Table 43-393. P Flags Operation - FLC

FLC	Meaning
000	clear flag0
001	set flag0
010	clear flag1
011	set flag1
100	copy flag1:flag0 from P[25:24]
101	copy flag1:flag0 from P[27:26]
110	copy flag1:flag0 from P[29:28]
111	no operation

43.14.9.3.2 Comparator and Time Base Selection

TBSA and TBSB fields (4-bit wide each) are used to configure the type of the comparator and the time bases used for match or capture (see the following tables). TBSA can also be used to control the Output Buffer Enable signal (See Pin Control Registers).

Table 43-394. Time Base Selection A - TBSA

TBSA[3] = 0	**TBSA bit**	**2**	**1**	**0**
	bitfield	**Comparator selection**	**Capture selection**	**Match TB selection**
	0	greater or equal	TCR1	TCR1
	1	equal-only	TCR2	TCR2
TBSA[3] = 1	**action**	**2**	**1**	**0**
	set OBE=1	0	0	0
	set OBE=0	0	0	1
	do nothing	1	1	1
	reserved	all other values		

Table 43-395. Time Base Selection B - TBSB

TBSB[3] = 0	**TBSB bit**	**2**	**1**	**0**
	bitfield	**Comparator selection**	**Capture selection**	**Match TB selection**
	0	greater or equal	TCR1	TCR1
	1	equal-only	TCR2	TCR2
TBSB[3] = 1	action	**2**	**1**	**0**
	do nothing	1	1	1
	reserved	all other values		

43.14.9.3.3 Transition Detection and Pin Action Control

IPACA/B and OPACA/B fields are used to configure transition detection sensitivity (for the channel input signal) or output pin action control (for the channel output signal), as defined in the table below. IPACA and IPACB have the same format, where IPACA is related to Match A and first transition detection, and IPACB to Match B and second transition detection. The same applies in analogue way to OPACA and OPACB.

For the output signal, configuring OPAC registers does not change the current signal state, but defines the action to be done when a match or transition detection occurs. See Match Recognition and Channel Modes on Output Signal Generation for more information.

IPACA/B=1xx also enables assertion of MRLA/B during Time Slot Transition. See Match Recognition.

Table 43-396. Input and Output Pin Action Control - IPACA/B and OPACA/B

value	IPAC meaning	OPAC meaning
000	Do not detect transitions	Do not change output signal
001	Detect rising edge only	Match[1] sets output signal high
010	Detect falling edge only	Match[1] sets output signal low
011	Detect both edges	Match[1] toggles output signal
100	Detect input signal = 0 on Match[1, 1]	Transition detection sets output signal low
101	Detect input signal = 1 on Match[1]	Transition detection sets output signal high
110	reserved	Transition detection toggles output signal
111	do not change IPAC	do not change OPAC

1. Match A is used for IPACA/OPACA, and Match B for IPACB/OPACB.

43.14.9.3.4 Immediate Pin State Control

It is possible to change output signal state immediately by using PSC (2 bits) and PSCS (1 bit) fields.

Table 43-397. Immediate Pin State Control - PSC and PSCS

PSC	PSCS	Meaning
00	0	set signal as specified by OPACA (see Transition Detection and Pin Action Control)
00	1	set signal as specified by OPACB (see Transition Detection and Pin Action Control)
01	x	set signal high
10	x	set signal low
11	x	don't change signal state

43.14.9.3.5 Write Channel Match and UDCM Registers

Match registers can have their values changed using ERWA and ERWB fields (1 bit each). They also set their respective MRLE register (see Match Recognition).

ERWA can also be used to program the UDCM register (see UDCM - User Defined Channel Mode). The field CMW selects where the contents of ERTA is copied when ERWA is active (see the following table).

Table 43-398. Write MatchA/B - ERWA/B

Field	CMW	Value	Action
ERWA	1	0	write ERTA value in MatchA. Enable matches for MatchA register (MRLEA=1)
	0	0	write ERTA value into UDCM
	1	1	don't change UDCM, MatchA and MRLEA
	0	1	reserved
ERWB	1	0	write ERTB value in MatchB. Enable matches for MatchB register (set MRLEB=1)
	1	1	don't change MatchB and MRLEB
	0		
	0	0	reserved

If ERTA or ERTB is a destination of an ALU operation and, at the same time, the respective ERWA/B field is active, the new ERTA value is the one written into the MatchA/B register or the UDCM register.

43.14.9.3.6 Clear Transition/Match Event Registers

Flags MRLA, MRLB, TDLA and TDLB (see ER - Event Registers) indicate the state of matches and transitions detected in the selected channel, and it is possible to clear those flags using the microcode fields MRLA, MRLB (1 bit each) and TDL (1 or 2 bits, depending on the format). The flags cleared by these microcode fields are the actual channel flags, and also the ones sampled into the branch logic.

TDL can be one or two bits wide, depending on the microinstruction format (see Microinstruction formats). Two-bit TDL allows independent clearing of TDLA and/or TDLB . Table 43-400 defines the two-bit TDL field.

Table 43-399. Clear Transition/Match Event Registers - MRLA/B, TDL

Field	Meaning
MRLA	0 = clear MRLA event register, 1 = don't change
MRLB	0 = clear MRLB event register, 1 = don't change
TDL (1 bit)	0 = clear TDLA and TDLB flags, 1 = don't change

Table 43-400. Independent TDLA/B clear - two-bit TDL

value	Meaning
0 0	clear TDLA
0 1	clear TDLB
1 0	clear both TDLA and TDLB
1 1	do not clear TDLA or TDLB

43.14.9.3.7 Disable Matches

Microcode field MRLE (1 bit) allows disabling matches on channel selected by CHAN register, for both MatchA and MatchB registers, by clearing their respective MRLE bits. Matches can be enabled for each Match register using ERWA and ERWB fields (see Write Channel Match and UDCM Registers).

Table 43-401. Disable Matches - MRLE

MRLE	Meaning
0	Disable matches for Match A and Match B
1	don't change match enabling

Some instruction formats have a two-bit MRLE field (see Microinstruction formats) which allows independent disabling of Matches 1 and 2, as shown in the following table.

Table 43-402. Two-bit MRLE

MRLE	Meaning
0 0	Disable Match A (clear MRLEA)
0 1	Disable both Matches (clear MRLEA and MRLEB)
1 0	Disable Match B (clear MRLEB)
1 1	nop

43.14.9.3.8 Disable Match and Transition Service Requests

Microcode field MTD (2 bits) disables match and transition service requests for the selected channel. MTD does not disable Link Service Request and Host Service Request. MTD sets or resets register SRI (for more details see SRI - Match/Transition Service Request Inhibit Latch) and TCCEA (see TCCEA - Transition Continuous Capture Enable) .

Table 43-403. Disable Match and Transition Service Request - MTD

MTD	Action on SRI	Action on TCCEA
00	SRI = 0: enable service requests for match and transition	TCCEA = 0: disable transition captures[1] when TDLA = 1
01	SRI = 1: disable service requests for match and transition	
10	SRI = 1: disable service requests for match and transition	TCCEA = 1: enable transition captures[2] when TDLA = 1
11	don't change	

1. Disables only captures on transition events specified by IPACA.
2. Enables only captures into CaptureA register, on transition events specified by IPACA.

43.14.9.3.9 Predefined Channel Modes

Microcode PDCM field (4 bits) defines the channel mode (see Channel Modes).

PDCM coding is shown in the table below. Note that PDCM bit 0 selects between Single Transition (PDCM[0]=0) and Double Transition (PDCM[0]=1) predefined modes.

PDCM is also used to select the user-defined channel mode, defined by the channel register UDCM (see UDCM - User Defined Channel Mode).

Table 43-404. Predefined Channel Modes

PDCM	Channel mode
0000	em_b_st
0001	em_b_dt
0010	em_nb_st
0011	em_nb_dt

Table continues on the next page...

Table 43-404. Predefined Channel Modes (continued)

PDCM	Channel mode
0100	m2_st
0101	m2_dt
0110	bm_st
0111	bm_dt
1000	m2_o_st
1001	m2_o_dt
1010	**user-defined channel mode**
1011	reserved
1100	sm_st
1101	sm_dt
1110	sm_st_e
1111	keep current channel mode

43.14.9.3.10 Channel Interrupt and Data Transfer Requests

Microcode can issue Interrupt Requests, Data Transfer Requests and Global Exception through CIRC field. For more information see Interrupts and Data Transfer Requests.

Table 43-405. Channel and Data Transfer Requests - CIRC

CIRC	Meaning
000	Channel Interrupt Request from selected channel
001	Data Transfer Request from selected channel
010	Channel Interrupt and Data Transfer requests from selected channel
011	Channel Interrupt and Data Transfer requests from serviced channel
100	Channel Interrupt Request from service channel
101	Data Transfer Request from service channel
110	Global Exception
111	don't request interrupt

43.14.9.3.11 Clear Link Service Request

Microcode LSR field (1 bit) is used to clear the Link Service Request flag of the serviced channel (may not be the one selected by CHAN). The LSR branch condition is always cleared, but not the Link Service Request, if another channel link was received by the serviced channel during the executing thread. See Channel Link for more information.

Table 43-406. Link Service Request Negation Control - LSR

LSR	Meaning
0	clear Link Service Request (flag LSR)
1	don't change

43.14.9.4 Flow Control Microoperations

eTPU has *jump* and *call* microoperations to change microcode flow. Besides, eTPU has *dispatch jump* and *dispatch call* that can be used to implement a jump table. In *call* (or *dispatch call*) microoperation, the return address is saved in RAR register. If nested sub-routine calls are necessary, return address values have to be saved in a stack, usually implemented with DIOB register.

Flow Control microoperations are also provided to repeat a given microinstruction, to finish the current thread execution, and to halt the microengine.

43.14.9.4.1 Ending Current Thread - END

Microcode END field (1 bit) finishes current thread and allows other channels to be serviced. If END field is 0, the current instruction is completed and the thread is finished. END = 1 has no effect, and the next microinstruction is executed Any MDU operation (see MAC and Divide Unit (MDU)) that could be still pending when the thread is finished is left incomplete. END also releases any semaphore locked by the Engine.

43.14.9.4.2 Branch Operations

Branch operations can be **jump** or **call** . The target address of *jump* or *call* microoperations is always immediate and absolute. Branch microoperation is affected by FLS field (refer to Flush Pipeline).

43.14.9.4.2.1 Selecting Jump or Call Microoperations

The only difference between *jump* and *call* microoperations is that when a *call* is executed the value of PC or PC+1 (depending on flush, see Flush Pipeline) is saved in RAR register. The microcode field J/C (1 bit) selects whether *jump* or a *call* is executed, according to the following table.

Table 43-407. Jump / Call Selection - J/C

J/C	Meaning
0	jump
1	call

43.14.9.4.2.2 Branch Target Address

Microcode BAF field (14 bits) indicates the absolute address of a *jump/call target*.

43.14.9.4.2.3 Conditional/Unconditional Branch

Jump and *call* can be conditional or unconditional, depending on the BCC (6 bits) and BCF (1 bit) fields, as shown in Table 43-408 and Table 43-409 . BCF determines whether branch is taken when condition specified by BCC is true or false. When a branch condition uses the channel flags, the channel context is related to the channel number written in CHAN register.

Table 43-408. Branch Condition Inversion - BCF

BCF	Meaning
0	branch if condition determined by BCC is false
1	branch if condition determined by BCC is true

Table 43-409. Branch Condition Selection - BCC

BCC	Meaning	BCC	Meaning
001110	Flag 0	001111	Flag 1
100000	V ALU flag	110000	PSS channel flag
100001	N ALU flag	110001	PRSS channel flag
100010	C ALU flag	110010	"Less Than" ALU flag combination (signed) [1]
100011	Z ALU flag	110011	"Lower or Equal" ALU flag combination (unsigned)[2]
100100	MV MDU flag	110100	P[24]
100101	MN MDU flag	110101	P[25]
100110	MC MDU flag	110110	P[26]
100111	MZ MDU flag	110111	P[27]
101000	TDLA channel flag	111000	P[28]
101001	TDLB channel flag	111001	P[29]
101010	MRLA channel flag	111010	P[30]
101011	MRLB channel flag	111011	P[31]
101100	LSR channel flag	111100	PSTO channel flag
101101	MB flag MDU flag	111101	PSTI channel flag

Table continues on the next page...

Table 43-409. Branch Condition Selection - BCC (continued)

BCC	Meaning	BCC	Meaning
101110	FM[1] channel flag	111110	SMLCK semaphore flag
101111	FM[0] channel flag	111111	false
all other values reserved			

1. "less than" is a signed comparison, equal to the xor between ALU flags V and N; e.g., 0 < 0xFFFFFF tests as false (0 < -1).
2. "lower equal" is an unsigned comparison, equal to Z or C; e.g., 0 < 0xFFFFFF tests as true.

43.14.9.4.3 Dispatch Microoperation

Dispatch microoperation is an unconditional branch where the target address is always PC+P[31:24] (unsigned). Dispatch is affected by FLS field (refer to Flush Pipeline). Dispatch microoperation is defined by R/D field (2 bits, see the following table). Field R/D can also be used to define return from sub-routine (see Return From Subroutine).

Table 43-410. Return and Dispatch - R/D

R/D	Meaning
00	return from subroutine (see Return From Subroutine)
01	dispatch jump
10	dispatch call
11	don't change microinstruction flow

43.14.9.4.4 Return From Subroutine

When a subroutine *call* or a *dispatch call* microoperation is executed, the return address is saved in RAR register. To return from a subroutine a microoperation is available to load the contents of RAR register back to the PC. Fields R/D (2 bits) or RTN (1 bit, see the table below) can be used to return from subroutine. For R/D field, see Table 43-410 .

Return from subroutine microoperation is affected by FLS (see Flush Pipeline) when field R/D is used. Return execution through RTN always flushes the pipeline.

Table 43-411. Return from Sub-routine - RTN

RTN	Meaning
0	return with pipeline flush
1	do not return

43.14.9.4.5 Flush Pipeline

When a branch, dispatch or subroutine return microoperation is executed, the next microinstruction can be executed unconditionally before the flow change takes effect, since microengine has a three-stage pipeline. Executing the next microinstruction after a branch maximizes execution performance. This feature is controlled by field FLS (1 bit, see the table below). When FLS=0 the pipeline is flushed, so the next microinstruction placed after a branch is decoded as NOP if the branch is taken. If FLS=1, the microinstruction placed after the branch is executed, either if the branch is taken or not, as shown in the figure below.

Flush also controls which value is stored in RAR in a call: in case of no flush, it is the address of the branch/dispatch instruction + 2, even if RAR is the ALU destination of the instruction after the call; in case of a flush, it is the address of the instruction following branch/dispatch.

If a branch with no flush is followed by another branch with no flush, the instructions are executed in the following order:

1. first branch
2. second branch
3. first branch's destination instruction
4. second branch's destination instruction, and the flow proceeds normally from then on

The destination of the first branch must not be another flow changing instruction (branch, return or dispatch). Similar flows apply when returns or dispatches are used instead of branches. This scheme can be used to implement quick table look-ups with a dispatch replacing the first branch, for instance.

Table 43-412. Flush Pipeline - FLS

FLS	Meaning
0	flush pipeline when jump / call / dispatch jump / dispatch call / return is executed
1	do not flush pipeline when jump / call / dispatch jump / dispatch call / return is executed

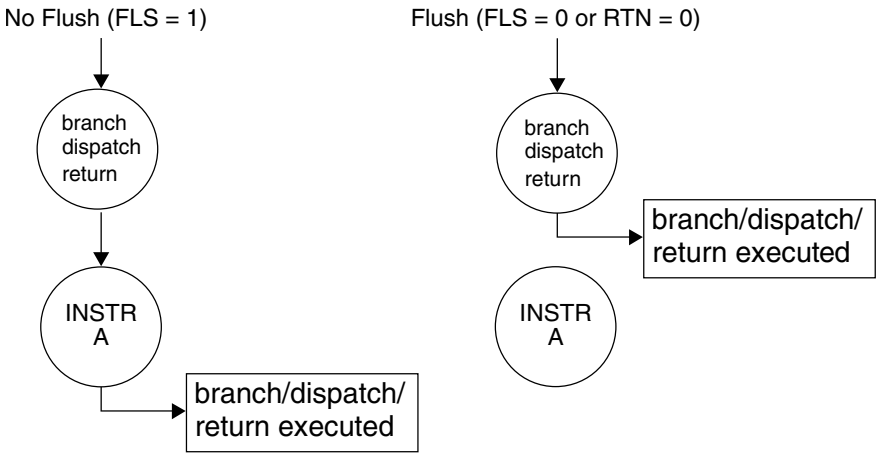


Figure 43-330. Flush Pipeline

43.14.9.4.6 HALT microinstruction

HALT is a microinstruction provided to implement software breakpoints (see Software Breakpoints). Note that HALT is coded as a microinstruction format, not a field (see Microinstruction formats). The execution of this instruction puts the microengine in halt state. For more information about the implications of microengine halt state, see Microengine Halt State . HALT is valid only if software breakpoints are enabled at the Debug interface . If software breakpoints are not enabled, HALT executes as a NOP and is treated as an Illegal Instruction (see Illegal Instructions).

43.14.9.4.7 NOP microinstruction

There is not a unique microinstruction with an assigned opcode to do No Operation. NOP microinstruction is achieved through any of the formats shown on Microinstruction formats where the user can assign to each individual field the corresponding value for "No Operation". However, to prevent future impacts of instruction changes on object code compatibility, the instruction value 0x4FFFFFFF should always be used for NOP.

43.14.9.5 Illegal Instructions

An instruction is considered illegal if any reserved field value is used, including when the fields marked rsv in the instruction formats (see Microinstruction formats are assigned value 0. A HALT instruction is exceptionally considered illegal when executed with software breakpoints disabled (see HALT microinstruction). Global Exception may be issued up to two microcycles after instruction fetch. The execution results of an illegal instruction on the microengine, channel logic or host interface are unspecified, except for the HALT case.

If the microengine decodes an illegal instruction, the following actions are taken:

- a Global Exception is issued.
- flag ILF1/2 on register ETPUMCR is set to indicate this occurrence to the Host.
- a breakpoint occurs, if NDEDI is present and configured to do so (for more details, see the NDEDI details).

43.14.9.6 Microinstruction Parallelism Issues

This section clarifies parallelism issues that arise when two non-commutative microoperations appear in the same microinstruction.

43.14.9.6.1 ALU Operations and Read Match Registers

ALU operations have only one destination register, but there is one case where source selection determines destination: read Match register in ERTA and ERTB registers. In this case if ALU destination is ERTA or ERTB a conflict arises. The ALU destination value overwrites the value read from the match registers.

43.14.9.6.2 ALU and SDM Operations

P and DIOB registers can be selected as destination by both ALU and SDM (read) microoperations in the same microinstruction. Since P and DIOB update from SDM data happens after P and DIOB update for ALU/MDU microoperations, the data read from SDM remains in P or DIOB after an operation when one of them is specified as destination for both ALU and SDM microoperations. In this case, the value loaded into P or DIOB is the one read from SDM. However, the ALU operation is executed and its flags are updated accordingly.

When P or DIOB is destination of an SDM read and also an ALU source at the same microinstruction, the value before the read is used for the ALU operation.

If DIOB is the ALU destination and P is loaded from SDM or vice-versa, no conflict occurs, and the result is the same as if operations occurred separately.

All the above also applies to Zero SDM operations.

When using P or DIOB as destination for ALU operations and also as source for a SDM write operation, the data written in SDM is the one calculated by ALU, which means it is possible to calculate a value and write it in an SDM address using only one microinstruction.

The old value of DIOB or the old value minus 4 (pre-decrement) is always used when DIOB is selected as address (indirect address mode), no matter if DIOB is selected as destination of either the SDM or ALU. For the value loaded into DIOB, refer to the following table.

Table 43-413. DIOB load from SDM and ALU

DIOB selected as SDM read destination?	DIOB selected as ALU destination?	DIOB load value
no	no	DIOB, --DIOB (pre-decrement), or DIOB++ (post-increment)
yes	no	SDM read data (post-inc and pre-dec ignored)
yes	yes	SDM read data (post-inc, pre-dec and ALU result ignored)
no	yes	ALU result (post-inc an pre-dec ignored)

When CHAN is the destination of an ALU operation in parallel with a channel-relative SDM access, the CHAN value used to calculate the address is the one before the ALU operation.

43.14.9.6.3 ERTA/B as ALU destination and ERWA/B

The value in ERTA and ERTB registers can be written in match registers of the selected channel by using fields ERWA and ERWB (Write Channel Match and UDCM Registers). If, at the same microinstruction, ERTA or ERTB is destination of an ALU/MDU microoperation, the value written in Match registers is the ALU/MDU result.

The same applies to UDCM when ERTA is the destination of an ALU operation and instruction fields ERWA and CMW are active .

If an ALU operation occurs in parallel with ERWA/B but ERTA/B are not the destination of an ALU/MDU operation, then UDCM and MatchA/B receives the ERTA/B value.

43.14.9.6.4 ERWA/B and MRLE

ERWA/B automatically sets the MRLEA/B channel latch, respectively (see Write Channel Match and UDCM Registers). Microinstruction fields ERWA/B independently sets MRLEA/B channel flags, regardless of MRLE.

43.14.9.6.5 CHAN assignment, Read Match and ERWA/B

When CHAN is a destination of an ALU operation it causes a read of the CaptureA/B register values into ERTA/B. The Capture registers loaded into ERTA/B are selected by the new CHAN value. The value of the CaptureA/B registers overwrites any read-match commanded simultaneously.

If CHAN assignment happens with an ERWA/B operation in the same instruction, the updated Match register(s) belong to the new selected channel.

43.14.9.6.6 Read Match and ERWA/B

If a read match operation is executed with ERWA/B in the same microinstruction, the MatchA/B registers receive the old values of ERTA/B, and the ERTA/B registers receive the old MatchA/B values simultaneously, i.e.: ERTA/B and MatchA/B swap their values.

If ERTA/B is the destination of an ALU operation at the same instruction, MatchA/B gets the ALU result (see ERTA/B as ALU destination and ERWA/B), but the ERTA/B not being written still receives the old MatchA/B values.

Note

Read match, ERWA/B and CHAN assignment can be active at the same instruction. Combining rules CHAN assignment, Read Match and ERWA/B and Read Match and ERWA/B, the result is: ERTA/B receives the CaptureA/B values of the new CHAN value, and MatchA/B of the new channel receives the old ERTA/B value(s).

43.14.9.6.7 Stack accesses and ALU operations

Post-increment is ignored in a stack operation (field STC) if DIOB is loaded from SDM: DIOB keeps the value read from SDM. Pre-decrement is ignored in a stack operation (field STC) if DIOB is destination of an ALU operation, for DIOB load value, but not for DIOB as address. Post-increment/pre-decrement remains valid in all other situations. These rules can be summarized in the following equivalent C code, and in Table 43-413 .

```
DIOB = *DIOB[15:2];       // read without posinc/predec
*DIOB[15:2] = DIOB;       // write without posinc/predec
DIOB = *(--DIOB[15:2]);   // read with predec
DIOB = *DIOB[15:2];       // read with posinc (ignored)
*(--DIOB[15:2]) = DIOB;   // write with predec
*DIOB[15:2]++ = DIOB;     // write with posinc (value written is before
increment)
```

43.14.9.6.8 SRC and ALU/MDU operations

If operation SRC is active (field SRC = 0) and register SR is selected as destination of an ALU operation, the value of the ALU operation prevails over the shifted value.

The value of SR used as source in the ALU/MDU operation is the one before the shift.

43.14.9.6.9 Semaphore lock/free and SMLCK branch condition

When the SMLCK branch condition is tested at the same microinstruction of a semaphore lock or free, the condition is evaluated after the semaphore action (either free or lock) is taken.

43.14.9.6.10 Dispatch and SDM Read

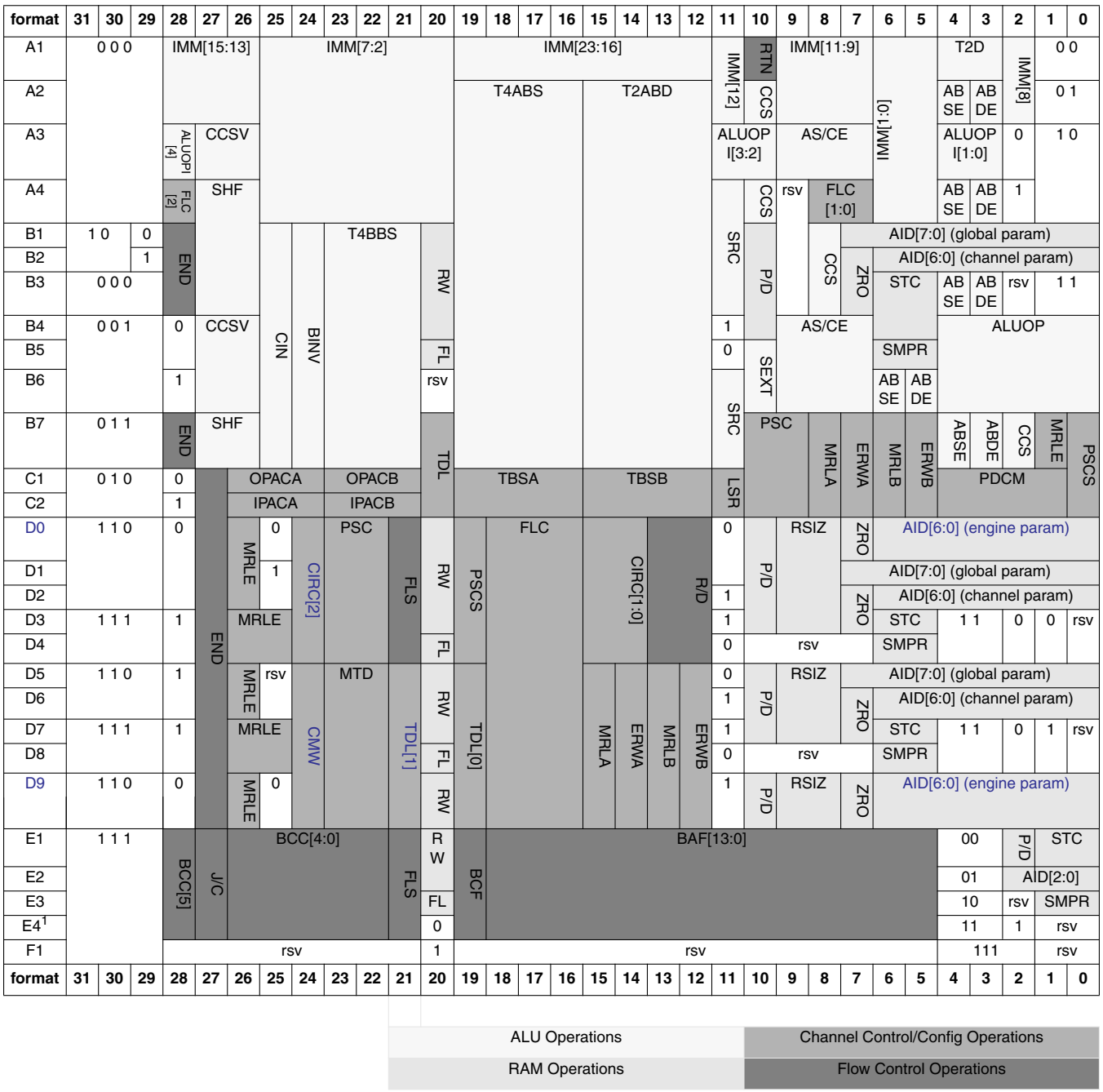
When the most significant byte of P is read from SDM (read 8 msb bits or 32 bits) and a dispatch instruction is executed simultaneously, the dispatch target address is calculated upon the P value before the read.

43.14.9.6.11 CHAN Assignment, PSC/PSCS, and clear MRLEA/B, MRLA/B, TDLA/B

When clear MRLEs, MRLA/B or TDLs is done and a CHAN assignment is done at the same time, the flag selected by the old CHAN value is cleared in the channel, but the branch conditions receive the state of the flags selected by the new CHAN.

When a pin action is commanded through PCS/PSCS and a CHAN assignment is done simultaneously, the output signal affected is selected with the old CHAN value.

43.14.9.7 Microinstruction formats



NOTES:
1 format E4 is deprecated; the same result can be obtained with format E1 with field STC=11

Figure 43-331. Microinstruction formats

All bits marked "rsvd" are reserved, and must be coded as 1.

43.14.10 Shared Subsystems Support

Shared Subsystems support allows eTPU to control other modules, either independently or sharing control with the host. A notable example of shared subsystem is the Reaction Module (see the **Reaction Module chapter** for details), which is connected to the eTPU as shown in the figure below. The host CPU accesses the Reaction Module through the eTPU, which can also access the same registers.

When bit SSMAE=1 in the CCR_ENG1 or CCR_ENG2 register, the eTPU channel's threads can control shared subsystems by accessing their registers, which are mapped onto the higher 16 Kbytes of the microcode data address space (byte addresses/DIOB values 0x4000 to 0x7FFF, equivalent to parameters 0x1000 to 0x1FFF), as shown in Figure 43-333. The shared subsystem registers are accessible by microcode only with the indirect address mode (see SDM Addressing Modes).

When ETPUCxCR bit SSMAE=0, the channel's threads can access only the SDM, mirrored throughout the whole DIOB address range .

The same Shared Subsystem register address space is accessible by the host through the Shared Subsystem Area within the eTPU memory map (see Figure 43-333 and Memory Map), allowing shared control and/or supervision between the host and the eTPU. The arbitration mechanism between host and eTPU accesses is the same used for SDM accesses (see SDM Arbitration). The CDC (see Coherent Dual-parameter Controller - CDC) cannot be used for coherent transfers between SDM and the Shared Subsystem Area, or within the Shared Subsystem Area itself.

Note

The detailed register map within the Shared Subsystem Area depends on the subsystem module.

A Shared Subsystem may be divided into submodules or channels, hereafter called **Shared Subsystem Units** (or **Shared Units** for short), which can be individually controlled by an eTPU function (see Functions and Threads)

A Shared Unit is logically linked with an eTPU channel through the ETPUSSSnIR configuration[34]. There is one ETPUSSSnIR register for each Shared Unit to configure its interface with the eTPU (see eTPU Shared Subsystem n Integration Register, SSSnIR). The Shared Unit uses the Link Service Request (LSR) mechanism to request service to the eTPU. The Link Service Request is targeted to the channel/engine configured in the ETPUSSSnIR fields LENG and LCHAN. The function assigned to this channel must be written to service the Shared Unit. The Shared Unit link service request may have the same source as an interrupt to the host.

34. However, in the case of Reaction Module, the ETPUSSSnIR configuration does not determine which eTPU channel output enables the Reaction Channel modulation; this is defined in the Reaction Module configuration.

Note

Shared Unit link service requests are level-sensitive to the eTPU. If it comes from an interrupt source, the link service request stays asserted as long as the interrupt is asserted and enabled, so the interrupt request must be cleared before the clear of the LSR flag in the link service request thread.

The ETPUSSSnIR field LENG can also be used by the host to disable Link Service Requests to the eTPU. The eTPU microcode can only disable LSRs from the Shared Unit through configuration specific to the Shared Unit itself.

Note that more than one Shared Unit can generate LSRs to the same eTPU channel, in which case the LSR source can be determined by microcode through the Shared Unit status register(s). An eTPU channel configured as a Shared Unit LSR target can also receive Link Service Requests from other eTPU channels or Shared Units. The same LSR service and flag behavior found in links between eTPU channels applies to Shared Unit links (see Channel Link).

The table below summarizes the mapping of Reaction Module resources into Shared Subsystem Support.

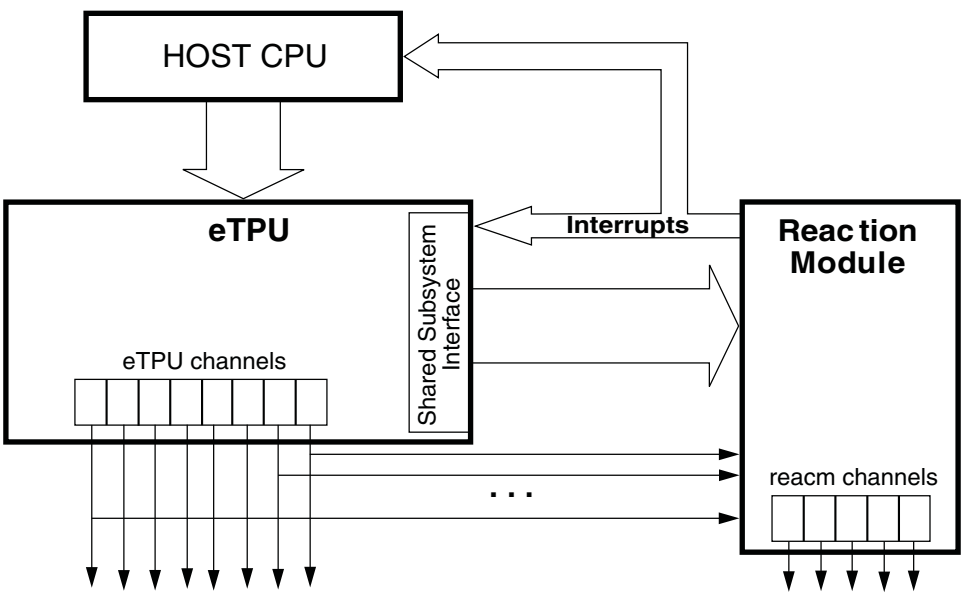


Figure 43-332. Shared Subsystem Block Diagram example: Reaction Module

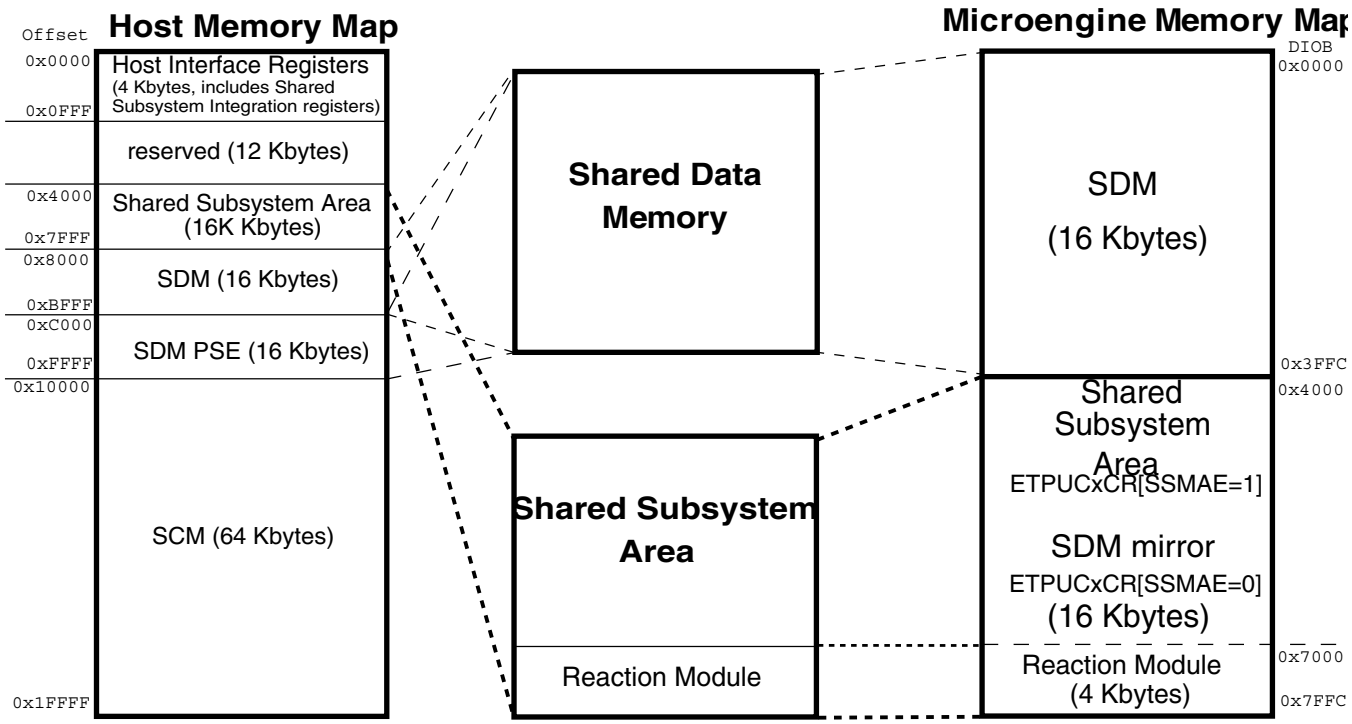


Figure 43-333. Shared Subsystem Memory Mapping

Table 43-414. Reaction Module Shared Subsystem Support

Reaction Module Resource	Host Mapping	eTPU/Microengine Mapping
Register access	eTPU Base + 0×7000	DIOB address 0×7000 (indirect address mode)
Interrupts	See Interrupts chapter	Link Service Requests mapped through Subsystem Integration registers ETPUSSSnIR
DMA Requests	See eDMA chapter	Not applicable

43.14.11 Development Support Features

Following sections describe several features available to support development. Most debug features are accessible through a separate debug bus, and are not available through registers in the standard eTPU memory map. The details of the access to this interface are MCU-dependent (see Debug Interface chapter). The microengines can also be put in halt state by a device-level debug request..

NOTE

The behavior of the debug support logic is not specified in the event of a non-correctable memory error on either SCM or SDM (see Memory Error Support).

43.14.11.1 Nexus Class 3 Support

NDEDI is an IP block designed to support Nexus functionality for the eTPU. It provides a Nexus class 3 or class 1 debug interface, depending on the MCU (see Configuration section). Nexus is a development support external interface defined by the **IEEE standard ISTO 5001-1999**. More details about it can be found in the NDEDI details.

Some of the next subsections describe debug features provided by the eTPU microengine debug support combined with the NDEDI block.

43.14.11.2 Microengine Halt State

Halt is a microengine state where it suspends execution during a thread, or does not start executing a scheduled thread from idle state. While Idle State is entered from END execution without any other scheduled thread, microengine enters Halt State by any of the following events:

- execution of the HALT microinstruction (software breakpoint).
- external halt request through the Debug Interface (includes Nexus breakpoint request via EVTI input pin - see Nexus Class 3 Support).
- the other Engine enters halt state and they are configured to halt simultaneously (bit HTWIN is asserted via Nexus Interface).
- System level device debug request assertion and NDEDI register NDEDIETPUx_DC field CBI=1. If same register's field CBT=1, microengine halts at the next time-slot boundary, if CBT=0 it halts immediately. As a particular case, microengines come halted out of reset if device debug request is asserted, since CBI reset value is 1. Microengine does not execute out of reset, either in halt (device debug request asserted) or idle state (device debug request negated), but halt enables several other features (see below).

- occurrence of any of the hardware breakpoint conditions. See Hardware Breakpoints for details.
- execution of a single-step microinstruction: microengine returns to halt state after executing a single microinstruction while in halt state. See Single-step Execution and Forced Microinstruction Execution for details.

When microengine enters halt state, it automatically triggers the following actions:

- suspends input signal sampling and filters (respective Engine channels only), if NDEDI_ENGINEx_DC[PINS] = 1 at the Debug Interface.
- releases the SDM arbitration for Host or CDC accesses, no matter if microengine was halted in the middle of a dual parameter (back-to-back) access.
- stops TCR1/2 clocks of the respective Engine, if signal ndedi_stop_tcr is asserted at the Debug Interface.
- if the other Engine is also in halt state or stopped, allows turning the eTPU Module Configuration Register (eTPU_SCR_MCR) VIS bit to 1.

If all halt conditions are cleared when VIS=1, microengine(s) keep on halt state until VIS=0, when it automatically exits halt state, except on single-step (see Single-step Execution), so that single-step execution is ignored while VIS=1.

MDU continues executing until it finishes any ongoing operation even if microengine is in halt state, except when the halted instruction is an END.

There are two kinds of halt state, depending on the previous microengine state when halted:

1. **halt_idle**, if the engine was not executing a thread when halted; the engine cannot leave halt_idle to fetch instructions, so one cannot single-step or follow a program flow; it can, however, execute forced instructions (see Forced Microinstruction Execution).
2. **halt_exec**, if the engine was executing a thread when halted. The engine can single-step and continue a program flow from halt_exec.

When microengine exits halt state, any dependable action is suspended and, if exiting halt_exec, the instruction pointed by the PC is fetched, while the instruction already fetched before halt is executed. Note that both the PC and the prefetched instructions can be modified during halt state, with a forced execution of a branch instruction (see Forced Microinstruction Execution).

43.14.11.3 Hardware Breakpoints

Microengine can enter halt state through a command from the Debug Interface, configuring a hardware breakpoint. Hardware breakpoints can halt the microengine on specific conditions, listed below. These conditions depend on NDEDI configuration (for more details see the NDEDI details).

- CHAN register assignment (only by microcode, not by time slot transition).
- SDM read and/or write to a given address and/or write data. The breakpoint is always qualified by the SDM address, but the following variations are allowed:
  - break on write only, read only, or read-and-write.
  - break on higher-byte write data value, lower 24-bit write value, full word (32-bit) write value, or regardless of data. Break on read data is not supported.
- PC (program counter) value.
- Beginning of a thread with a Host Service Request pending.
- Beginning of a thread with a Link Service Request pending.
- Beginning of a thread with a Match Service Request pending.
- Beginning of a thread with a Transition Service Request pending.
- End of a thread.
- Illegal instruction execution.

All these conditions can also be qualified by the value of the CHAN register.

NOTE

Data breakpoints are not supported on Shared Subsystem Area accesses (see Shared Subsystems Support).

Hardware breakpoints halt the microengine before the halted instruction is executed, except on data breakpoints. Data breakpoints allow the instruction that performs the data access to be executed before the microengine is halted.

Note

Data breakpoints in eTPU versions before eTPU2 halt the breakpoint instruction before its execution in all cases, including data breakpoints.

On any of these conditions, halt of one microengine does not depend on the halt of the other, unless the other engine is configured to do so, via Nexus Interface. Occurrence of any of these conditions halts the microengine, i.e., the conditions are logically "ored" together, and they can be individually enabled.

While in halt state, the microengine can also execute any forced microinstruction not in the normal program flow (see Forced Microinstruction Execution) or, if in halt_exec, in single-step (see Single-step Execution).

There are situations when requests for stopping an engine, breakpoint and service can occur simultaneously. Breakpoint requests always prevails over a stop request (eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) or eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2) bit MDIS=1 or device debug request=1). When the eTPU is idle: stop request prevails over Service Request if there is not a hardware breakpoint request; a hardware breakpoint request leads to debug mode immediately if there is no Service Request, and after TST if there is Service Request (regardless of stop requests). The rules above are summarized in the the following table, showing the destination state of the microengine in each situation.

Table 43-415. Breakpoint, Stop and Service Requests resolution from Idle

Breakpoint Request	MDIS	Service Request	Final State
no	0	0	Idle
no	1	0	Stop
no	0	1	TST
no	1	1	Stop
Yes	0	0	Halt_idle
Yes	1	0	Halt_idle
Yes	0	1	TST[1, 1]
Yes	1	1	TST[1]

1. breaks after TST, if signal ndedi_sync_break is still asserted.

When a thread is ending, it goes to Idle or TST only if there is neither a hardware breakpoint request nor a request to stop (eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) or eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2) bit MDIS=1 or device debug request=1). When thread is ending and there are simultaneous hardware breakpoint and stop (MDIS=1 or device debug request=1) requests, hardware breakpoint prevails and the engine enters Debug mode (Halt_idle state).

If the engine entered Debug mode after a thread finished (Halt_idle state) and a "go" command comes from the debug interface, the engine state machine goes to Idle and the rules above apply. It means that if a "go" is issued in Halt_idle state with MDIS=1, the engine goes to Idle for one microcycle and then stops (if MDIS or device debug request keeps asserted and there is no other breakpoint request).

Note

Hardware breakpoint requests are ignored for the first microinstruction executed when microengine leaves halt.

43.14.11.4 Hardware Watchpoints

Debug Interface allows watchpoints on the same conditions available for hardware breakpoints (see Hardware Breakpoints).

Note

Data breakpoints are not supported on Shared Subsystem Area accesses (see Shared Subsystems Support).

43.14.11.5 Software Breakpoints

A software breakpoint occurs when microengine executes a HALT microinstruction. Any number of software breakpoints can be set in code, usually replacing an active microinstruction.

Like any other microinstruction, HALT increments the PC and pre-fetches the next instruction. So, before the halt state is suspended, if the original program flow must be followed, the original instruction at the HALT address must be executed, regardless if the software breakpoint is removed (replacing HALT by the original microinstruction) or not. Follows the procedure to resume execution from a software breakpoint:

1. restore the original instruction in SCM (replace HALT).
2. force a jump with flush to the original instruction (see Forced Microinstruction Execution).
3. if the software breakpoint must be kept: single-step and replace the original instruction with a HALT.
4. let the flow continue, issuing a GO command (leaving halt state).

Special care must be taken if HALT is followed by another HALT, and the second HALT is removed when microengine was halted by the first one. In this case, replacing the second HALT with the original microinstruction is not enough to remove the second breakpoint, because the second HALT was already prefetched and would be executed anyway when halt was suspended. The debugger must also do a forced execution of unconditional branch with flush to the original microinstruction address. That will clear the pipeline, replacing the prefetched instruction with a NOP, and load PC with the address of the removed breakpoint. So, when halt state is suspended, the original microinstruction will be fetched while NOP is executed, and program flow continues normally from then on.

Note

A HALT instruction placed after a no-flushing branch, dispatch or return may be a problem from the debugger application standpoint: after the HALT is executed, the eTPU debug interface informs the address of the branch/dispatch/return destination, and the debugger application has no direct way to identify which HALT instruction was executed, if multiple HALTs lead to the same address. This can be solved if the debug support block (NDEDI) has a register holding the address of the last instruction executed, otherwise one should forbid non-flushed HALT instructions. See the NDEDI details.

Software breakpoint setting and removal is possible only with SCM RAM implementations or ROM implementations with SCM RAM emulation (see). There is only one way of inserting software breakpoints into SCM RAM: writing bit VIS=1 in register ETPUMCR eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1), and then accessing SCM as an ordinary RAM from the slave bus. This can be done only if both Engines are halted or stopped.

43.14.11.6 Single-step Execution

When microengine is already in halt_exec state, it can run the next microinstruction in the normal program flow and get back to halt state. PC is incremented, or assigned the BAF value in a branch with satisfied condition. Note that the executed instruction was already prefetched in the instruction pipeline, and a new microinstruction is fetched during its execution. The prefetched instruction may be cleared during halt state by the forced execution of a branch with flush (see Forced Microinstruction Execution), making single-step execute a NOP instead of the next instruction in the program flow.

Single-step execution is controlled by the debug interface, and is a feature available from Nexus if eTPU is connected to the NDEDI block (see NDEDI details).

The single-step execution of a NOP instruction can be useful to control input signal sampling and filtering, if NDEDI_ENGINEx_DC[PINS] = 1 at the Debug Interface (see NDEDI details). Single-step does not happen if VIS=1.

43.14.11.7 Forced Microinstruction Execution

When microengine is already in halt state (either halt_idle or halt_exec), it can run forced microinstructions through the debug interface. This feature is available from Nexus if eTPU is connected to the NDEDI block (see NDEDI details). The microinstruction, specified by the user, is not fetched from SCM and comes directly from the debug interface. MDU start commands issued by forced instructions are executed, and the MDU runs the operation until the end, independently of the halt state. The microinstruction field END is ignored.

During forced execution of any instruction except Branches, Returns and Dispatches, the PC does not change, and the prefetched instruction in the pipeline is bypassed, but not discarded. When halt state is suspended, the prefetched instruction is executed and the instruction pointed by the PC is prefetched in parallel (two-stage pipeline).

Forced execution of a Branch, Dispatch or Return loads the PC with the BAF field (if branch condition is satisfied), PC+P or RAR, respectively. If branch condition is not satisfied, PC value stays unaltered. The flush control (field FLS) also works, so that a successful forced branch with flush replaces the prefetched instruction with a NOP. So, to clear the instruction pipeline during halt, all one has to do is an unconditional branch to the desired address with flush. HALT instructions must not be executed as forced.

Forced operations that depend on the serviced channel are unspecified when executed in halt_idle and must be avoided.

Note

Forced instructions must not be commanded when eTPU Module Configuration Register (eTPU_SCR_MCR) bit VIS=1.

43.14.11.8 Microengine Register Access

eTPU provides no direct access to microengine and channel registers from the slave bus or any other interface. However, these registers can be read and written in halt state by executing forced microinstructions (see Forced Microinstruction Execution). Immediate data microinstructions may be used to set register values. Some registers are not

selectable for immediate data destination, so intermediary register(s) - notably P - may have to be used to carry the desired new value to the target register in two or more microinstructions. Usually the previous values of intermediary register(s) must be previously saved and restored after the whole operation.

Similar procedures apply for register reads: their contents must be dumped to SDM, where they can be read from the slave bus.

43.14.11.9 Microengine Flag Access

Microengine halt state allows reading the branch conditions flags through forced microinstructions or, more easily, through the NDEDI register NDEDI_ENGINEx_CFSR (see the NDEDI details). Flag conditions set by the user are seen by microengine for the next microinstruction execution. The flag set options are limited by the possibilities of forced microinstruction execution.

If the eTPU runs (not single-stepping) after exiting the halted state, the conditions modified during halt may remain only for the first microcycle after the halted state. After the first microcycle, branch conditions are altered only according to their regular update scheme.

43.14.11.10 Microengine Stall

Microengine can get into a stall state, attending a request from a debug interface signal assertion. The reason for a Stall request from NDEDI (or from any other debug support block) should be a temporary lack of resources, for instance queue full. During stall the microengine suspends execution, but all the other Engine logic continues operating: time bases, angle logic, channel logic, input sampling and filters. Stall differs from Halt, not enabling any of the debug features that Halt enables (see Microengine Halt State). It also does not break an atomic microengine access, unlike halt.

The Microengine can be stalled when idle and from the moment TST ends, before executing the first thread microinstruction, until just before the last thread microinstruction is executed. Stall requests are ignored in any other occasions. Microengines in a dual engine system can be independently stalled. If a forced end is issued when microengine is in stall coming from execution, the END is executed only when the microengine resumes execution from stall.

43.14.12 Safety Features

This section describes embedded safety features: MISC and memory error support.

The Multiple Input Signature Calculator - MISC - is an SCM test feature accessible through registers ETPUMCR and ETPUMISCCMPR (see eTPU Module Configuration Register (eTPU_SCR_MCR) and eTPU MISC Compare Register (eTPU_SCR_MISCCMPR)). MISC allows SCM test "on the fly", i.e., while eTPU is running, with no impact on eTPU functionality or performance.

Memory Error support features comprises SCM and SDM error detection, correction, report, and soft error.

43.14.12.1 SCM Test - Multiple Input Signature Calculator

The Multiple Input Signature Calculator (MISC) comprises special hardware that sequentially reads all SCM positions and calculates, in parallel, a 32-bit signature from a 32-input CRC signature calculator with the following polynomial:

$$1 + x^{1} + x^{2} + x^{22} + x^{31}$$

A complete description of the signature calculation procedure can be found in MISC Algorithm.

Once started by the Host the MISC runs continuously, restarting after the completion of each cycle, when it sets the ETPUMCR register flag SCMMISC (see eTPU Module Configuration Register (eTPU_SCR_MCR)). The average time for a MISC calculation can be measured by checking SCMMISC state at regular intervals, incrementing a counter and clearing SCMMISC if it is set.

MISC accesses to the SCM array are executed if none of the engines is accessing the SCM, to avoid degradation of the microengine performance: it happens while no channel is being serviced. An ongoing MISC operation can be aborted by writing 0 to SCMMISEN.

The Host must load the register ETPUMISCCMPR (see eTPU MISC Compare Register (eTPU_SCR_MISCCMPR)) with the expected value to be found at the end of the MISC cycle, and then start the signature calculation writing bit SCMMISEN=1 in register ETPUMCR (see eTPU Module Configuration Register (eTPU_SCR_MCR)). MISC zeroes the signature accumulator and starts reading SCM data and calculating the signature. After last SCM position is read, MISC compares the value in signature accumulator against the value in ETPUMISCCMPR: if there is a mismatch MISC stops, a Global Exception is issued and the bit SCMMISF in register ETPUMCR assumes value

1. If no mismatch is found, MISC repeats the procedure automatically. When signature is being calculated, SCM address starts at the last SCM address and counts down to 0. The conditions for executing a MISC operation are (see also Table 43-334):

- Both microengines in idle state (no channel is being serviced) or stopped, in any combination (e.g., engine 1 idle with engine 2 stopped).
- ETPUMCR bit VIS = 0.
- ETPUMCR bit SCMMISEN=1.

Note that MISC can run regardless of SCM implementation type (RAM or ROM).

If SCMMISEN=0 or VIS=1, the MISC logic stays at its initial state, with address counter pointing to the last SCM position and accumulator reset.

MISC can also be used to perform single-bit error fixes in the SCM. For more details, see Error Fixing).

43.14.12.2 Memory Error Support

Error Correction Code (ECC) support for SDM and SCM provides guaranteed single bit error correction and double bit error detection. Error detection and correction is performed on read accesses and on write accesses less than 32-bit wide to the SCM.

SDM ECC is calculated per byte, 5 bits wide each, totalling 20 ECC bits for each 32-bit word. That allows the ECC to be updated without a read-modify-write operation on writes of any size.

Error is not checked on bytes not selected on SDM read accesses, either by the Host, CDC or microengine. For instance, the most significant byte is not checked for errors on a microengine 24-bit read access.

Error detection also works for host accesses to the PSE area (see Parameter Sign Extension Area). In this case, error is not detected in the most significant byte (bits 31:24). Also, because of sign extension, error is always detected on the bits 23:16, regardless if they are selected, whenever the most significant byte is read.

SCM ECC is 7 bits wide, comprising the whole 32-bit word. Read-modify-write is automatically made when 16-bit or 8-bit wide writes are made into the SCM in a host access. SCM errors are also detected on Entry Table accesses during TST (see Time Slot Transition) and on MISC accesses, which are considered microengine accesses with regards to interrupt generation and error reporting.

43.14.12.2.1 Error Detection Disabling

The error detection logic in SDM and SCM can be disabled by setting the eTPU Memory Error Control Register (eTPU_MESR_MECR) bits DEDD and CEDD, respectively. The disabled state turns ineffective any action due to error detection, including error correction. Note that error detection is always enabled out of reset.

43.14.12.2.2 Memory Error Flags

SDM single (correctable) and non-correctable bit errors are flagged in eTPU Memory Error Status Register (eTPU_MESR_MESR) bits DCERR and DNCERR, respectively..

SCM correctable and non-correctable errors are flagged in the ETPUMESR register bits CCERR and CNCERR, respectively.

Double bit errors are guaranteed to be flagged, while three or more simultaneous bit errors in an SDM byte or SCM word are not. An error flagged as non-correctable can be in two or more bits.

Overrun flags DCOVR and CCOVR indicate, for SDM and SCM respectively, when a correctable error occurs when DCERR, CCERR are already asserted. Similarly, overrun flags DNCOVR/CNCOVR assert when a non-correctable error occurs when DNCERR/ CNCERR are asserted already.

43.14.12.2.3 Memory Error Reports

The address, syndrome and data of a non-correctable SDM access can be sampled into the registers eTPU Data Error Report Address Register (eTPU_MESR_DERAR), eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR), and eTPU Data Error Report Data Register (eTPU_MESR_DERDR), respectively.. The register ETPUDERAR also captures the SDM read access originator, whether the host, CDC, or specific eTPU channel service (see ETPUDERAR register, ETPUDERSR register, and ETPUDERDR register).

Similarly, non-correctable SCM accesses address, syndrome and data are stored in the registers eTPU Code Error Report Address Register (eTPU_MESR_CERAR), eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR), and eTPU Code Error Report Data Register (eTPU_MESR_CERDR), respectively. The register ETPUCERAR also captures the SCM read access originator, whether the host, MISC, or specific eTPU channel service (see ETPUCERAR register, ETPUCERSR register, and ETPUCERDR register). The occurrence of a non-correctable error overwrites the report register values from a correctable error, either on SDM (DCERR=1) or SCM (CCERR=1).

The last address, syndrome and data of a correctable SDM access can also be sampled into registers ETPUDERAR, ETPUDERSR and ETPUDERDR, but only if the ETPUMESR bit DNCERR is 0. That prevents access address, syndrome and data from a correctable error to overwrite the same information from a non-correctable access.

Similarly, the last address, syndrome and data from a correctable SCM access are stored in registers ETPUCERAR, ETPUCERSR and ETPUCERDR only if ETPUMESR bit CNCERR is 0.

For either SDM or SCM, a new correctable error overwrites the report registers' values from an earlier correctable error, but report values from a non-correctable error are not overwritten, as long as bit DNCERR/CNCERR is asserted, neither by a new correctable error nor by a non-correctable one.

The sampling of SDM and SCM error report registers can be disabled through the ETPUMECR bits DRDIS and CRDIS, respectively. When DRDIS (CRDIS) is asserted, the update of the SDM (SCM) error report registers is disabled, but not of ETPUMESR bits DCERR, DNCERR (CCERR, CNCERR). DRDIS, CRDIS must be set before reading the respective report registers to assure their coherent reading, preventing them to be updated between reads.

43.14.12.2.4 Error Detection and Wait-states

Microengine SDM accesses do not take any wait-state due to ECC support. Host SDM read accesses take two wait-states minimum[35, 35] in MCUs with SDM ECC support, regardless if an error was detected or not, if the SDM error detection is enabled (eTPU Memory Error Control Register (eTPU_MESR_MECR) bit DEDD=0). Host SDM read accesses take a minimum of one wait-state if the SDM error detection is disabled (ETPUMECR bit DEDD=1). Host SDM write accesses take one wait-state minimum, regardless of ECC support presence or enabling.

Host SCM read accesses have two wait-states[35], regardless if an error was detected or not. Writes from the host to the SCM take one wait-state on 32-bit accesses, and one additional wait-state (two wait-states total[35] on 16-bit and 8-bit wide accesses. Those figures are not affected by the SCM error detection disabling (ETPUMECR bit CEDD).

43.14.12.2.5 Memory Error Interrupts

For **microengine accesses**, the Global Exception interrupt asserts whenever a non-correctable error is detected, either on SCM or SDM (see Interrupts and Data Transfer Requests). These errors are flagged by eTPU Module Configuration Register

35. one wait-state more, compared with eTPU versions before eTPU2 and previous eTPU2 without ECC support

(eTPU_SCR_MCR) bits SCMERR and SDMERR, respectively.. Single bit errors are corrected and activate the Memory Error Correction Interrupt if it is enabled (see Interrupt Types and Sources), but do not generate Global Exceptions.

For **host accesses** a bus error is issued whenever a non-correctable error is detected, either on SCM or SDM (but not through the CDC) and it also generates a Global Exception. Single bit errors are corrected (if the error correction feature is available) and do not cause bus transfer errors, but assert the Memory Error Correction Interrupt if it is enabled (see Interrupt Types and Sources).

43.14.12.2.6 Effect of Errors on Microcode Execution

Correctable errors, either on SCM or SDM, do not affect microcode execution if detection/correction is enabled.

The executing thread is affected by a non-correctable SDM error as follows:

- If ETPUMECR bit DTEND = 1: on SDM errors due to a read operation, if the instruction is not also an END, the following instruction automatically executes an END, and its operations may be executed, totally or partially. On SDM errors during the TST preload, the thread automatically ends on the first instruction, if the error occurs on a preload parameter selected by the entry point bit PP (see Entry Point Format section.) The first instruction may be executed totally or partially. Data read into P or DIOB may be corrupted in either case.
- If ETPUMECR bit DTEND = 0: the TST or thread continues the execution flow normally. Data read into P or DIOB may be corrupted in either case.

The executing thread is affected by a non-correctable SCM error as follows:

- If ETPUMECR bit CTEND=1: on SCM errors, the corrupted instruction is not executed, instead replaced by a simple END. Due to instruction prefetching, an error may be detected and yet not affect the thread execution. However, an error detected in a prefetched instruction may cause error report and global or memory error interrupts, even if not executed. If a non-correctable error occurs on the entry table read during TST (see Time Slot Transition section), the first instruction in the thread is replaced with a simple END, a global exception is issued and the reports and flags are normally updated.
- If ETPUMECR bit CTEND=0: the execution flow continues normally, and the results depend on the particular instruction corruption.

Table 43-416. Error Support Summary if CEDD = DEDD = 0

Action	SCM		SDM	
	single bit error	double bit error	single bit error	double bit error
Memory Error Correction Interrupt	Asserts	Not asserted	Asserts	Not asserted

Table continues on the next page...

Table 43-416. Error Support Summary if CEDD = DEDD = 0 (continued)

Action		SCM		SDM	
		single bit error	double bit error	single bit error	double bit error
Global Exception		Not asserted	Asserts	Not asserted	Asserts
Bus Error (host reads)		Not issued	Issued	Not issued	Issued
Thread	CTEND = 0	Normal execution	Potential erroneous execution[1]	Normal execution	see DTEND
	CTEND = 1		Terminates before erroneous execution[2]		
	DTEND = 0		see CTEND		Potential erroneous execution
	DTEND = 1				Terminates after erroneous read[3]
Code Error Report Registers	CNCERR = 0	Updates ETPUCERAR, ETPUCERSR, ETPUCERDR		Not related	
	CNCERR = 1	ETPUCERAR, ETPUCERSR, ETPUCERDR not updated			
Data Error Report Registers	DNCERR = 0	Not related		Updates ETPUDERAR, ETPUDERSR, ETPUDERDR	
	DNCERR = 1			ETPUDERAR, ETPUDERSR, ETPUDERDR not updated	

1. The thread executes correctly if the error is detected on an instruction fetched but not executed.
2. The erroneous instruction is not executed at all, and replaced by a simple END.
3. The instruction following the erroneous read may be executed totally or partially.

43.14.12.2.7 Error Correction Code (ECC) and Syndrome Definition

SDM ECC is composed of 5 Hsiao code[36] bits for each data byte. **SCM ECC** is composed of 7 Hsiao code bits for each 32-bit word.

Table 43-417 shows the encoding of the SDM syndrome and global parity fields as they read on the eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR) register. There is one syndrome and one global parity for each byte in an SDM word, numbered SYND0 to SYND3 from the least to the most significant.

Table 43-417. SDM Syndrome Definition

SYNDn (hex)	Result	SYNDn (hex)	Result
0E	error on data bit 8n+7	10	error on **code** bit 8n+4
15	error on data bit 8n+6	08	error on **code** bit 8n+3
07	error on data bit 8n+5	04	error on **code** bit 8n+2

Table continues on the next page...

36. Hsiao, M.Y;.A Class of Optimal Minimum Odd-weight-column SEC-DED Codes; IBM Journal of Research and Development http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5391627

Table 43-417. SDM Syndrome Definition (continued)

SYNDn (hex)	Result	SYNDn (hex)	Result
1A	error on data bit 8n+4	02	error on **code** bit 8n+1
19	error on data bit 8n+3	01	error on **code** bit 8n
13	error on data bit 8n+2		
16	error on data bit 8n+1	00	no error
1C	error on data bit 8n	any other	non-correctable error

The following table shows the definition of the SCM syndrome field as it reads on the eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR) register.

Table 43-418. SCM Syndrome Definition

SYND (hex)	Result	SYND (hex)	Result	SYND (hex)	Result
40	error on **code** bit 6	58	error on data bit 25	46	error on data bit 11
20	error on **code** bit 5	0B	error on data bit 24	4A	error on data bit 10
10	error on **code** bit 4	54	error on data bit 23	52	error on data bit 9
08	error on **code** bit 3	15	error on data bit 22	62	error on data bit 8
04	error on **code** bit 2	16	error on data bit 21	13	error on data bit 7
02	error on **code** bit 1	34	error on data bit 20	29	error on data bit 6
01	error on **code** bit 0	25	error on data bit 19	31	error on data bit 5
		26	error on data bit 18	43	error on data bit 4
49	error on data bit 31	64	error on data bit 17	45	error on data bit 3
0D	error on data bit 30	2C	error on data bit 16	19	error on data bit 2
0E	error on data bit 29	1A	error on data bit 15	51	error on data bit 1
38	error on data bit 28	23	error on data bit 14	61	error on data bit 0
4C	error on data bit 27	2A	error on data bit 13	00	no error
1C	error on data bit 26	32	error on data bit 12	any other	non-correctable error

43.14.12.2.8 Error Injection

Errors can be injected into the SDM read data path through the eTPU Data Error Injection Address Register (eTPU_MESR_DEIAR), eTPU Data Error Injection Data Pattern Register (eTPU_MESR_DEIDPR) and eTPU Data Error Injection Parity Pattern Register (eTPU_MESR_DEIPPR) registers. They define bit flips on accesses at the specific address specified in ETPUDEIAR.

SDM error injection is enabled independently for host and microengine accesses by the eTPU Module Configuration Register (eTPU_SCR_MCR) bits HDEIE and MDEIE, respectively. The user must program ETPUDEIAR, ETPUDEIDPR and ETPUDEIPPR with HDEIE=MDEIE=0, and only then enable error injection by setting HDEIE=1 and/or MDEIE=1 to avoid races (see ETPUMESR register).

Similarly, SCM errors can be injected into the SCM read data path through the eTPU Code Error Injection Address Register (eTPU_MESR_CEIAR), eTPU Code Error Injection Data Pattern Register (eTPU_MESR_CEIDPR) and eTPU Code Error Injection Parity Pattern Register (eTPU_MESR_CEIPPR) registers, enabled by the eTPU Module Configuration Register (eTPU_SCR_MCR) bit CEIE. Errors are injected on host accesses to the address specified in ETPUCEIAR when ETPUMCR bit VIS=1, or on microengine and MISC accesses when VIS=0.

43.14.12.2.9 Error Fixing

Single bit **soft errors** can be fixed by rewriting the right value, as corrected by the ECC logic, into the same address of the erroneous data. eTPU provides means to perform such fixes safely, coherently and concurrently with the normal operation.

SDM fixes can be done using the CDC (see Coherent Dual-parameter Controller - CDC), limited to the lower 8 Kbytes of SDM. The Host application simply has to program a transfer with both origin and destination positions pointing to the address of the data to be fixed. Because the PBBASE field has an address granularity of 8 bytes, a word adjacent to the one to be fixed must also be rewritten. The CDC transfer is atomic, therefore the fix is done coherently and does not interfere with the application code or data flows.

SCM fixes can be done by programming the eTPU Code Error Fix Register (eTPU_MESR_CEFR) (see the eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR) register). The fixes are performed by the MISC logic, which must be enabled (see SCM Test - Multiple Input Signature Calculator). The modes available, selected by the ETPUCEFR field CFIXM, are:

- **Automatic Fix Mode (CFIXM=11)**: the MISC logic automatically rewrites the contents of an address whenever it finds a correctable error; the CFIXM field keeps its value if a fix is made.

43.14.13 Performance Monitoring Features

43.14.13.1 Idle Counter

The Idle Counter Registers ETPUIDLER eTPU Idle Register for Engine 1 (eTPU_ER_IDLER_ENG1) and eTPU Idle Register for Engine 2 (eTPU_ER_IDLER_ENG2) continuously counts microcycles in which the microengine is not busy with channel service. It can be used to measure the microengine utilization by rating the count measured during a period of time to the number of microcycles contained in the period. The Idle counter does not count microcycles when the engine is stopped, or is in TST or halt states.

43.14.13.2 Latency Monitor

eTPU implements six Latency Monitors to measure the time taken to service a request. Each Latency Monitor provides:

- The measure of the greatest latency found during actual thread executions, accessed from the eTPU Latency Monitor Measure Register for Engine 1 (eTPU_LMR_LM*n*MR_ENG1) and eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM*n*MR_ENG2) registers.
- A comparison between the measured latency and a threshold, determined through the eTPU Latency Monitor Measure Register for Engine 2 (eTPU_LMR_LM*n*MR_ENG2) and eTPU Latency Monitor Configuration Register for Engine 2 (eTPU_LMR_LM*n*CR_ENG2) registers.

Each Latency Monitor works independently in one of the following modes, selected with the ETPULMnCR field LMM:

- **Disabled mode**: no latency measures or actions are taken.
- **Channel mode**: monitors the latency of an individual channel.
- **Priority mode**: monitors the combined latency of all channels in a given priority level - High, Medium or Low (see Scheduler).

Channel and Priority modes are detailed in the subsections below. The following applies to both modes :

- after each latency measurement, the value measured is compared with the MAXLAT value in the register ETPULMnMR; and if MAXLAT is not zero and the new measured value is greater.
- after each latency measurement, the value measured is compared with the MAXLAT value in the register ETPULMnCR, and if the new measured value is greater:
  - a Global Exception is issued.
  - the ETPUMCR flag LTO1 (engine 1) or LTO2 (engine 2) is set, independently of the LATENCY value in ETPULMnMR.

NOTE

The LATENCY value is updated even if MAXLAT=0.

NOTE

MAXLAT=0 just inhibits the assertion of the LTO1/2 flag and Global Exception due to the respective Latency Monitor. Other Latency Monitor instances with non-zero MAXLAT value may assert LTO1/2 and Global Exception.

43.14.13.2.1 Channel Mode

When the Latency Monitor operates in Channel mode:

- only the channel selected by the ETPULMnCR field MONCHAN is monitored,
- any combination of Host Service Requests (HSR), Channel Service Requests and Link Service Requests can be selected for monitoring, based on the ETPULMnCR field SRSEL,
- the latency is counted in microcycles, from the moment the logic OR of all the service requests selected by SRSEL asserts to the moment this logic OR negates.
- the latency count is interrupted when the channel is disabled, either by writing ETPUCnCR field CPR = 0, or by a watchdog timeout. Latency is counted until the channel is disabled and is not discarded; a new count can start from zero when the channel is re-enabled.

NOTE

If a selected service request type or a combination of them continues asserted during and after its service thread execution, its latency counter will keep on running; this can happen, for instance, if either matches can service requests (e.g. channel modes em_st/em_dt), one match happens before the other is serviced, and only one of them is cleared. The Latency Monitor can, in cases like this, yield pessimistic measurements.

NOTE

The Latency Monitor in channel mode must be disabled before the channel it monitors is disabled.

43.14.13.2.2 Priority Mode

When the Latency Monitor operates in Priority mode:

- only the combination of channels assigned with the priority selected by the ETPULMnCR fields SRSEL is monitored.
- the latency is measured in microcycles, starting from either of these two events, whichever happens first:
  - the logic OR of service requests of the selected priority transitions from 0 to 1 at a moment when all their grants are at state negated.
  - the grants of the selected priority are cleared by the scheduler at the end of the thread, and any of the service requests at the same priority is still active soon after the thread ends.
- the latency is measured until the grants of the selected priority are cleared by the scheduler, after the measurement started.

Note

A Latency Monitor in priority mode must be disabled before any of the channels in the selected priority is disabled, or any disabled channel is assigned with the selected priority.

43.14.13.3 Watchdog Monitor Operation

The watchdog mechanism can also be used to monitor, without interfering with the application execution, whether the execution time of any individual thread or a sequence of threads reaches a given limit. This can be achieved by asserting the eTPU Watchdog Timer Register for Engine 1 (eTPU_ER_WDTR_ENG1) and eTPU Watchdog Timer Register for Engine 2 (eTPU_ER_WDTR_ENG2) bits WMO.

43.15 Initialization/Application Information

43.15.1 Configuration Sequence

After initial power-on reset the eTPU remains in an idle state[37, 37], requiring initialization of several registers before any function can begin execution. Also, if the SCM is implemented in RAM, it should be initialized with the eTPU application code prior to configuring the eTPU. Configuration procedures are summarized as follows:

- If SCM is implemented as RAM, load the eTPU application code (see SCM access).
- Initialize SDM (optional if no ECC support available)
- Initialize the SCM MISC logic (optional, see SCM Test - Multiple Input Signature Calculator).
- Initialize Watchdog and Latency Monitors (optional)
- Initialize the eTPU time base configuration registers eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1) and eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2) to setup:
  - TCR1 and TCR2 prescalers and clock sources.
  - Select digital filtering mode.
  - TCRCLK signal filter control.
  - Angle mode operation (if necessary).
- Initialize the eTPU engine configuration registers eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2) to setup:
  - Entry table base.
  - Filter prescaler clock control.
- Initialize eTPU STAC configuration registers eTPU Engine 1 STAC Configuration Register (eTPU_TBR_STAC_ENG1) and eTPU Engine 2 STAC Configuration Register (eTPU_TBR_STAC_ENG2) if one needs to setup TCR1/2 resource Client/Server operation.
- Write to the Channel Configuration registers eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2) to choose the Function to be performed by each channel, and its parameter base address.

37. except when device debug request is asserted on power-on reset: in this case, the microengines wake-up in halt state.

- Write to channel status control registers eTPU Channel n Status Control Register for Engine 1 (eTPU_CCCR_C*n*SCR_ENG1) and eTPU Channel n Status Control Register for Engine 2 (eTPU_CCCR_C*n*SCR_ENG2) to choose among the possible variations within the function flow (FM bits).
- Write to SDM for parameter initialization of each configured channel.
- Write to registers eTPU Watchdog Timer Register for Engine 1 (eTPU_ER_WDTR_ENG1) and eTPU Watchdog Timer Register for Engine 2 (eTPU_ER_WDTR_ENG2) if one needs to enable and setup the Watchdog(s) mode and timeout.
- Write to channel n Host Service Request registers eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C*n*HSRR_ENG1) and eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C*n*HSRR_ENG2) to initialize the active channels.
- Write to the channel interrupt enable register eTPU Channel Interrupt Enable Register for Engine 1 (eTPU_GCR_CIER_ENG1) and eTPU Channel Interrupt Enable Register for Engine 2 (eTPU_GCR_CIER_ENG2) if interrupts are to be enabled from the appropriate channels. Likewise for Data Transfer Requests (ETPUCDTRER). This can also be done through ETPUCxCR.
- Write to channel n configuration registers eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2) to enable each channel by assigning it a high, middle, or low priority (CPR field).[37]
- Monitor the Host service request registers (ETPUCxHSRR) eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C*n*HSRR_ENG1) and eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C*n*HSRR_ENG2) for completion of initialization.
- Write ETPUMCR eTPU Module Configuration Register (eTPU_SCR_MCR) bit GTBE=1 to start TCR1/TCR2 time base counting at same time in both Engines (may be done before or never, depending on the particular application and use of Red Line bus).

43.15.2 Reset Options

43.15.2.1 Hardware Reset

Hardware reset is achieved by assertion of device synchronous reset . Both Engines and common logic is reset, and even the System Configuration and Global Channel registers assume their reset values.

Note

All eTPU input clocks must pulse during reset so that both engines are reset, even if they are in Module Disable or Stop mode.

43.15.2.2 Software Reset

eTPU has no Software reset. To abort infinite microcode loops, the Force END mechanism must be used (see field FEND in eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2)). .

43.15.3 Multiple Parameter Coherency Methods

Follows a description of two methods for coherent transfer of multiple parameters between Host and eTPU. Both methods involve the use of two parameter areas: the Transfer Parameter Area (hereafter called TPA), which is the SDM area directly accessed by the Host for reads and writes, and the Permanent Parameter Area (hereafter called PPA), which are the SDM positions where channel parameters are normally accessed by the Function microcode. Note that parameters in either TPA or PPA do not have to be in sequential addresses. TPAs and PPAs allocation are completely defined by the application, and there may be any number of them, independently of the channels.

The methods described here are not the only solutions for the coherent transfer problem, and both can coexist in eTPU and even used in combination. Also note that for transfers of a pair of parameters, the Coherent Dual-parameter Controller is faster and have less impact on both eTPU and Host performance. That said, the methods are:

- **Transfer Service**: a microengine thread transfers, upon Host Service Request, data from/to a TPA to/from a PPA.Coherency is guaranteed by the fact that a thread is atomic with respect to other threads in the same Engine, and so are its transfers. If parameters in PPA are shared by both Engines, hardware semaphores have to be used to access them.
- **Mailbox**: for Host to eTPU transfers, the microcode checks a flag, set by the host, indicating the existence of new parameter data in the TPA. It can, then, either access TPA data directly or copy it to the PPA. For eTPU to Host transfers, when microcode changes PPA, it copies them to the TPA and flags updated TPA data to Host, possibly using an Interrupt or a Data Transfer Request. The Mailbox flag is reset when data is copied: by the eTPU microcode, when it transfers TPA to PPA (possibly followed by an Interrupt); by the Host, when it reads data from the TPA. This indicates that TPA is free for another transfer.

Transfer Service has the advantage of separating the task of data transfer from the functional service thread that accesses the parameters, with less impact to the latter. Compared to the Mailbox method, however, it has bigger average latency, because the Transfer Service thread has to contend for a time slot to execute. This latency can be minimized if Transfer Service thread is assigned to a separate channel with higher priority, but even so it does not guarantee that PPA is updated before the next execution of the functional thread that uses it.

Mailbox method, on the other hand, makes the functional thread check for the existence of new data (Host to eTPU). It does not have to be responsible for the transfer, though: it may access the TPA directly, and a Transfer Service can then be used to copy data from TPA to PPA.

43.15.4 Programming Hints and Caveats

43.15.4.1 Atomic dual access after a call, return

A dual, back-to-back parameter access is not atomic after a call, a jump, or a return if they occurred in parallel with an odd SDM access. It is safer to make a pair of parameter accesses that must be coherent begin at the second instruction after a call/jump/return.

43.15.4.2 Resource polling

The use of polling while waiting for a condition or a resource (except semaphore lock) should be avoided in order not to hang the microengine in long loops. This general programming guideline is greatly enforced in eTPU, as a thread cannot be preempted for any reason. Safer polling, albeit with long and indeterministic latency, can be obtained if a thread issues a channel link to the serviced channel and terminates. The microengine is then free to other tasks, and the next poll happens at the next time the channel is serviced. This mechanism can be combined with finite (timed out) loops for better latency.

43.15.4.3 Changing Channel Function, Parameter Base, or Entry Table Scheme

Channel Function, Parameter Base Address and Entry Table Scheme are determined by the eTPU Channel n Configuration Register for Engine 1 (eTPU_CCCR_C*n*CR_ENG1) and eTPU Channel n Configuration Register for Engine 2 (eTPU_CCCR_C*n*CR_ENG2) fields CFS, CPBA and ETCS. They cannot be changed when the channel is enabled. If the channel is disabled first, one may still have service requests from the previous function, so before the channel is enabled again one must be sure that:

- the first thread executed in the new function is the initialization one.
- the initialization thread of the new function clears any previously pending service request.

Follows a safe procedure for function changing:

1. disable the channel (write ETPUCnCR field CPR=00).
2. change the function configuration (ETPUCnCR fields CFS and/or CPBA and/or ETCS).
3. request the initialization thread, writing eTPU Channel n Host Service Request Register for Engine 1 (eTPU_CCCR_C*n*HSRR_ENG1) and eTPU Channel n Host Service Request Register for Engine 2 (eTPU_CCCR_C*n*HSRR_ENG2) with the initialization HSR (channel still disabled).
4. enable the channel (write ETPUCnCR field CPR > 0); the initialization HSR is serviced before any other formerly pending service requests, clearing them.

43.15.4.4 Checking and Clearing Interrupts of a Stopped Engine

An Engine may be stopped with interrupts (or DMA requests) pending. This includes the case when the Engine's MDIS bit is set and a thread is still running: the thread will complete execution, possibly issuing an interrupt or DMA request before the engine stops, setting the STF bit.

As soon as the engine stops the channel registers become inaccessible, issuing bus errors when accessed. Interrupts and DMA requests can still be checked and cleared through the Global Channel Registers, though. DMA requests can also be cleared by the hardware handshaking with the DMA controller when the engine is stopped.

43.15.5 Estimating Worst Case Latency

Reliable systems are designed to work under worst-case conditions. This section explains how to estimate worst-case latency (WCL) for any eTPU function in any system. The appendix covers the following topics:

- Introduction to Worst-Case Latency
- Using Worst-Case Latency Estimates to Evaluate Performance
- Priority Scheme Details used in WCL Analyses
- First-Pass WCL Analysis
- Second-Pass WCL Analysis

The first-pass WCL analysis is based on a deterministic, generalized formula that is easy to apply. Because of the generalizations in the formula, the first analysis result is almost always much worse than the real worst case. If the desired system performance is within the limits of this first analysis, then no further analysis is required; the system is well within the performance limits of the eTPU. If the desired system performance exceeds that indicated by the first analysis, the second-pass WCL analysis should be applied. The second-pass analysis is not a generalized formula, but rather uses specific system details for a realistic worst-case estimation.

43.15.5.1 Using Worst-Case Latency Estimates to Evaluate Performance

Once WCL is found for a channel, the user must determine how to use this number to analyze performance. To analyze the performance of a channel running the PWM function, for example, some information about what happens in each thread is necessary.

The following example refers to old TPU PWM function, which is not optimized to the eTPU enhanced hardware. For PWM, thread 1 is the initialization thread, and threads 2 and 3 are used during normal function execution. (PWM threads 4, 5, and 6 are for special modes and will be assumed to be unused on channel 5). Thread 2 writes a time into the channel 5 match register and performs other operations that will cause the channel 5 signal to go from low to high at the time indicated in the match register (match time). At match time, the signal goes high and channel 5 requests service from the eTPU Microengine to execute thread 3. Thread 3 writes a time into the channel 5 match register and performs other operations that will cause the channel 5 signal to go from high to low at match time. At match time, the signal goes low and channel 5 requests service from the eTPU Microengine to execute thread 2. A PWM wave is kept running on the system by the eTPU executing thread 2, then thread 3, then thread 2, then thread 3, and so on.

Since the definition of worse-case latency assumes a fully loaded running system, initialization threads are not part of worst-case calculations. For the channel 5 example, the two PWM threads in are thus the two normal running threads, threads 2 and 3.

does not define which thread is thread 2 and which is thread 3. Since the worst-case latency derived from the first-pass analysis is the worst case between any2 threads (not counting initialization threads), it is safe to say that the worst-case latency shown in represents both the worst-case high time and the worst-case low time.

Notice in that worst-case latency is drawn from the end of the execution of the first PWM thread to the end of the execution of the next PWM thread. It is drawn from end to end because the microcode instructions that make up the threads control the channel hardware. To make sure that all the microcode instructions needed to change the pin thread have been executed, it is necessary to include the execution time of the second thread.

Thread information for each function is found in the programming notes for individual TPU functions. Refer to Freescale Programming Note TPUPN00/D, Using the TPU Function Library and TPU Emulation Mode, for a list of available programming notes. Similar documentation will we provided for the eTPU new functions.

43.15.5.2 Priority Scheme Details Used in WCL Analysis

The user assigns functions to channel numbers and gives each active channel a priority level of high, middle, or low. The Scheduler uses the channel number and channel priority level to determine the order in which to grant service.

The scheduler allocates time slots to specific priority levels of high, middle, or low. One function thread is executed in each time slot. The length of a time slot varies according to the length of the executing thread. When fully loaded, the scheduler always assigns time

slots in a seven-slot sequence (see the following table). After a seven-slot sequence is completed, another seven-slot sequence begins (see Figure 43-335). Note that in eTPU, when no service request exists, the scheduler goes to thread 1, but WCL calculation considers full load.

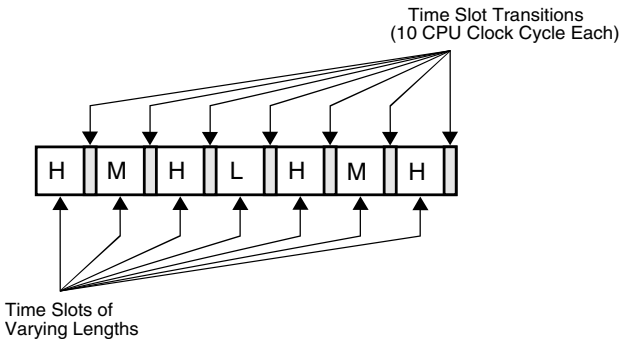


Figure 43-334. Time-Slot Sequence

This sequence scheme gives higher-priority channels more service time than lower-priority channels. High-priority channels are allocated four of seven time slots, middle-priority channels are allocated two of seven time slots, and low-priority channels are allocated one of seven time slots.

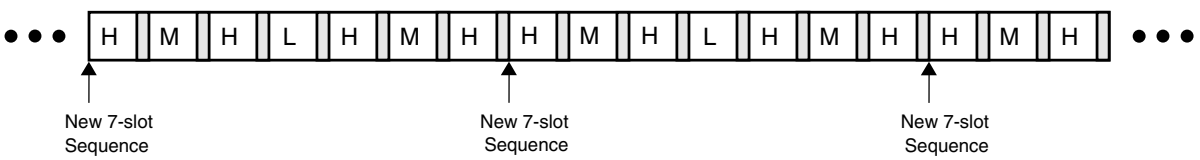


Figure 43-335. Multiple Time-Slot Sequences

43.15.5.2.1 Priority Passing

If no channel of the priority level assigned to the time slot is requesting service, the eTPU scheduler can pass priority to other levels. If no high-level channel is requesting service during a high level time slot, a middle-level channel is granted service; or, if no middle level-channel is requesting service, a low-level channel is granted service. If no middle-level channel is requesting service during a middle-level time slot, a high-level channel is granted service; or, if no high-level channel is requesting service, a low-level channel is granted service. If no low-level channel is requesting service during a low-level time slot, a high-level channel is granted service; or, if no high-level channel is requesting service, a middle-level channel is granted service. If no channel is requesting service, the time slot sequence is reset to state 1 and the scheduler idles until a request is received.

Priority passing is implemented in hardware and does not contribute to worst-case latency.

43.15.5.2.2 Time-Slot Transition

After each time slot, the eTPU must prepare for the next time slot. This preparation time between each time slot is called a time-slot transition. See Time Slot Transition . Time-slot transitions can take from six up to ten eTPU clocks.

43.15.5.2.3 Channel Number Priority

If more than one channel of a priority level is requesting service, the lowest numbered channel is granted service first. For example, if channels 0, 5, and 9 are all high-level channels requesting service during a high time slot, channel 0 is granted service first. Continuing this example, if channel 0 requests service again immediately after being serviced, it is not serviced again until channels 5 and 9 are serviced. This scheme is implemented so that continuously-requesting low numbered channels do not take all the time on the eTPU execution unit and leave no time for other channels.

The scheduler uses registers to keep track of which channels have been serviced and which require servicing. Each channel has two register bit: a service request register (SRR) and a service grant register (SGR). The SRR is set when a channel requests service. After the channel has been granted service, the SGR is set and the SRR is cleared.

SGRs are not cleared individually by channel, but rather as priority level groups. The clearing of a group of SGRs begins a new cycle for that priority level. An SGR group is cleared on the condition that a channel of that priority level has just been serviced, and no other channel of that priority level is requesting service (has a set SRR) and has not been granted service (has a clear SGR).

For example, if a middle-priority channel has just been serviced (either in a middle-priority time slot or a high or low-priority time slot gained by priority passing), the SRRs and SGRs of all middle-priority channels are compared. If there is no middle-priority channel with its SRR set and SGR cleared, the scheduler clears all middle-level SGRs. If there is a middle-level channel with its SRR set and SGR cleared, the scheduler does not clear the SGR group, and the requesting middle-level channel is serviced on the next middle-level time slot (or possibly sooner by priority passing).

43.15.5.2.4 SDM Collision Rate

Most function threads read or write to the eTPU SDM at least once. Because both the eTPU Microengine and Host can access the SDM but not at the same time, the Microengine may suspend execution during the SDM access while waiting for the Host to finish accessing the SDM. At other times the Host may wait for the Microengine. Wait states can take up to two eTPU clocks, when the Host accesses the SDM directly, without

using CDC. Microengine(s) wait-states must be added into the worst-case latency calculation. The system designer should estimate the percentage of SDM accesses in the system that will result in Microengine wait-states. This percentage is called the RAM collision rate (RCR). In each collision with direct Host accesses to the SDM the Microengine(s) wait for two eTPU clocks.

In eTPU the Coherent Dual-parameter Controller (CDC) may also access the SDM for atomic transfers of two parameters. eTPU Microengine may wait on this operation (if it is in service time) until the transfer is complete. CDC always transfers two parameters, making four consecutive accesses (read, write, read, write) of one eTPU clock each. The system designer should estimate the percentage of SDM accesses in the system that will result in a Microengine wait due to coherent transfer, and multiply it with the average number of eTPU clocks the Microengine waits for each transfer. This percentage is called Coherent Parameter Collision Rate (CPCR).

In addition, Microengine to Microengine multiple parameter coherent communication, using the hardware semaphores, may hold one Microengine which waits to lock the semaphore while the other Microengine is holding it. This waiting is due to a software loop, not hardware wait-states. Note that single parameter access of one Microengine does not affect the timing of the other Microengine due to SDM time interlace. This implies that single parameter Microengine to Microengine communication does not affect the performance. The Microengine which waits for the semaphore will loop until it is freed by the other Microengine. This time depends on the eTPU application. The system designer should estimate the percentage of Microengine to Microengine coherent parameter communication that will result in eTPU semaphore loops, and multiply it with the average number of eTPU clocks the Microengine loops for each such transfer. This percentage is called CCR (Communication Collision Rate).

A 100% collision rate for a system is the theoretical worst case. In many systems, however, the RCR, CPCR and CCR would be very low, sometimes even near 0%. This is because the eTPU is an independent processor capable of servicing most function needs, so that the Host rarely needs to access the eTPU SDM. Also coherent Microengine to Microengine communication of more than one parameter may be rare. To find a realistic RCR, CPCR the system designer should evaluate the Host code and find the percentage of time it accesses the eTPU SDM with or without using the CDC. This percentage gives a good RCR and CPCR. The eTPU application provides a good estimation of CCR.

Note

The programming practice of polling a flag in the eTPU SDM causes a very high RCR and should be avoided in high-performance systems.

After the collision rate for a system is found, it can be applied to the WCL calculations for each channel. The system designer can use the collision percentage and the number of SDM accesses (with and without semaphores) to estimate the eTPU loop time for a function. Note that in old TPU functions CPCR and CCR are both zero.

The estimation of eTPU wait time is as follows:

Variables:

N1 = Number of simple RAM accesses in the longest thread

RCRWait = Maximal eTPU clocks wait time for simple RAM collision = 2

CPCRWait = Average eTPU clocks for Coherent Parameter Transfer (using CDC).

N2 = Number of eTPU-eTPU semaphore RAM accesses in the longest thread

CCRWait = Average eTPU clocks for Microengine-Microengine communication transfer.

Estimated Wait Time:

Function eTPU maximal wait time =

N1 *(RCR * RCRWait + CPCR * CPCRWait) + N2* CCR * CCRWait

43.15.5.3 First-Pass Worst-Case Latency Analysis

Following is the first-pass calculation of worst-case latency for a channel. Remember that this analysis uses generalizations that usually produce a result much worse than the real worst case. If the worst-case result from the first analysis is too long for the desired performance, use the second analysis for a more realistic worst-case analysis.

43.15.5.3.1 Worst-Case Assumptions and Formula

To estimate worst-case latency for a channel, assume this worst-case condition: the channel has just been serviced in a time slot of its priority level, and all other channels in the system are continuously requesting service and have cleared SGRs. The worst-case latency is the time from the end of the channel's service until the end of the channel's next service. See the following figure.

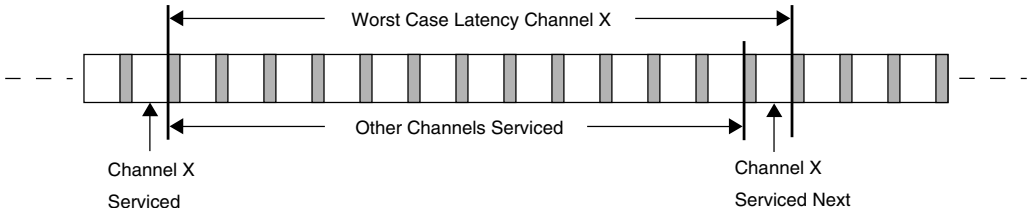


Figure 43-336. First-Pass Worst-Case Latency

To estimate worst-case latency:

- Find the worst-case service time for each active channel.
- Using the H-M-H-L-H-M-H time-slot sequence, map the channels that are granted for each time slot.
- Add time for six-clock time-slot transitions.

43.15.5.3.1.1 Mapping the Channels for Each Time Slot

To determine when a channel will be serviced again, it is necessary to determine which other channels will be serviced first. Do this by assuming all channels are continuously requesting service and mapping the channels into the time-slot sequence.

43.15.5.3.1.2 Adding Time for Time-Slot Transitions

Add six eTPU clocks for time-slot transitions which occur after each time slot.

43.15.5.3.2 First-Pass Analysis Worst-Case Latency Examples

The examples in this section assume the system configuration shown in the following table.

Table 43-419. System Configuration Example

Channel	Priority	Function[1, 2]
0	High	PWM (driving a DC motor)
1	Middle	PPWA (Mode 0, measuring the DC motor speed)
2	Low	DIO (Input)

1. 9% RAM Collision Rate (RCR)
2. CPU clock rate = 40 MHz, or 25 ns per clock period

43.15.5.3.2.1 Finding the WCL for PWM on Channel 0

The following shows how to find the WCL for PWM on channel 0.

1. Find the worst-case service time for each active channel.

   a. Longest thread of PWM is 24 CPU clocks with four RAM accesses.

      24 + ((4 RAM accesses+1) * 0.09 * 2 CPU clock waits) = 24.9 CPU clocks, rounded up to 25 CPU clocks (since there are no partial clock periods)

      Channel 0 worst-case service time = 25 CPU clocks.

   b. Longest thread of PPWA in mode 0 is 44 CPU clocks with nine RAM accesses.

      44 + ((9 RAM accesses+1) * 0.09 * 2 CPU clock waits) = 45.8 CPU clocks, rounded up to 46 CPU clocks

      Channel 1 worst-case service time = 46 CPU clocks.

   c. Longest thread of DIO is ten CPU clocks with four RAM accesses.

      10 + ((4 RAM accesses+1) * 0.09 * 2 CPU clock waits) = 10.9 CPU clocks, rounded up to 11 CPU clocks

      Channel 2 worst-case service time = 11 CPU clocks.

2. Assume channel 0 has just been serviced and that channels 1 and 2 are continuously requesting service. Using the H-M-H-L-H-M-H time-slot sequence, map the channels that are granted for each time slot. See the following figure.

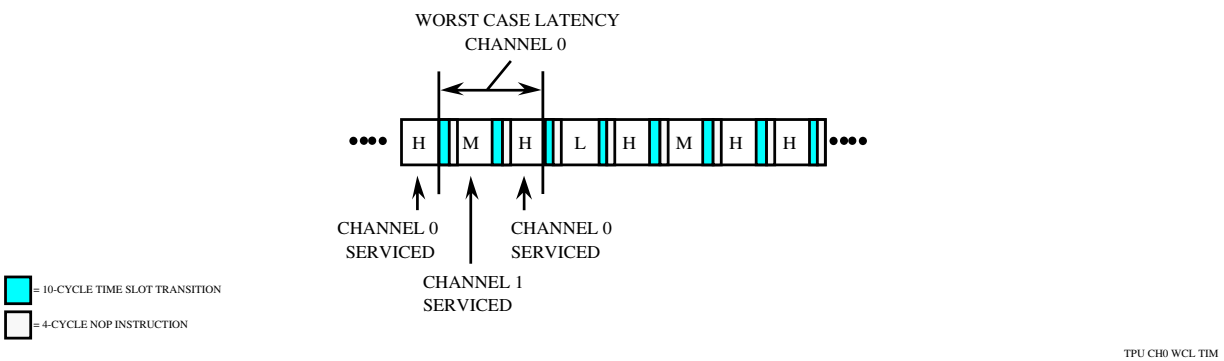


Figure 43-337. Next Servicing for Channel 0

   Channel 1 will be serviced in the middle-priority time slot before channel 0 is serviced again.

3. Add time for the six-clock CPU time-slot transitions. See the above figure and the following table.

   A four-clock NOP occurs after each channel is serviced since there is one channel in each priority level, i.e., a new cycle for a priority level is started after each channel is serviced. Time-slot transitions occur after each time slot.

Table 43-420. Worst-Case Latency for Channel 0

Channel 0 worst-case service time	25 clocks
Channel 1 worst-case service time	46 clocks
Two 6-clock time-slot transitions	12 clocks
Total clocks	83 clocks

83 clocks * 25 ns/clock = 2075 ns

Conclusion: in this system configuration PWM can run with a minimum high time or low time of 2075 ns.

Note that in double match eTPU system the PWM can be serviced once in each period, and there is no latency for minimum high time. The latency in eTPU PWM function will represent the minimum PWM period.

43.15.5.3.2.2 Finding the WCL for PPWA on Channel 1

The following shows how to find the WCL for PPWA on channel 1.

1. Find the worst-case service time for each active channel. See step 1 of previous example.
2. Assume channel 1 has just been serviced and that channels 0 and 2 are continuously requesting service. Using the H-M-H-L-H-M-H time-slot sequence, map the channels that are granted for each time slot. See the following figure.

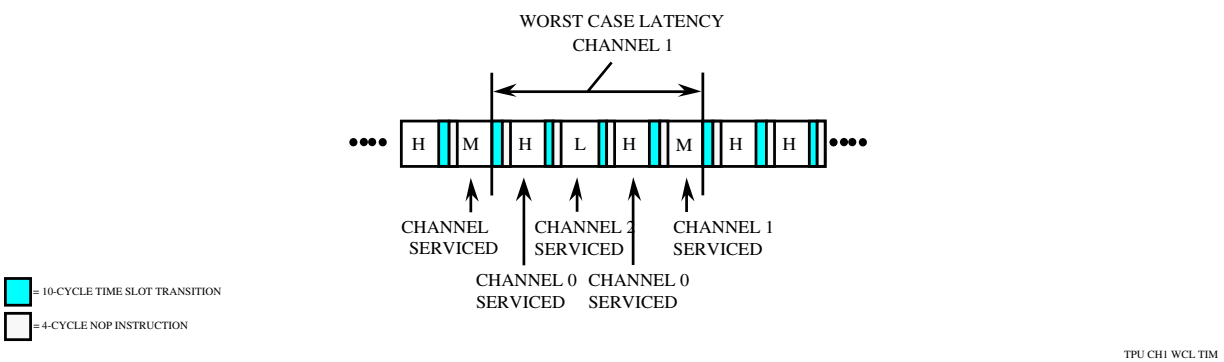


Figure 43-338. Next Servicing for Channel 1

Channel 0 will be serviced twice and channel 2 once before channel 1 is serviced again.

3. Add time for the six-clock CPU time-slot transitions. See the above figure and the following table.

Table 43-421. Worst Case Latency for Channel 1

Two Channel 0 worst-case service times	50 clocks
Channel 1 worst-case service time	46 clocks
Channel 2 worst-case service time	11 clocks
Four 6-clock time-slot transitions	24 clocks
Total clocks	131 clocks

131 clocks * 25 ns/clock = 3275 ns

Conclusion: in this system configuration PPWA can measure a period or pulse of minimum 3275 ns.

Note that PPWA function optimized for eTPU hardware can use double transition mode to measure very narrow pulses with one service after the second transition, and latency will affect only the minimum gap between two input pulses. Also the function threads would have more efficient coding.

43.15.5.3.2.3 Finding the WCL for DIO on Channel 2

The following shows how to find the WCL for DIO on channel 2.

1. Find the worst-case service time for each active channel. See step 1 of previous examples.
2. Assume channel 2 has just been serviced and that channels 0 and 1 are continuously requesting service. Using the H-M-H-L-H-M-H time-slot sequence, map the channels that are granted for each time slot. See the following figure.

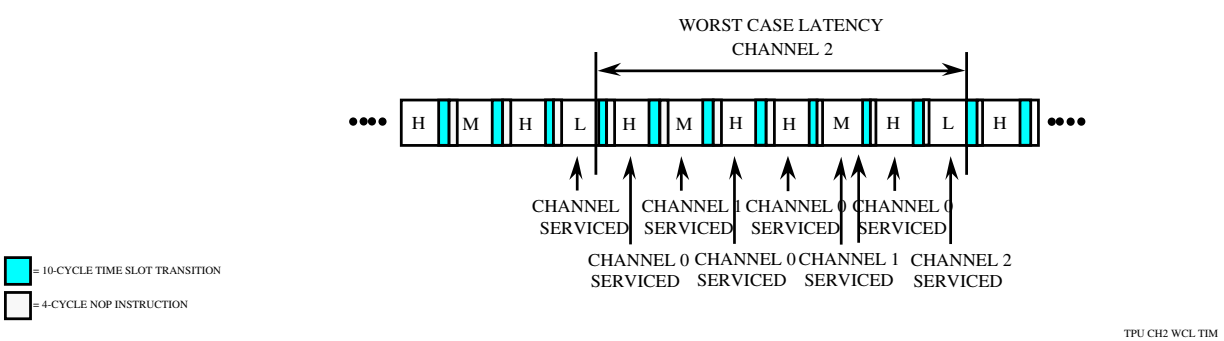


Figure 43-339. Next Servicing for Channel 2

Channel 0 will be serviced four times and channel 1 twice before channel 2 is serviced again.

3. Add time for the ten-clock CPU time-slot transitions and the four-clock NOPs. See the above figure and the following figure.

Table 43-422. Worst Case Latency for Channel 2

Four Channel 0 worst-case service times	100 clocks
Two Channel 1 worst-case service time	92 clocks
Channel 2 worst-case service time	11 clocks
Seven 6-clock time-slot transitions	42 clocks
Total clocks	245 clocks

245 clocks * 25 ns/clock = 6125 ns

Conclusion: in this system configuration DIO can keep track of the input level at a minimum of every 6125 ns.

Note that DIO function optimized for eTPU hardware can use double transition mode to measure two pin transitions at a time and reduce the service time, improving the overall system performance and latency.

43.15.5.4 Second-Pass Worst-Case Latency Analysis

Following is an example of a second-pass analysis for calculating worst-case latency for a channel. The second-pass analysis is useful for higher-performance systems, since it gives a more realistic worst-case latency result than first-pass analysis.

This example uses a relatively simple system in order to illustrate the basic principles of second-pass analysis. For a more complex example of second-pass analysis, refer to Multiphase Motor Commutation TPU Function (COMM)(TPUPN09/ D).

43.15.5.4.1 Second-Pass Analysis Guidelines

Rather than use a fixed formula, a second-pass analysis relies on the application of the following guidelines.

1. The first-pass analysis makes the assumption that all channels in the system are continually requesting service. For many systems this is an unrealistic assumption. For example, if TCR1 is counting at a rate of 2 MHz (500 ns per count) and a

channel is running the DIO function with a match rate of 20,000 TCR1 counts, the DIO will request service every 10 ms (20,000 * 500 ns = 10,000,000 ns or 10 ms). It is therefore unrealistic to assume that the channel running this DIO function is continuously requesting service. Figure out a realistic service request rate for each channel. Time slots can then be mapped to each channel at the real rate of request.

2. If a function is active during system initialization but not during the high-speed running mode of the system, then that system does not need to be included in the high-speed worst-case latency calculations.
3. Use a realistic SDM collision rate.
4. Be careful when assigning functions priority levels and channel numbers. Decide which function or functions will be most difficult to make perform at the desired level. Assign those channels high priority and low channel numbers. Try different priority and channel assignments to see how it affects the system.
5. The seven-slot sequence of || H | M | H | L | H | M | H || is asymmetrical when put back-to-back with other seven-slot sequences. Note that in the following sequence there are two high-priority slots next to each other:

   || H | M | H | L | H | M | H |||| H | M | H | L | H | M | H ||
6. Make sure that when mapping out channels to the sequence, you choose a worst-case slot to start the mapping. For example, when estimating WCL for a high-priority channel, do not start the mapping in the last high-priority slot in a seven-slot sequence, as that is a best case for a high-priority channel since another high-priority time slot is next.
7. Instead of always using the longest thread in the function as the worst-case thread, evaluate the threads in the function that will be used in the system and use the appropriate worst-case threads. For example, in the preceding example of first-pass analysis, the PWM was shown to be able to achieve a high time and low time of 2475 ns under worst-case conditions. This was derived using the longest PWM thread of 24 CPU clocks. This longest thread is actually thread 2, the thread that is entered after the pin has just gone high. Thread 3, the thread that is entered after the pin has just gone low, requires only 2 CPU clocks. Therefore, in the first-pass example, the high time was correctly derived, but the low time is actually shorter than was estimated.

43.15.5.4.2 Second-Pass Analysis Example

This example requires three 50% PWM waveforms: one 5 kHz (200 ms/period) and two 50 kHz (20 ms/period), each running DC motors. (Remember that the PWM function requests service from the eTPU after each high time and after each low time, so the eTPU must handle a request every 100 ms for the 5 kHz PWM and every 10 ms for the 50 MHz PWM.)

Note

This example uses square waves for simplicity. Notice that to use a PWM waveform in the typical way, in which the pulse is modulated, the pulse must not be modulated in a way that violates the worst-case latency requirements.

This example also uses one DIO channel monitoring a signal level every millisecond and one PPWA channel in mode 0 monitoring the speed of the 5-kHz DC motor. The PPWA must measure periods of 5 kHz (200 ms/period).

The CPU is interrupted by the channel running the PPWA function after measuring 200 periods (every 40 ms). The interrupt service routine performs an averaging of the period accumulation and checks it against a known parameter. The interrupt service time is so short and infrequent that it is a tiny fraction of total system time. The interrupt service routine contains no polling of the SDM. Therefore a realistic RCR = 0%.

43.15.5.4.2.1 First-Try System Configuration

Try a system configuration that seems likely to work. If it does not, change priority levels or channel numbers.

The 5 kHz and 50 kHz PWMs are the most time-critical functions. Those are assigned high priority. PPWA is assigned middle priority. The DIO is low performance and is assigned low priority. Refer to the following table.

Table 43-423. First-Try System Configuration

Channel	Priority	Function[1], [2]
0	High	PWM at 50 kHz (needs a 4-µs WCL)
1	High	PWM at 50 kHz (needs a 4-µs WCL)
2	High	PWM at 5 kHz (needs a 40-µs WCL)
8	Middle	PPWA at 5 kHz (needs a 80-µs WCL)
15	Low	DIO as input at rate of 1 ms

1. 0% RAM collision rate
2. CPU clock rate = 40 MHz, or 60 ns per clock period

With this system configuration, worst-case service time for each active channel is determined as follows:

1. Longest thread of PWM is 24 CPU clocks with four RAM accesses.

   24 + ((4 RAM accesses+1) * 0 * 2 CPU clock waits) = 24 CPU clocks

   Channels 0-2 worst-case service time = 24 CPU clocks.

2. Longest thread of PPWA in mode 0 is 44 CPU clocks with nine RAM accesses.

   44 + ((9 RAM accesses +1)* 0 * 2 CPU clock waits) = 44 CPU clocks

   Channel 8 worst-case service time = 44 CPU clocks.

3. Longest thread of DIO is ten CPU clocks with four RAM accesses.

   10 + ((4 RAM accesses+1) * 0 * 2 CPU clock waits) = 10 CPU clocks

   Channel 15 worst-case service time = 10 CPU clocks.

   To find the WCL for channel 0, assume channel 0 has just finished service.

   Map the channels in the H-M-H-L-H-M-H sequence. See the following figure.

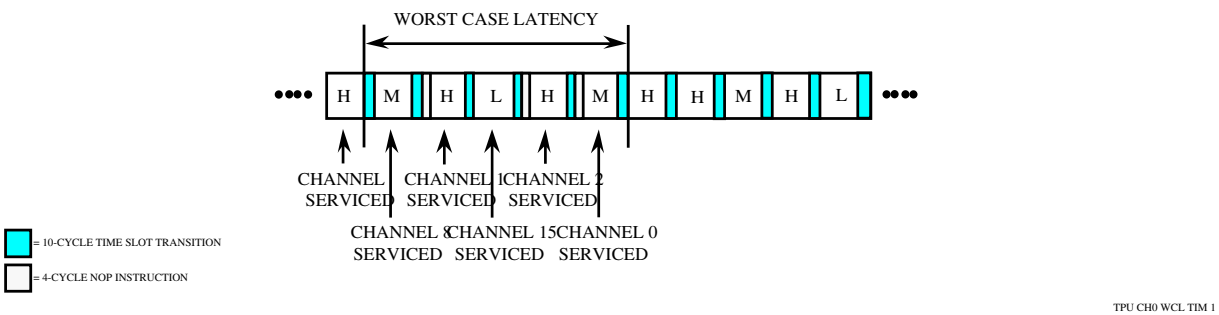


Figure 43-340. Worst-Case Latency for Channel 0 (First Try)

Conclusion: with this system configuration, worst-case latencies for channels 0 and 1 are too high (WCL for channel 1 is the same as WCL for channel 0). Try a different system configuration.

43.15.5.4.2.2 Second-Try System Configuration

The second-try system configuration is shown in the following table.

Table 43-424. Second-Try System Configuration

Channel	Priority	Function[1], [2]
0	High	PWM at 50 kHz (needs a 4-µs WCL)
1	High	PWM at 50 kHz (needs a 4-µs WCL)
2	Middle	PWM at 5 kHz (needs a 40-µs WCL)
8	Middle	PPWA at 5 kHz (needs a 80-µs WCL)
15	Low	DIO as input at rate of 1 ms

1. 0% RAM collision rate
2. CPU clock rate = 40 MHz, or 60 ns per clock period

To find the WCL for channel 0, assume channel 0 has just finished service. Map the channels in the H-M-H-L-H-M-H sequence. See the following figure.

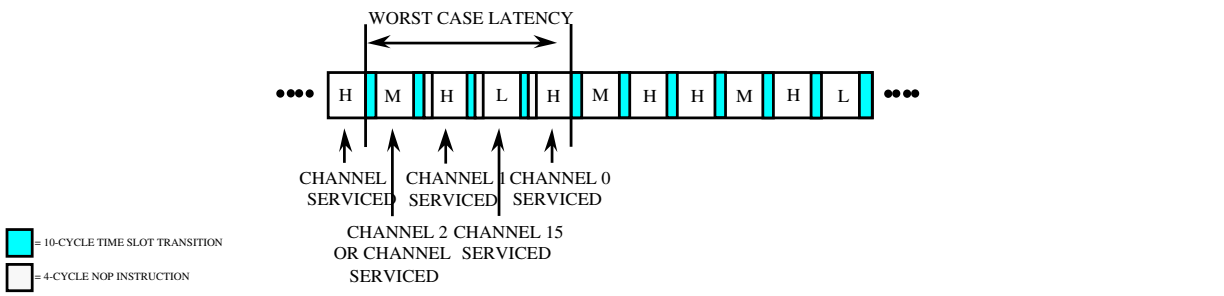


Figure 43-341. Worst-Case Latency for Channel 0 (Second Try)

Conclusion: with this system configuration, the WCL of both channel 0 and channel 1 is 3.85 ms, which is within the limit of 4 ms needed for a 50-kHz PWM.

Next, find the WCL for channel 2. Assume channel 2 has just finished service. Map the channels in the H-M-H-L-H-M-H sequence. See the following figure.

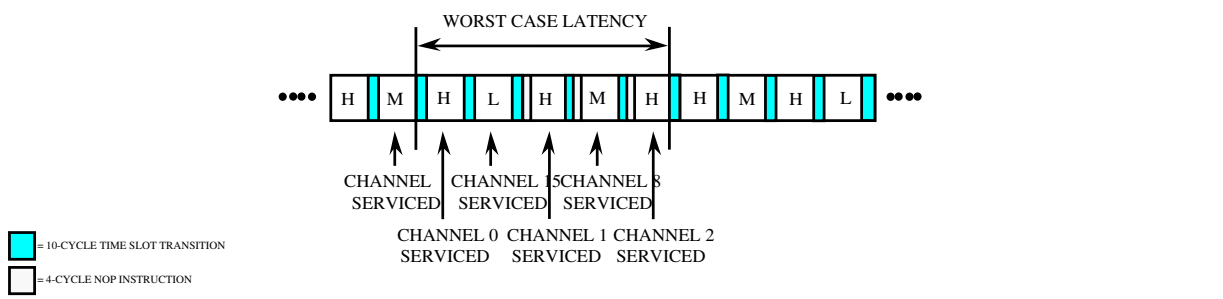


Figure 43-342. Worst-Case Latency for Channel 2

Conclusion: with this system configuration, the WCL for channels 2 and 8 is 4.7 ms, which is within the 40 and 80 ms WCL requirements.

Notice that channels 2 and 8 are well within their WCL requirements. The system could be reconfigured as shown in the following table to give channels 0 and 1 a larger margin while still keeping channels 2, 8 and 15 within their WCL requirements.

Table 43-425. Second-Try System with Channel 0 and 1 Reconfigured

Channel	Priority	Function[1], [2]
0	High	PWM at 50 kHz (needs a 10-µs WCL)
1	High	PWM at 50 kHz (needs a 10-µs WCL)
2	Middle	PWM at 5 kHz (needs a 40-µs WCL)
8	Low	PPWA at 5 kHz (needs a 80-µs WCL)
15	Low	DIO as input at rate of 1 ms

1. 0% RAM collision rate
2. CPU clock rate = 40 MHz, or 60 ns per clock period

43.15.6 Endianness

The following table shows the parameter addresses for big and little endian machines.

Table 43-426. Parameter Addresses and Endianness

Parameter	byte address offset (n = word address offset)	
	big endian	little endian
32-bit	4*n	
24-bit	4*n + 1	4*n
32-bit parameter's most significant byte	4*n	4*n + 3
24-bit parameter's most significant byte	4*n + 1	4*n + 2
least significant byte	4*n + 3	4*n

43.15.7 Memory Error Service Procedures

Follows separate procedures for correctable and non-correctable error interrupt service routines. It is assumed that the non-correctable error ISR has higher priority, and cannot be preempted by the correctable error ISR. For memory error support details, see Memory Error Support.

Note

The memory error support hardware does not (and is not intended to) guarantee that no error occurred will go unreported. However, combined with the procedures below, the information loss is minimized to statistical irrelevance on most scenarios.

43.15.7.1 Non-correctable Error Service Procedure

The basic procedure below can be followed in a Global Exception or a bus error machine check service routine. It handles the service for both SCM and SDM errors, as they are symmetrical at this level. However, a real implementation is likely to split the procedure in two, one for each memory.

1. Confirm/identify the error source, checking assertion of the eTPU Module Configuration Register (eTPU_SCR_MCR) bits SCMERR and SDMERR.
2. Save the state(s) of eTPU Module Configuration Register (eTPU_SCR_MCR) bit(s) CRDIS/DRDIS.
3. Set the eTPU Module Configuration Register (eTPU_SCR_MCR) CRDIS/DRDIS=1 to disable further updates to the respective report registers, guaranteeing coherency.
4. Check whether the report registers were updated by a correctable or non-correctable error; to do this, check the eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR) syndrome (for SCM) and/or the eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR) syndromes (for SDM, only those with the respective BE flags asserted). If it is a correctable error report, skip the steps 5 and 6 for the respective memory.
5. Identify the error source (host, microengine or MISC/CDC) by checking the field ERR_ACC in the register(s) eTPU Code Error Report Address Register (eTPU_MESR_CERAR) and eTPU Data Error Report Address Register (eTPU_MESR_DERAR); take or schedule the appropriate application-dependent recover actions.
6. Update error statistics as needed; it may include recording the access source, address, data and syndrome from the registers ETPUCERAR and ETPUDERAR, eTPU Code Error Report Data Register (eTPU_MESR_CERDR) and eTPU Data Error Report Data Register (eTPU_MESR_DERDR), ETPUCERSR and ETPUDERSR.
7. Save the status of non-correctable error overflows flagged in the eTPU Memory Error Status Register (eTPU_MESR_MESR) bit(s) CNCOVR/DNCOVR.

8. Set the eTPU Memory Error Control Register (eTPU_MESR_MECR) bit(s) CRDIS/DRDIS to the state(s) saved in the step 2. That assures the access coherency of a preempted Correctable Error Service Procedure (which may have set CRDIS/DRDIS before this procedure) is not disturbed.
9. Clear ETPUMESR bits CNCOVR/DNCOVR and CNCERR/DNCERR at once, in a single write to ETPUMESR.
10. Use the overrun status saved on step 7 to update the error statistics, as needed.
11. Check the other Global Exception flags in ETPUMCR; take or schedule the appropriate (application dependent) recover actions.
12. Clear the Global Exception, writing ETPUMCR bit GEC=1.

43.15.7.2 Correctable Error Service Procedure

The basic procedure below can be followed in a Memory Error Interrupt service routine (see Interrupts and Data Transfer Requests). It handles the service for both SCM and SDM errors, as they are symmetrical at this level. However, a real implementation is likely to split the procedure in two, one for each memory.

1. Confirm/identify the error source, checking assertion of the eTPU Memory Error Status Register (eTPU_MESR_MESR) bits CCERR/DCERR.
2. Set the eTPU Memory Error Control Register (eTPU_MESR_MECR) bits CRDIS=1 and/or DRDIS=1, to disable further updates to the respective report registers, guaranteeing coherency.
3. Check whether the report registers were updated by a correctable or non-correctable error; to do this, check the eTPU Code Error Report Syndrome Register (eTPU_MESR_CERSR) syndrome (for SCM) and/or the eTPU Data Error Report Syndrome Register (eTPU_MESR_DERSR) syndromes (for SDM, only those with the respective BE flags asserted). If it is a non-correctable error report, skip the steps 4 and 5 for the respective memory.
4. Identify the error source (host, microengine or MISC/CDC) by checking the field ERR_ACC in the register(s) eTPU Code Error Report Address Register (eTPU_MESR_CERAR) and eTPU Data Error Report Address Register (eTPU_MESR_DERAR); take or schedule the appropriate application-dependent recover actions, which may include soft-error fixing (see Error Fixing).

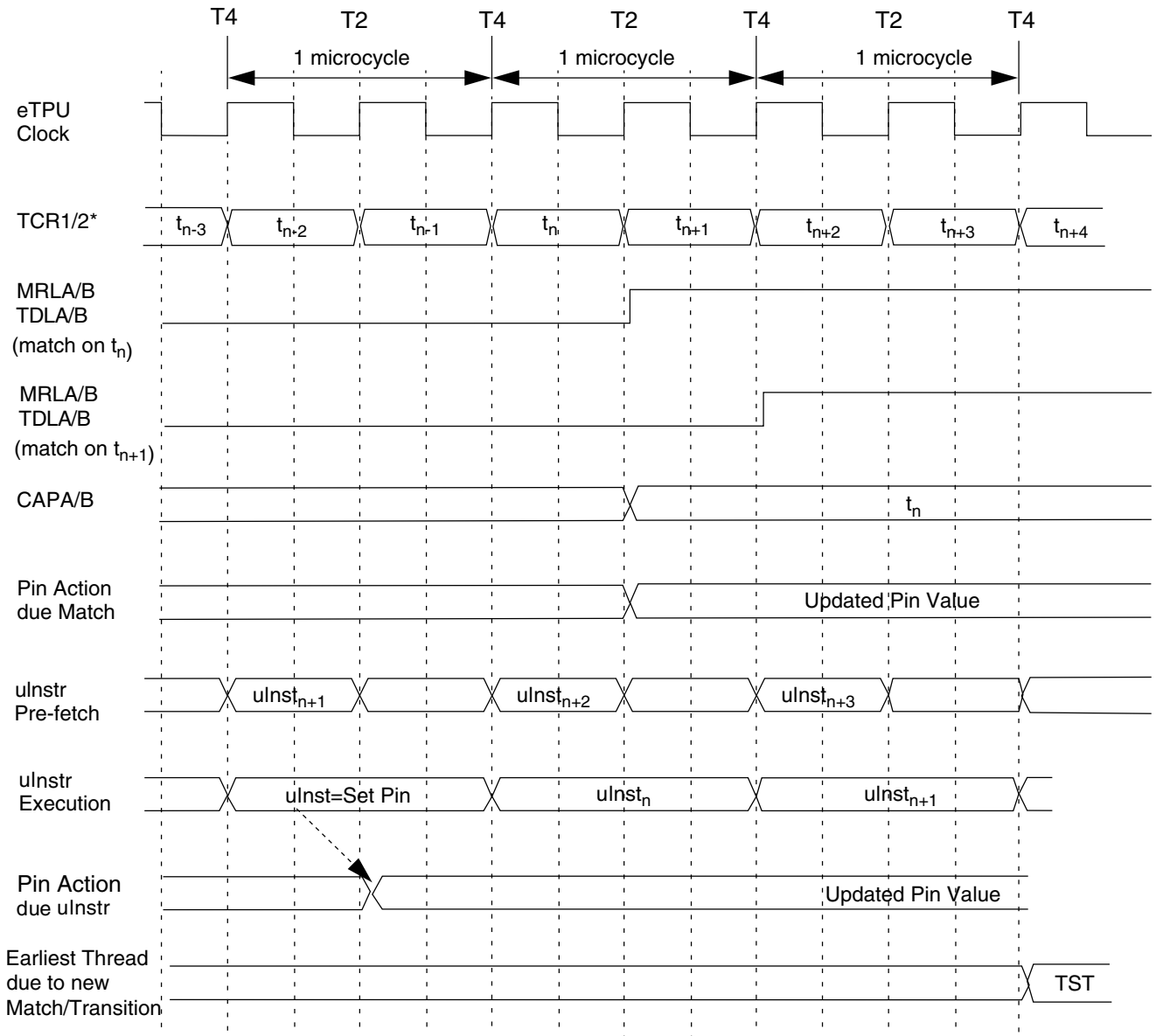
5. Update error statistics as needed; it may include recording the access source, address, data and syndrome from the registers ETPUCERAR and ETPUDERAR, eTPU Code Error Report Data Register (eTPU_MESR_CERDR) and eTPU Data Error Report Data Register (eTPU_MESR_DERDR), ETPUCERSR and ETPUDERSR.
6. Save the status of correctable error overflows flagged in the eTPU Memory Error Status Register (eTPU_MESR_MESR) bits CCOVR/DCOVR.
7. Set the ETPUMECR bit(s) CRDIS/DRDIS=0.
8. Clear ETPUMESR bit(s) CCOVR/DCOVR and CCERR/DCERR at once, in a single write to ETPUMESR.
9. Use the saved overrun status to update the error statistics, as needed.

43.16 Appendices

43.16.1 Microcycle and I/O Timing

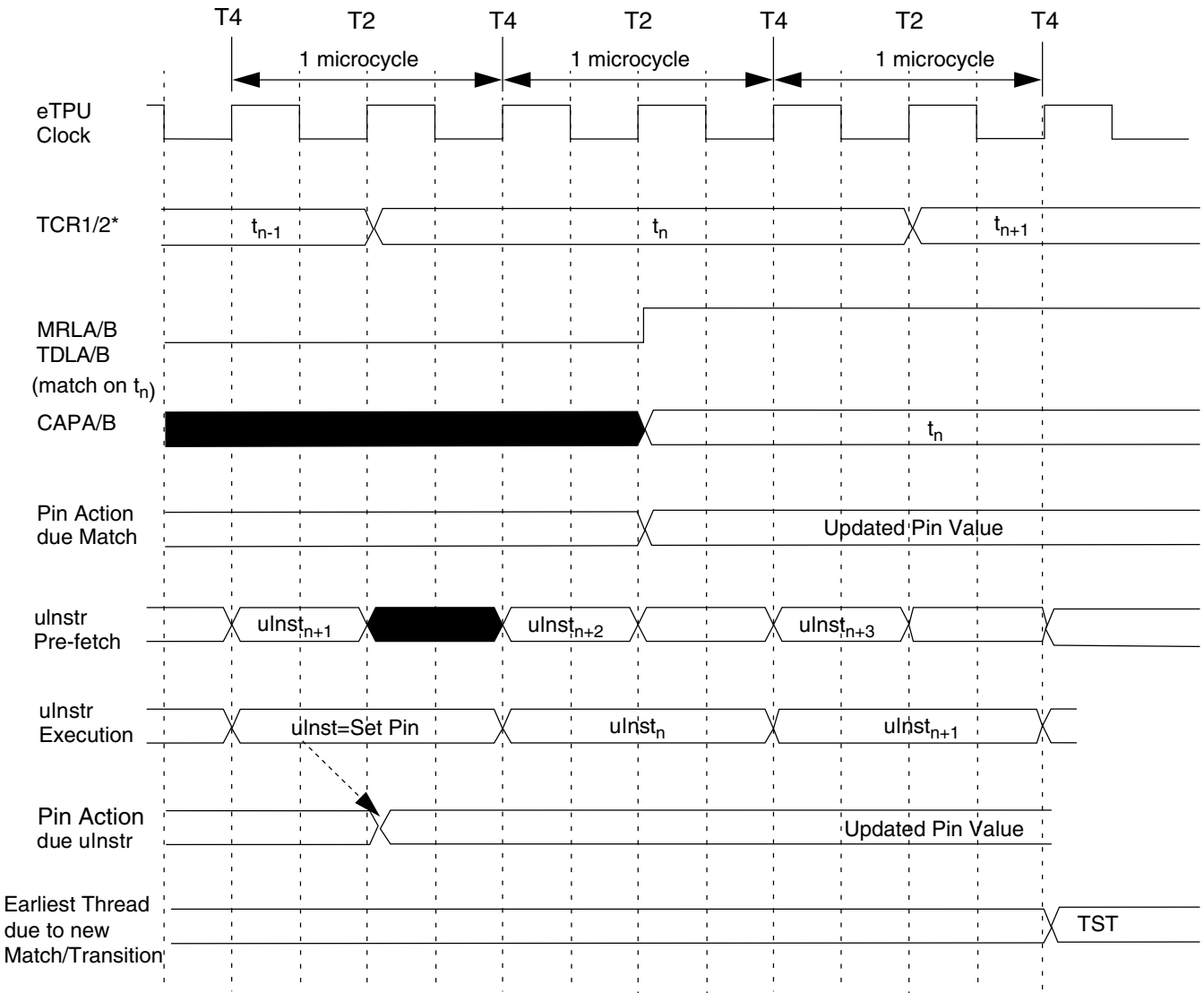
43.16.1.1 Execution and Channel Timing

The following figure shows the main timings related to microinstruction execution when channels and timebases run on T2 timing. For timings of eTPU block internal interface signals, consult the eTPU Integration Guide. For timings internal to the eTPU, refer to the **eTPU Creation Guide**.



Note: *TCR clock/prescaler selection = 1 x eTPU clock

Figure 43-343. Execution, Timebase and Channel T2 Timing



Note: *TCR clock/prescaler selection = 4x eTPU clock

Figure 43-344. Execution, Timebase and Channel T2/T4 Timing

The sequential occurrence of the four T states (T1-T4) constitutes a microcycle. Only T2 and T4 are taken as reference for timings, either internal or external. T2 and T4 have the timing of the positive eTPU clock pulses, and are used in most of the eTPU logic in an edge-triggered design style.

Two additional T states are derived from the eTPU clocks: $\overline{T2}$ and $\overline{T4}$. $\overline{T2}$ occurs when the eTPU loses SDM arbitration to a bus master. $\overline{T4}$ occurs in halt state (due to a breakpoint or device debug request assertion, for instance), or in stall state (due to an NDEDI queue full); see Development Support Features for more details.

$\overline{T2}$ and $\overline{T4}$ states are defined as microcycle timing states (not to be confounded with logic states) of one eTPU clock in which the T clocks continue to run, but the control signals associated with the clocks are unaffected. That is, no operation occurs during these states.

Both $\overline{T2}$ and $\overline{T4}$ states occur in multiples of two eTPU clocks to keep the microengine synchronized with the free running channels, which are unaffected and keep on working as in T2 and T4.

Thus, the eTPU has two types of timing states:

- $\overline{T2}$: hold execution for SDM access, from clock pulse T2 until one of the next T2 clock pulses of another microcycle.
- $\overline{T4}$: hold execution in debug mode or stall, from clock pulse T4 until one of the next T4 clock pulses of another microcycle.

The two following figures show the timing of $\overline{T2}$ and $\overline{T4}$ timing states, respectively.

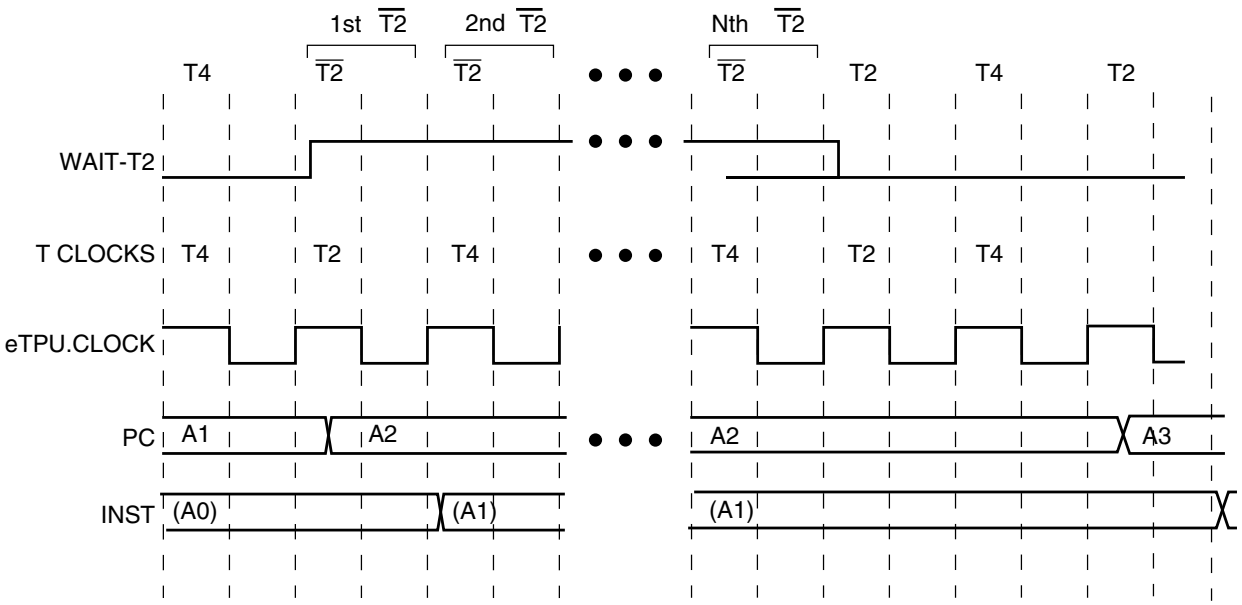


Figure 43-345. $\overline{T2}$ timing

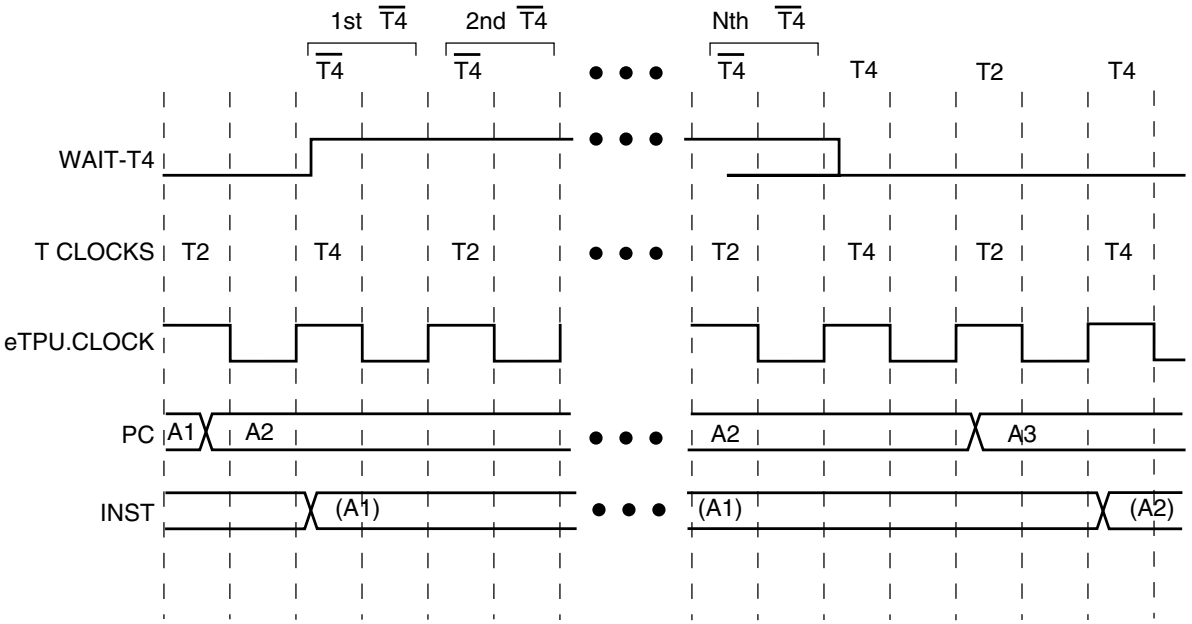


Figure 43-346. $\overline{T4}$ timing

43.16.1.2 Input/Output Signal Delays

The synchronizer, filter and edge detection logic delay the input signal transitions.

The Filter Delay varies with the filter clock (eTPU Engine 1 Configuration Register (eTPU_SCR_ECR_ENG1) and eTPU Engine 2 Configuration Register (eTPU_SCR_ECR_ENG2) field FPSCK) and the filter mode used, as shown in the Table 43-353 . For any given transition, it depends on the phase of the filter clock when the input transition happens. In integration mode (TCRCLK filtering only), it also depends on the state of the integrator counter. The Total Delay is defined as the number of eTPU clock rising edges between the input transition and the setting of TDLA/B, TCR1/2 incrementing, or EAC tooth sensing (TCRCLK) in angle mode. The synchronizer delay is 2 or 3 eTPU clocks, depending on the phase of the synchronizer when the input transition happens. The edge detection takes 1 more eTPU clock. The total delays are, thus:

```
Min. Total Delay = Min. Synchronizer Delay + Min. Filter Delay + Edge Detection Delay
Min. Total Delay = 3 + Min. Filter Delay
Max. Total Delay = Max. Synchronizer Delay + Max. Filter Delay + Edge Detection Delay
Max. Total Delay = 4 + Max. Filter Delay
```

The channel filters can be bypassed, so nullifying the filter delays in the equations above.

See the chip-specific eTPU information for connections between eTPU and input glitch filters.

The channel output flip-flops drive the eTPU output signals directly, without any synchronous delays.

43.16.2 Channel Mode Summary

The initial state considered for all submodes is channel flags MRLEA and MRLEB asserted, MRLA, MRLB, TDLA and TDLB negated. From initial state one can follow the table below and verify how each submode behaves in a determined sequence of events. Note that the actions performed by an event type depend on all previous events following the initial state, for a given channel submode.

There are three columns for each event: one for event type, one for enable/disable actions and one for capture. **Event type** column can be matchA, matchB, transA and transB (for double transition modes). **Enable/disable actions** column (identified as "**[blocks] (enables)**" in column head) specifies which other events are enabled or disabled. Initially disabled events (specified in **"initially blocked"** column) are usually enabled by other events.

In double transition submodes, the first transition detected is always considered transA and the second is considered transB. This means that transA event actually enables the detection of transB event. This is not explicit in the table, since it is a general behavior for all double transition submodes.

A sequence of four events (two matches and two transitions) are necessary to describe the behavior of some channel submodes. When a determined sequence of events has less than four events, the other event columns are left blank.

Cells in an **"event type"** column that have light-grayed background indicate that a service request is generated. More than one event in the same event sequence can issue service request.

Note

The two tables below do not exhaust all possibilities of channel logic event sequences, because it does not account for possible microcode interventions. For instance, if matches are blocked

by first transition and microcode resets TDLA, the matches become enabled again, and from this point on the channel behaves as if the first transition had never occurred.

Table 43-427. Channel Mode Summary 1 of 2

Mode	Initially Blocked	1st Event			2nd Event		
		Event Type	[Blocks] (Enables)[1]	Capture	Event Type	[Blocks] (Enables)	Capture
em_nb_st	none	match 1/2[2]	none	1/2	match 2/1	none	2/1
					trans 1	[matches]	both
		trans 1	[matches]	both	trans 2	none	2
em_nb_dt	none	match 1/2	none	1/2	match 2/1	none	2/1
		match 1	none	1	trans 1	none	1
		match 2	none	2	trans 1	[match 1]	1
		trans 1	[match 1]	1	match 2	none	2
					trans 2	[match 2]	2
em_b_st	none	match 1	[match 2]	both	trans 1	[matches]	both
		match 2	[match 1]	both			
		trans 1	[matches]	both	trans 2	none	2
em_b_dt	none	match 1	[match 2]	both	trans 1	none	1
		match 2	[match 1]	both			
		trans 1	[match 1]	1	match 2	none	2
					trans 2	[match 2]	2
bm_st	none	match 1/2	none	1/2	match 2/1	none	2/1
					trans 1	[matches]	both
		trans 1	[matches]	both	trans 2	none	2
bm_dt	none	match 1/2	none	1/2	match 2/1	none	2/1
					trans 1	none	1
		trans 1	none	1	match 1/2	none	none/2
					trans 2	[matches]	2
m2_st	trans 1	match 1	(trans 1)	1	match 2	none	2
					trans 1	[matches]	both
		match 1 and match 2	(trans 1)	both	trans 1	[matches]	both
		match 2	[match 1]	2	n/a	n/a	n/a
m2_dt	trans 1	match 1	(trans 1)	1	trans 1	none	1
					match 2	none	2
		match 1 and match2	(trans 1)	both	trans 1	none	1
		match 2	[match 1]	2	n/a	n/a	n/a
m2_o_st	trans 1 match 2	match 1	(match 2) (trans 1)	1	trans 1	[matches]	both
					match 2	[trans 1]	2

Table continues on the next page...

Table 43-427. Channel Mode Summary 1 of 2 (continued)

Mode	Initially Blocked	1st Event			2nd Event		
		Event Type	[Blocks] (Enables)[1]	Capture	Event Type	[Blocks] (Enables)	Capture
m2_o_dt	trans 1 match 2	match 1	(match 2) (trans 1)	1	trans 1	none	1
					match 2	[trans 1]	2
sm_st[3]	match 2	match 1	none	both	trans 1	none	both
		trans 1	[match 1]	both	trans 2	none	2
sm_dt	match 2	match 1	none	both	trans 1	none	1
		trans 1	none	1	match 1	none	2
		trans 1	none	1	trans 2	[match1]	2
sm_st_e[4]	match 2 trans 2	match 1	none	1	trans 1	none	1
		trans 1	[match 1]	1	n/a	n/a	n/a

1. Transition 1 always enables Transition 2.
2. Shading indicates that a service request is generated.
3. sm_st is compatible with TPU3 channel logic.
4. It is not possible to include all functionality of this submode in table.

Table 43-428. Channel Mode Summary 2of 2

Mode	Initially Blocked	3rd Event			4th Event		
		Event Type	[Blocks] (Enables)[1]	Capture	Event Type	[Blocks] (Enables)	Capture
em_nb_st	none	trans 1[2]	[matches]	both	trans 2	none	2
					n/a	n/a	n/a
		trans 2	none	2	trans 2	none	2
em_nb_dt	none	trans 1	none	1	trans 2	none	2
		match 2	none	2	trans 2	none	2
		trans 2	[match 2]	2	n/a	n/a	n/a
		trans 2	[match 2]	2	n/a	n/a	n/a
		trans 2	none	2	n/a	n/a	n/a
em_b_st	none	trans 2	none	2	n/a	n/a	n/a
em_b_dt	none	trans 2	none	2	n/a	n/a	n/a
		trans 2	none	2	n/a	n/a	n/a
bm_st	none	trans 1	[matches]	both	trans 2	none	2
		trans 2	none	2	n/a	n/a	n/a
bm_dt	none	trans 1	none	1	trans 2	[matches]	2
		match 2/1	none	2 / none	trans 2	[matches]	2
		trans 2	[matches]	2	n/a	n/a	n/a
		match 2/1	none	2 / none	trans 2	[matches]	2
		trans 2	[matches]	2	n/a	n/a	n/a

Table continues on the next page...

Table 43-428. Channel Mode Summary 2of 2 (continued)

Mode	Initially Blocked	3rd Event			4th Event		
		Event Type	[Blocks] (Enables)[1]	Capture	Event Type	[Blocks] (Enables)	Capture
m2_st	trans 1	trans 1	[matches]	both	trans 2	none	2
		trans 2	none	2	n/a	n/a	n/a
		trans 2	none	2	n/a	n/a	n/a
m2_dt	trans 1	trans 2	[match 2]	2	n/a	n/a	n/a
		match 2	none	2	trans 2	none	2
		trans 1	none	1	trans 2	none	2
		trans 2	none	2	n/a	n/a	n/a
m2_o_st	trans 1 match 2	trans 2	none	2	n/a	n/a	n/a
m2_o_dt	trans 1 match 2	trans 2	[match 2]	2	n/a	n/a	n/a
		match 2	[trans 2]	2	n/a	n/a	n/a
sm_st[3]	match 2	trans 2	none	2	n/a	n/a	n/a
sm_dt	match 2	trans 2	none	2	n/a	n/a	n/a
		trans 2	none	2	n/a	n/a	n/a
sm_st_e[4]	match 2 trans 2	n/a	n/a	n/a	n/a	n/a	n/a

1. Transition 1 always enables Transition 2.
2. Shading indicates that a service request is generated.
3. sm_st is compatible with TPU3 channel logic.
4. It is not possible to include all functionality of this submode in table.

43.16.3 MISC Algorithm

The MISC generator is based on the following polynomial:

$G(x) = 1 + x^1 + x^2 + x^{22} + x^{31}$ (equivalent to feedback mask = 0x80400007)

The MISC signature generation starts by clearing the MISC Accumulator value to 0 and preloading the MISC Counter with the highest SCM address. It then steps through each address decrementing the counter, reading 32 bit values and following the algorithm below:

```
If the least significant bit in MISC is 1 then
    MISC = MISC right shifted by 1 bit
    MISC = MISC XOR 0x80400007
else
    MISC = MISC right shifted by 1 bit
end if
MISC = MISC XOR RAM data
```

The code example below shows an excerpt of C code that calculates the MISC signature for a given array of data, based on the previous algorithm:

```
#define SCM_size    (MAX_SCM_ADDRESS / 4)    /* last byte address - converted to 32-bit word */
#define POLY        0x80400007               /* G(x) = 1 + x^1 +x^2 + x^22 + x^31 */
/*****************************************************************************
 FUNCTION : void calc_misc()
 PURPOSE : This function calculates the MISC value.
 INPUTS NOTES : none
 RETURNS NOTES : MISC value
 GENERAL NOTES : the array 'unsigned int data[]' represents the actual memory
                 array, organized in 32-bit words.
*****************************************************************************/
unsigned int calc_misc (void)
{
  int j; /* loop counter */
  unsigned int misc = 0;
  for (j = (SCM_size-1); j >= 0 ; j--) {    /* SCM_size has the number of 32-bit words in SCM
*/
      if (misc & 0x1) {
          misc >>= 1;
          misc ^= POLY;
      }
      else {
          misc >>= 1;
      }
      misc ^= data[j];          /* data[j] is the actual 32-bit word taken from the SCM
array
*/
  }
  return (misc);          /* final signature calculated */

};
```

The value calculated by this algorithm must be loaded into register eTPU MISC Compare Register (eTPU_SCR_MISCCMPR) prior to activating the SCM MISC calculator in eTPU. Once the MISC calculator is activated (bit SCMMISEN in register eTPU Module Configuration Register (eTPU_SCR_MCR) is written to 1) eTPU itself will start this procedure[38] reading the SCM whenever allowed by microengine. At the end of the cycle, when all the array has been read and the SCM signature is calculated, the Host CPU can be notified via Global Exception if the MISC Accumulator does not match the value in ETPUMISCCMPR.

The average time taken by MISC to complete the signature of the whole SCM can be given by the formula:

```
Average MISC period = S / (4 * F * (1 - L))
```

where f is clock frequency, S is SCM size in bytes and L is eTPU load (as a percentage of execution clocks over a period of time, including TST clocks).

Further detail on MISC calculation can be found on SCM Test - Multiple Input Signature Calculator. The application note **AN2192 - Detecting Errors in the Dual Port RAM (DPTRAM) Module** is also a good source of information (although it refers to TPU) on MISC signature.

38. eTPU MISC hardware is optimized to read 32-bit words from memory and to calculate this CRC in parallel, rather than shifting one bit at a time. The actual implementation inside eTPU, although bringing to the same results, does not match exactly the algorithm shown here.

Chapter 44
Reaction Module 2 (REACM2)

44.1 Reaction Module 2 chip-specific module information

The Reaction Module 2 (REACM2) is a dedicated module designed to control 10 output channels. Each channel has three output pins.

The device uses the Reaction Module 2 to provide closed loop control functionality. It receives SAR ADC results from the BCTU and drives output signals. The device supports one instance of reaction module.

- Timer input channel 0-3 is directly connected with the eTPU (engine1).
- Timer inputs channel 4-15 of the Reaction Module 2 may come from eTPU or eMIOS. The mux is implemented for these timer inputs in IOMUX (see Generic pins/balls).
- The Reaction Module 2 receives the conversion results from SAR-ADC via BCTU. BCTU triggers the SAR-ADC and then sends the conversion results it received from the SAR-ADC to the Reaction Module.

Table 44-1. REACM2 Inputs

REACM2 channel	eTPU_A	eTPU_B	eMIOS0	eMIOS1
0	10	—	—	—
1	11	—	—	—
2	12	—	—	—
3	13	—	—	—
4	14	10	8	—
5	15	11	9	—
6	16	12	10	—
7	17	13	11	—
8	18	14	—	12
9	19	15	—	13
10	20	16	—	14
11	21	17	—	15

Table continues on the next page...

Table 44-1. REACM2 Inputs (continued)

REACM2 channel	eTPU_A	eTPU_B	eMIOS0	eMIOS1
12	—	18	12	8
13	—	19	13	9
14	—	20	14	10
15	—	21	15	11

The following figure shows the interaction of the Reaction Module 2 with other modules in the device.

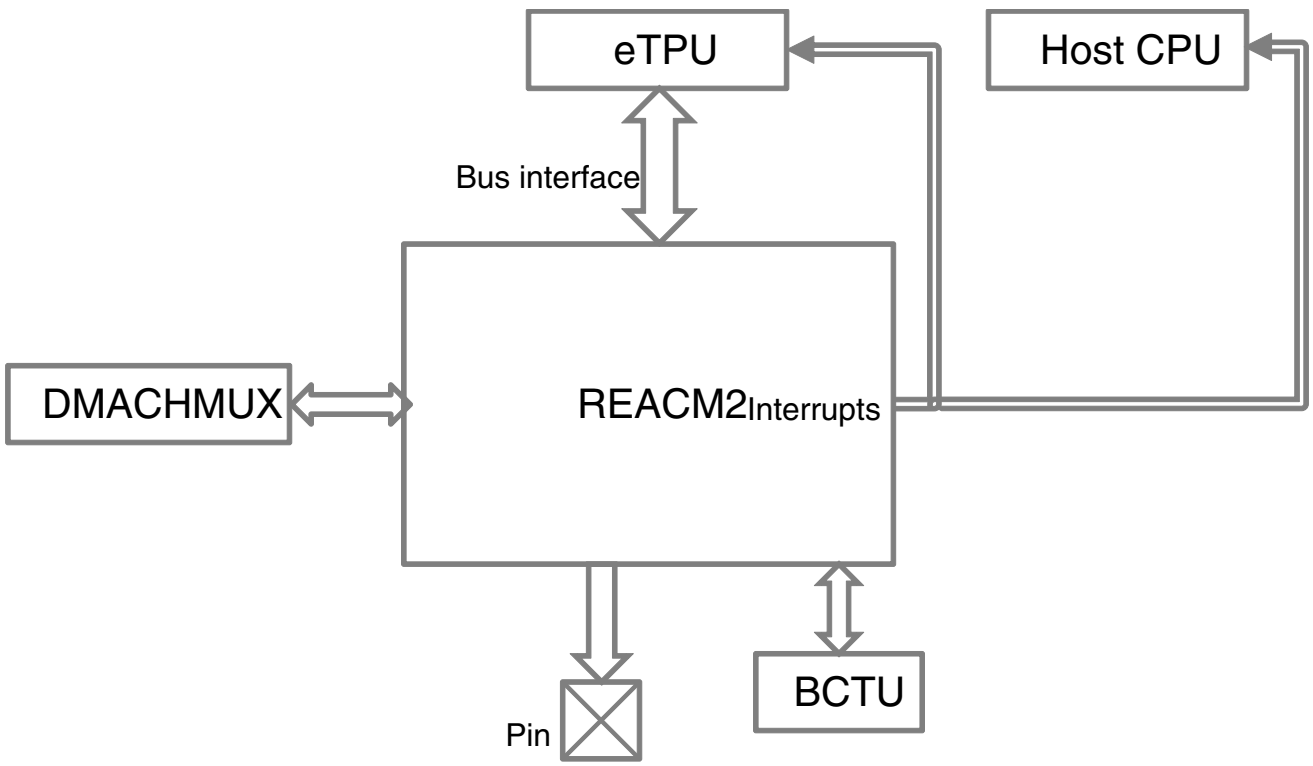


Figure 44-1. Reaction Module 2 SoC Interaction

44.2 Introduction

The REACM2 is composed of 10 reaction channels. Each channel contains three outputs. Its main application is in the area of solenoid control for direct injection systems, valve control in automatic transmissions, and other current control applications. It is connected to the on-chip ADC which monitors the current on the solenoid or valve. Based on that the reaction channel generates an output signal that modulates the current circulating in

the solenoid or valve. It is a cost-effective solution due to extensive sharing of several resources among channels and parameterized register banks for adequate dimensioning of resources and functionality.

44.2.1 Features

The REACM2 features include:

- Per channel architecture for independent output control
- Interface with on-chip ADC for fast response times
- Interfaced to timer modules
- Hardware connection with on-chip timer channels with channel routing capability
- Shared Modulation Control bank
- Dynamic timer allocation
- Three outputs per channel to support different driver architectures
- Flexibility to operate based on timing and threshold
- On-the-fly capture of ADC result reference for fast calibration
- Open and short circuit monitoring capability

44.2.2 Modes of Operation

Operation modes of this module are described below.

44.2.2.1 Programming Mode

After reset, the Reaction module is in the Programming mode. In this mode all channels are disabled and outputs are at logic zero. Note that this state does not necessarily mean that zero is the neutral state for the channel load, thus care must be taken in order to disconnect the channel from the load in this case.

In the programming mode the host CPU writes all module parameters including:

1. Modulation word data
2. Channel control data

3. Threshold data
4. Timer Bank data
5. Hold (on/off) Timer bank data
6. Timer router data
7. ADC router data
8. Period Shift Delay time
9. Period Generator timers

The last action to perform is to *enable* the channel. After that the channel is able to respond to timer signals and ADC data, thus able to perform modulation on the output pins. It is recommended to keep the timer signals inactive until all data to all reaction channels are programmed and all channels have been put in the *enabled* mode.

44.2.2.2 Low Power Mode

Coming out from reset all channels are in the disabled state. The channel may also be in low power mode depending on an instantiation parameter that configures the initial state of the MDIS in the Module Configuration Register. If MDIS=1 the module clock may be disabled, allowing for a low power state. The low power mode is controlled either by the MDIS bit or by a global stop signal. There is no explicit clock gating implemented in hardware within the Reaction Module. Reaction module clock gating is implemented in the SoC. Low power mode must be entered only when all channels are disabled by CHEN=00.

44.2.2.3 Channel Modes

After a channel is put in *enabled* mode that channel is also said to be in the *normal* mode of operation, which means it responds to timer signals from the timer inputs connected to the Reaction Module and also to ADC results received from the on-chip ADC module. Channel outputs are controlled in accordance with those inputs in order to perform an output modulation process. When performing a modulation the Reaction Channel is said to be in the *active state*. The Reaction Channel modes are described below in relation to the ability to execute a modulation:

- Disabled: the channel cannot execute modulation.
- Enabled: the channel is able to execute a modulation. It may be in the *Active* or *Inactive* states.
  - Inactive state: the channel is not executing a modulation.
  - Active state: the channel is executing a modulation.

44.2.2.4 Debug Mode

The Reaction Module Debug operation is defined by bits FRZ and FREN in the Module Configuration Register. In debug mode all Timers are freezed (halted), including the timers in the Shared Time Bank and Hold (on/off) timers. The module can enter debug mode in two different ways, either by software control or by the hardware debug input signal controlled by the chip logic. In both cases the Reaction Module enters debug mode only if enabled by the FREN bit:

- If both FREN and FRZ bits are set, the module enters debug mode.
- If FREN bit is set and a global debug signal is asserted, the module enters debug mode.

In debug mode, the channel outputs are held at HOD, LOD, or DOFF state, as determined by the channel current state. When resuming normal operation after exiting debug mode, the channel performs the read of the first Mod Word and the channel output is set to DOFF until the next timer control rising edge occurs.

MAXL flag is the only flag that operates in debug mode. All other flags retain the present state while the module is in debug mode. Note that the corresponding error flag in the Global Error Flag register is also set.

The ADC Input Sensor Input Register allows direct access for write to the TAG and ADC result values input to the Reaction Module. This software control may be used for module debug purposes.

44.2.3 Block Diagram

Figure 44-2 shows the on-chip connections of the reaction module. The ADC module is the source of data monitored from external sensors, usually voltage information, which is used by the reaction channel modulation to generate a PWM signal control signal in order

to maintain the load current within a certain predefined boundary. eTPU is commonly used as the source for controlling the activation of a reaction channel. Alternatively eMIOS or PIT can be used to provide that control signal.

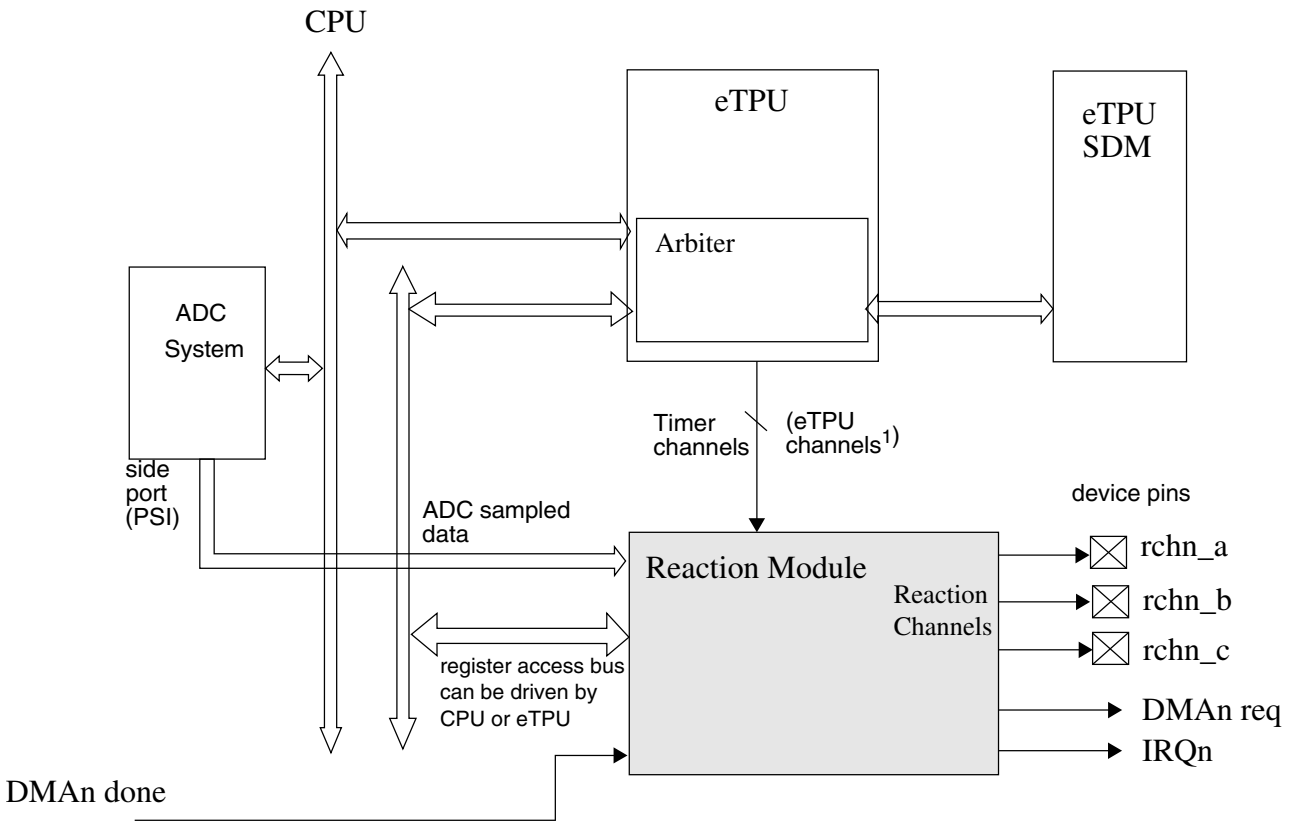


Figure 44-2. Reaction Module System Interconnection

The Reaction Module provides 10 channels with three outputs per channel. The channel outputs are programmed independently from each other. The outputs do not have a predefined ON or OFF state meaning as the definition of ON and OFF depends upon the application.

Figure 44-3 describes the interconnection between the reaction channel and the blocks in the Reaction Module. The indexes next to the blocks indicate the order in which each submodule executes an action during the modulation process. A simplified sequence of these steps is described as follows:

1. A new ADC result from the BCTU module is received, triggering one reaction channel
2. The triggered reaction channel generates the Modulation Word address based on the data programmed in the internal configuration register
3. A Modulation word is selected by the reaction channel

4. The selected Modulation word generates the address for the Threshold Bank
5. The Threshold bank selected data is compared against the incoming ADC result
6. The comparison result is sent back to the channel
7. The reaction channel executes modulation action and updates the channel output

In order to execute steps 1 through 7 the Reaction module takes five Module clock cycles.

The reaction channel sets the outputs according to HOD or LOD fields provided by the selected Modulation word.

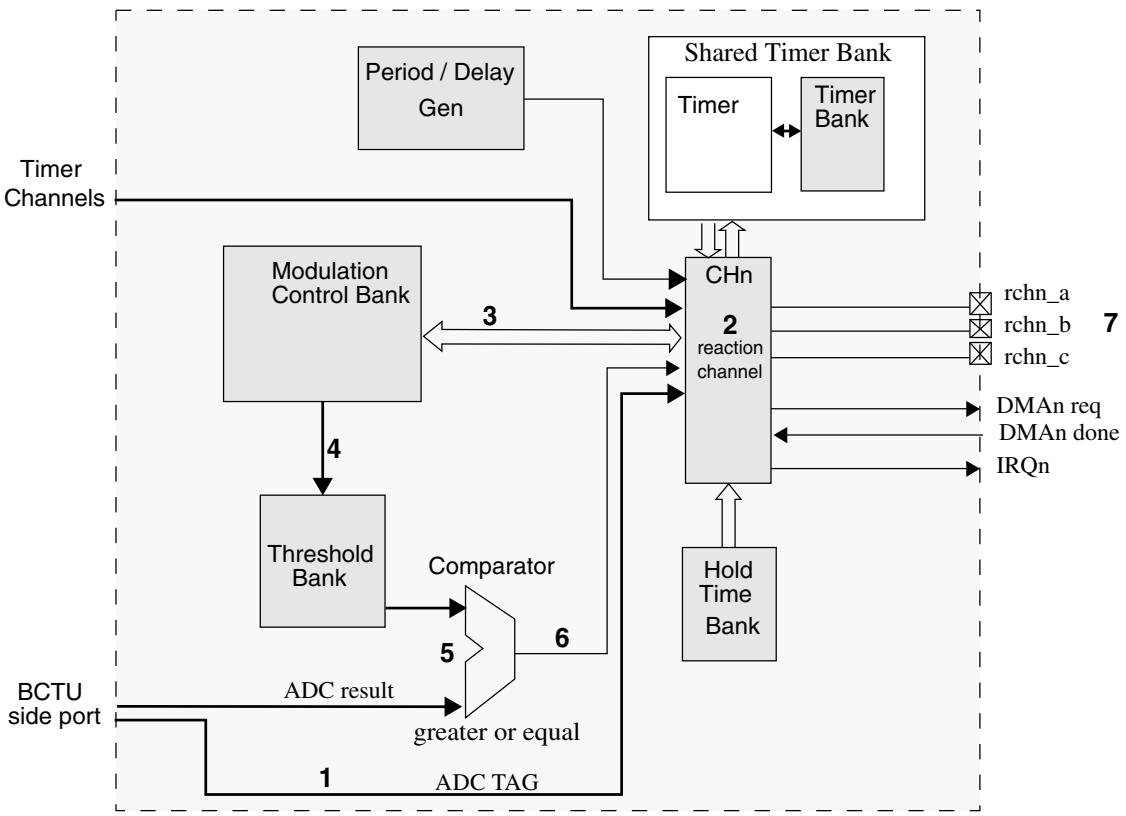


Figure 44-3. Channel Interaction with internal submodules

Figure 44-4 shows the Reaction Module's internal architecture. The channels are controlled by a Modulation Control Word provided by the Modulation Control Bank submodule. The Modulation Word has information about the type of modulation to be executed as well as the addresses for the Threshold and Timer banks. The channel stores information such as the address of the modulation word and initial values for the output pins after the channel is *enabled*.

Note that the modulation process only initiates after the channel is moved to the *active* state. This process will be described in more detail in the following sections.

After the activation, the channel sets the outputs with a predefined value stored in the channel configuration register. Only after this initialization process the channel is ready to start a modulation process. Note that the channel only considers an ADC result as valid if it is within a window defined by the timer input, usually connected to the on-chip eTPU. The Reaction Channel can access more than one Modulation Word by incrementing the internal MODULATION_ADDR field in the channel configuration register. The original value of this field is preserved thus when a new modulation cycle is initiated the same modulation address value can be used.

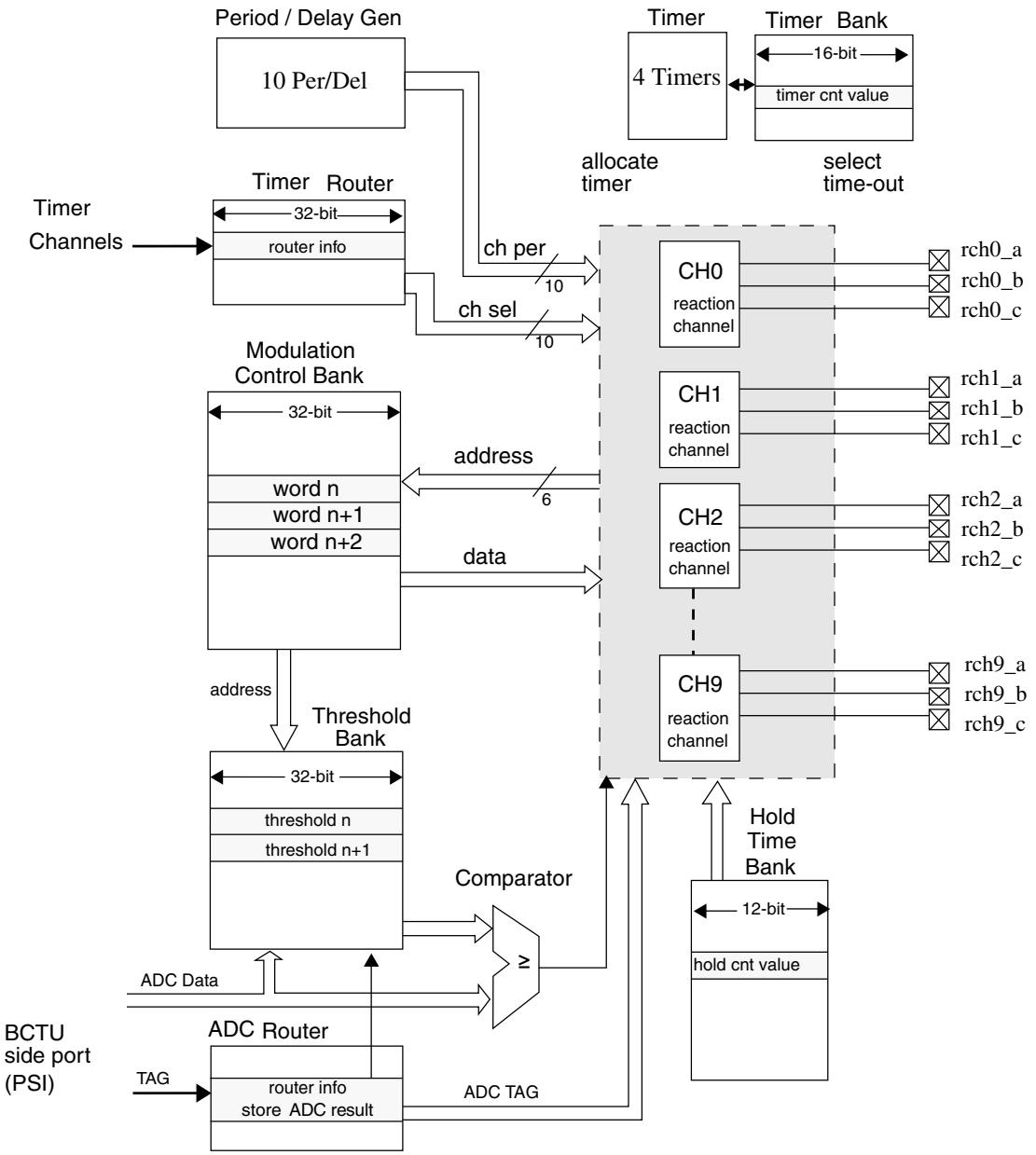


Figure 44-4. Reaction Module Block Diagram

Timer channels, such as eTPU channel outputs, are connected to the Timer Channel Router. This submodule routes each timer channel to a reaction channel. Note that one timer channel can be routed to more than one reaction channel.

The modulation process initiates when an ADC result arrives and the time window is active. The ADC Router sends a trigger signal to all channels indicating that an ADC result is available. The channel address resolution is based on the TAG field received with the incoming ADC result. After decoding the TAG field the channel accesses the Modulation Word using the information stored in the channel configuration register.

The Threshold Bank submodule stores values to be used on the comparison with incoming ADC results. The address for the stored values is generated by the Modulation Control Bank. This address generation is actually executed in a two-step process since the Modulation Word is first addressed by the channel which then generates the address for the Threshold Bank. After having both inputs defined the Comparator generates the comparison result back to the channels.

The Hold timer bank address is also stored in the modulation word. The reaction channel uses that information for the modulation process which requires the output to remain ON or OFF during a certain amount of time. The Hold counter itself is located inside each channel.

The Period Delay generator provides timing signals for the channels operating in the threshold-period modulation mode. It provides period starting signal for each channel with a configurable constant delay between two consecutive channels.

44.3 Signal Description

The following table shows the chip level signals for the Reaction module.

Table 44-2. Signal Properties

Name	Function
RCHn_a	Output pin 'a' of Reaction Channel 'n'
RCHn_b	Output pin 'b' of Reaction Channel 'n'
RCHn_c	Output pin 'c' of Reaction Channel 'n'

44.3.1 REACM2_RCHn — REACM Channel (n) Output Pin a, b and c

Each Reaction Channel provides three outputs to be connected to device primary output PADs. These outputs can be configured as necessary in order to meet application requirements. Thus the active state as well as overall functionality are defined by the registers within the Reaction Module and the defined sequence of operation. The reset state for these outputs is zero.

44.4 Memory Map and Register Definition

This section provides a detailed description of all Reaction Module registers accessible to the end user.

The REACM2 is in the eTPU address space to allow the eTPU direct access to the registers.

NOTE

For reset value of the REACM2_MCR, refer to the section describing this register.

REACM2 memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration (REACM2_MCR)	32	R/W	4000_0000h	44.4.1/1687
4	Timer Configuration (REACM2_TCR)	32	R/W	0000_0000h	44.4.2/1689
8	Threshold Router Register (REACM2_THRR)	32	R/W	0000_0000h	44.4.3/1690
10	ADC Sensor Input Register (REACM2_SINR)	32	R/W	0000_0000h	44.4.4/1691
14	Period Generator Configuration Register (REACM2_PCR)	32	R/W	0000_0000h	44.4.5/1692
18	Period Shift Delay Configuration Register (REACM2_PSCR)	32	R/W	0000_0000h	44.4.6/1693
20	Global Error Flag Register (REACM2_GEFR)	32	R	0000_0000h	44.4.7/1694
100	Channel n Configuration Register (REACM2_CHCR0)	32	R/W	0000_0000h	44.4.8/1695
104	Channel n Status Register (REACM2_CHSR0)	32	R	0000_0000h	44.4.9/1699
108	Channel n Router Register (REACM2_CHRR0)	32	R/W	0000_0000h	44.4.10/ 1702
110	Channel n Configuration Register (REACM2_CHCR1)	32	R/W	0000_0000h	44.4.8/1695
114	Channel n Status Register (REACM2_CHSR1)	32	R	0000_0000h	44.4.9/1699
118	Channel n Router Register (REACM2_CHRR1)	32	R/W	0000_0000h	44.4.10/ 1702
120	Channel n Configuration Register (REACM2_CHCR2)	32	R/W	0000_0000h	44.4.8/1695

Table continues on the next page...

REACM2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
124	Channel n Status Register (REACM2_CHSR2)	32	R	0000_0000h	44.4.9/1699
128	Channel n Router Register (REACM2_CHRR2)	32	R/W	0000_0000h	44.4.10/ 1702
130	Channel n Configuration Register (REACM2_CHCR3)	32	R/W	0000_0000h	44.4.8/1695
134	Channel n Status Register (REACM2_CHSR3)	32	R	0000_0000h	44.4.9/1699
138	Channel n Router Register (REACM2_CHRR3)	32	R/W	0000_0000h	44.4.10/ 1702
140	Channel n Configuration Register (REACM2_CHCR4)	32	R/W	0000_0000h	44.4.8/1695
144	Channel n Status Register (REACM2_CHSR4)	32	R	0000_0000h	44.4.9/1699
148	Channel n Router Register (REACM2_CHRR4)	32	R/W	0000_0000h	44.4.10/ 1702
150	Channel n Configuration Register (REACM2_CHCR5)	32	R/W	0000_0000h	44.4.8/1695
154	Channel n Status Register (REACM2_CHSR5)	32	R	0000_0000h	44.4.9/1699
158	Channel n Router Register (REACM2_CHRR5)	32	R/W	0000_0000h	44.4.10/ 1702
160	Channel n Configuration Register (REACM2_CHCR6)	32	R/W	0000_0000h	44.4.8/1695
164	Channel n Status Register (REACM2_CHSR6)	32	R	0000_0000h	44.4.9/1699
168	Channel n Router Register (REACM2_CHRR6)	32	R/W	0000_0000h	44.4.10/ 1702
170	Channel n Configuration Register (REACM2_CHCR7)	32	R/W	0000_0000h	44.4.8/1695
174	Channel n Status Register (REACM2_CHSR7)	32	R	0000_0000h	44.4.9/1699
178	Channel n Router Register (REACM2_CHRR7)	32	R/W	0000_0000h	44.4.10/ 1702
180	Channel n Configuration Register (REACM2_CHCR8)	32	R/W	0000_0000h	44.4.8/1695
184	Channel n Status Register (REACM2_CHSR8)	32	R	0000_0000h	44.4.9/1699
188	Channel n Router Register (REACM2_CHRR8)	32	R/W	0000_0000h	44.4.10/ 1702
190	Channel n Configuration Register (REACM2_CHCR9)	32	R/W	0000_0000h	44.4.8/1695
194	Channel n Status Register (REACM2_CHSR9)	32	R	0000_0000h	44.4.9/1699
198	Channel n Router Register (REACM2_CHRR9)	32	R/W	0000_0000h	44.4.10/ 1702
300	Shared Timer Bank (REACM2_STBK0)	32	R/W	0000_0000h	44.4.11/ 1703
304	Shared Timer Bank (REACM2_STBK1)	32	R/W	0000_0000h	44.4.11/ 1703
308	Shared Timer Bank (REACM2_STBK2)	32	R/W	0000_0000h	44.4.11/ 1703
30C	Shared Timer Bank (REACM2_STBK3)	32	R/W	0000_0000h	44.4.11/ 1703
310	Shared Timer Bank (REACM2_STBK4)	32	R/W	0000_0000h	44.4.11/ 1703
380	Hold Timer Bank (REACM2_HOTBK0)	32	R/W	0000_0000h	44.4.12/ 1703

Table continues on the next page...

REACM2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
384	Hold Timer Bank (REACM2_HOTBK1)	32	R/W	0000_0000h	44.4.12/ 1703
388	Hold Timer Bank (REACM2_HOTBK2)	32	R/W	0000_0000h	44.4.12/ 1703
38C	Hold Timer Bank (REACM2_HOTBK3)	32	R/W	0000_0000h	44.4.12/ 1703
390	Hold Timer Bank (REACM2_HOTBK4)	32	R/W	0000_0000h	44.4.12/ 1703
400	Threshold Bank (REACM2_THBK0)	32	R/W	0000_0000h	44.4.13/ 1704
404	Threshold Bank (REACM2_THBK1)	32	R/W	0000_0000h	44.4.13/ 1704
408	Threshold Bank (REACM2_THBK2)	32	R/W	0000_0000h	44.4.13/ 1704
40C	Threshold Bank (REACM2_THBK3)	32	R/W	0000_0000h	44.4.13/ 1704
410	Threshold Bank (REACM2_THBK4)	32	R/W	0000_0000h	44.4.13/ 1704
414	Threshold Bank (REACM2_THBK5)	32	R/W	0000_0000h	44.4.13/ 1704
418	Threshold Bank (REACM2_THBK6)	32	R/W	0000_0000h	44.4.13/ 1704
41C	Threshold Bank (REACM2_THBK7)	32	R/W	0000_0000h	44.4.13/ 1704
420	Threshold Bank (REACM2_THBK8)	32	R/W	0000_0000h	44.4.13/ 1704
424	Threshold Bank (REACM2_THBK9)	32	R/W	0000_0000h	44.4.13/ 1704
428	Threshold Bank (REACM2_THBK10)	32	R/W	0000_0000h	44.4.13/ 1704
42C	Threshold Bank (REACM2_THBK11)	32	R/W	0000_0000h	44.4.13/ 1704
430	Threshold Bank (REACM2_THBK12)	32	R/W	0000_0000h	44.4.13/ 1704
434	Threshold Bank (REACM2_THBK13)	32	R/W	0000_0000h	44.4.13/ 1704
438	Threshold Bank (REACM2_THBK14)	32	R/W	0000_0000h	44.4.13/ 1704
43C	Threshold Bank (REACM2_THBK15)	32	R/W	0000_0000h	44.4.13/ 1704
440	Threshold Bank (REACM2_THBK16)	32	R/W	0000_0000h	44.4.13/ 1704
444	Threshold Bank (REACM2_THBK17)	32	R/W	0000_0000h	44.4.13/ 1704

Table continues on the next page...

REACM2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
448	Threshold Bank (REACM2_THBK18)	32	R/W	0000_0000h	44.4.13/ 1704
44C	Threshold Bank (REACM2_THBK19)	32	R/W	0000_0000h	44.4.13/ 1704
450	Threshold Bank (REACM2_THBK20)	32	R/W	0000_0000h	44.4.13/ 1704
454	Threshold Bank (REACM2_THBK21)	32	R/W	0000_0000h	44.4.13/ 1704
458	Threshold Bank (REACM2_THBK22)	32	R/W	0000_0000h	44.4.13/ 1704
45C	Threshold Bank (REACM2_THBK23)	32	R/W	0000_0000h	44.4.13/ 1704
460	Threshold Bank (REACM2_THBK24)	32	R/W	0000_0000h	44.4.13/ 1704
464	Threshold Bank (REACM2_THBK25)	32	R/W	0000_0000h	44.4.13/ 1704
468	Threshold Bank (REACM2_THBK26)	32	R/W	0000_0000h	44.4.13/ 1704
46C	Threshold Bank (REACM2_THBK27)	32	R/W	0000_0000h	44.4.13/ 1704
470	Threshold Bank (REACM2_THBK28)	32	R/W	0000_0000h	44.4.13/ 1704
474	Threshold Bank (REACM2_THBK29)	32	R/W	0000_0000h	44.4.13/ 1704
478	Threshold Bank (REACM2_THBK30)	32	R/W	0000_0000h	44.4.13/ 1704
47C	Threshold Bank (REACM2_THBK31)	32	R/W	0000_0000h	44.4.13/ 1704
600	ADC Result Maximum Limit Check (REACM2_ADCMAX)	32	R/W	0000_0000h	44.4.14/ 1705
680	Modulation Range Pulse Width (REACM2_RANGE_PWD)	32	R/W	0000_0000h	44.4.15/ 1705
6C0	Modulation Minimum Pulse Width (REACM2_MIN_PWD)	32	R/W	0000_0000h	44.4.16/ 1706
700	Modulation Control Word Bank (REACM2_MWBK0)	32	R/W	0000_0000h	44.4.17/ 1707
704	Modulation Control Word Bank (REACM2_MWBK1)	32	R/W	0000_0000h	44.4.17/ 1707
708	Modulation Control Word Bank (REACM2_MWBK2)	32	R/W	0000_0000h	44.4.17/ 1707
70C	Modulation Control Word Bank (REACM2_MWBK3)	32	R/W	0000_0000h	44.4.17/ 1707
710	Modulation Control Word Bank (REACM2_MWBK4)	32	R/W	0000_0000h	44.4.17/ 1707

Table continues on the next page...

REACM2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
714	Modulation Control Word Bank (REACM2_MWBK5)	32	R/W	0000_0000h	44.4.17/ 1707
718	Modulation Control Word Bank (REACM2_MWBK6)	32	R/W	0000_0000h	44.4.17/ 1707
71C	Modulation Control Word Bank (REACM2_MWBK7)	32	R/W	0000_0000h	44.4.17/ 1707
720	Modulation Control Word Bank (REACM2_MWBK8)	32	R/W	0000_0000h	44.4.17/ 1707
724	Modulation Control Word Bank (REACM2_MWBK9)	32	R/W	0000_0000h	44.4.17/ 1707
728	Modulation Control Word Bank (REACM2_MWBK10)	32	R/W	0000_0000h	44.4.17/ 1707
72C	Modulation Control Word Bank (REACM2_MWBK11)	32	R/W	0000_0000h	44.4.17/ 1707
730	Modulation Control Word Bank (REACM2_MWBK12)	32	R/W	0000_0000h	44.4.17/ 1707
734	Modulation Control Word Bank (REACM2_MWBK13)	32	R/W	0000_0000h	44.4.17/ 1707
738	Modulation Control Word Bank (REACM2_MWBK14)	32	R/W	0000_0000h	44.4.17/ 1707
73C	Modulation Control Word Bank (REACM2_MWBK15)	32	R/W	0000_0000h	44.4.17/ 1707
740	Modulation Control Word Bank (REACM2_MWBK16)	32	R/W	0000_0000h	44.4.17/ 1707
744	Modulation Control Word Bank (REACM2_MWBK17)	32	R/W	0000_0000h	44.4.17/ 1707
748	Modulation Control Word Bank (REACM2_MWBK18)	32	R/W	0000_0000h	44.4.17/ 1707
74C	Modulation Control Word Bank (REACM2_MWBK19)	32	R/W	0000_0000h	44.4.17/ 1707
750	Modulation Control Word Bank (REACM2_MWBK20)	32	R/W	0000_0000h	44.4.17/ 1707
754	Modulation Control Word Bank (REACM2_MWBK21)	32	R/W	0000_0000h	44.4.17/ 1707
758	Modulation Control Word Bank (REACM2_MWBK22)	32	R/W	0000_0000h	44.4.17/ 1707
75C	Modulation Control Word Bank (REACM2_MWBK23)	32	R/W	0000_0000h	44.4.17/ 1707
760	Modulation Control Word Bank (REACM2_MWBK24)	32	R/W	0000_0000h	44.4.17/ 1707
764	Modulation Control Word Bank (REACM2_MWBK25)	32	R/W	0000_0000h	44.4.17/ 1707
768	Modulation Control Word Bank (REACM2_MWBK26)	32	R/W	0000_0000h	44.4.17/ 1707

Table continues on the next page...

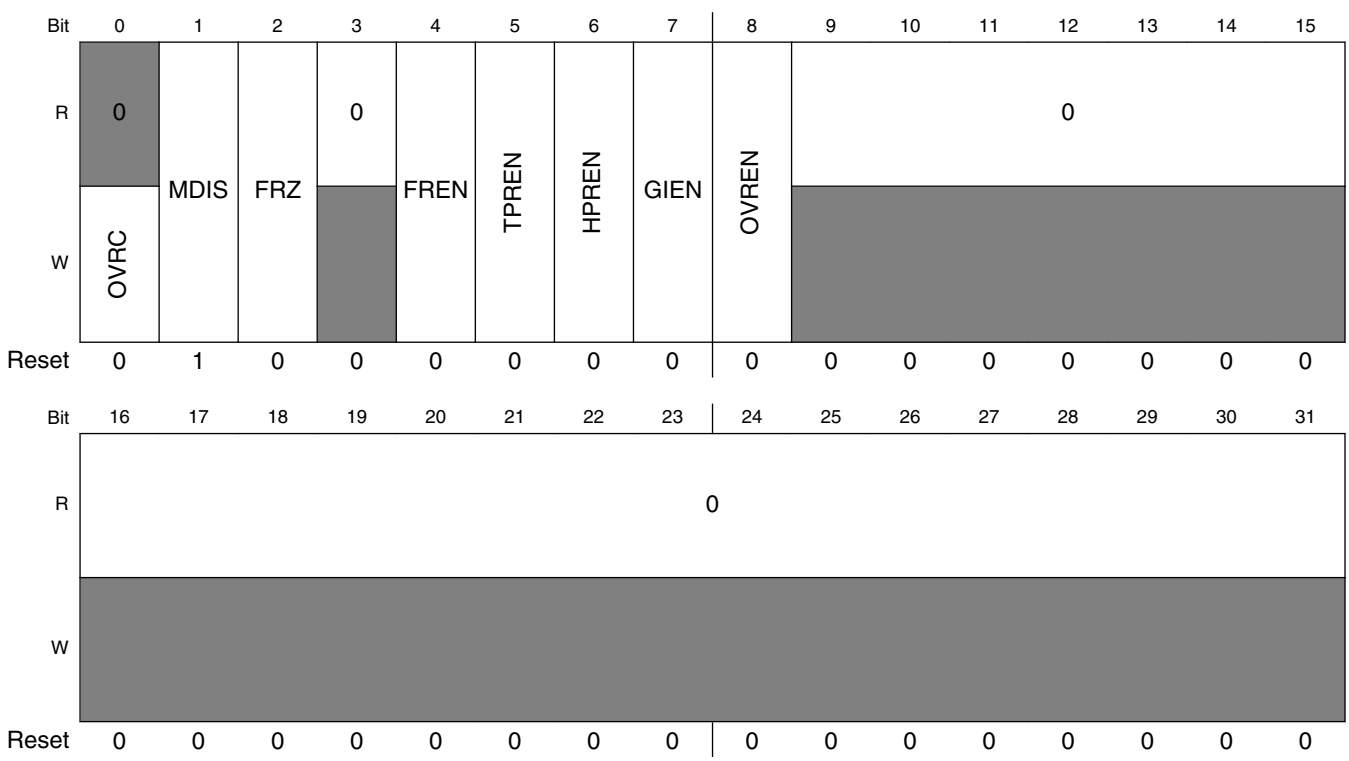
REACM2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
76C	Modulation Control Word Bank (REACM2_MWBK27)	32	R/W	0000_0000h	44.4.17/ 1707
770	Modulation Control Word Bank (REACM2_MWBK28)	32	R/W	0000_0000h	44.4.17/ 1707
774	Modulation Control Word Bank (REACM2_MWBK29)	32	R/W	0000_0000h	44.4.17/ 1707
778	Modulation Control Word Bank (REACM2_MWBK30)	32	R/W	0000_0000h	44.4.17/ 1707
77C	Modulation Control Word Bank (REACM2_MWBK31)	32	R/W	0000_0000h	44.4.17/ 1707

44.4.1 Module Configuration (REACM2_MCR)

The REACM_MCR contains the control bits to configure the general operation of the Reaction Module.

Address: 0h base + 0h offset = 0h



REACM2_MCR field descriptions

Field	Description
0 OVRC	Overrun Detection Flag Clear The OVRC clears the OVR flag if "1" is written. This bit reads always as 0x0. If a set event occurs at the same time a flag clear is done, the set event has precedence over the clear thus the flag remains set. 0 No action 1 Clears OVR bit
1 MDIS	Module Disable The MDIS bit puts the Reaction Module in low power mode. Communication through the slave-bus Interface is ignored in this mode except writes to the REAC_MCR register which are allowed, except for the FRZ and FREN bits. The global debug signal state is not changed internally while in low power mode. 0 Normal Mode 1 Low Power Mode
2 FRZ	Freeze Control The FRZ bit controls the state of the Reaction Module regarding debug operation. If both FREN and FRZ bits are set the module enters debug mode. In this mode all time bases are halted and the outputs of channels are controlled solely by software. See section Debug Mode.This bit cannot be written if MDIS bit is set. 0 Normal Mode 1 Debug Mode
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 FREN	Freeze Enable The FREN bit enables the Reaction Module to enter debug mode. The debug mode is controlled either by the FRZ bit or by a global debug signal. This bit cannot be written if MDIS bit is set. 0 Debug Mode disabled 1 Debug Mode enable
5 TPREN	Timer Prescaler Enable The TPREN bit enables the Shared Timer Prescaler in the Reaction Module. 0 Prescaler Disabled 1 Prescaler Enabled
6 HPREN	Hold Timer Prescaler Enable The HPREN bit enables the Hold Prescaler in the Reaction Module. 0 Prescaler Disabled 1 Prescaler Enabled
7 GIEN	Global Interrupt Enable The GIEN bit enables the assertion of the global interrupt request to the CPU when any of the channel interrupts is set or when the OVR flag and its corresponding enable OVREN are both set. Note that this signal does not control the individual channel interrupt request. See section Global Interrupt and Channel Interrupt for more details.

Table continues on the next page...

REACM2_MCR field descriptions (continued)

Field	Description
	0 Interrupt disabled 1 Interrupt enabled
8 OVREN	Overrun Detection Interrupt Enable The OVREN enables the OVR flag, when set, to generate a global interrupt request for the CPU. 0 Interrupt disabled 1 Interrupt enabled
9–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

44.4.2 Timer Configuration (REACM2_TCR)

The REACM_TCR Timer Configuration Register contains the prescaler settings to configure the operation of the Reaction Module Hold Timer and Shared Timers. It is mandatory to change the value of the prescalers either when the prescalers are disabled by its control enable bits HPREN and TPREN in the Configuration Register or when the counters are not being used by any channel. Note that the prescalers are completely independent and thus modifying HPRE does not affect TPRE and vice-versa.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				HPRE												0								TPRE							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

REACM2_TCR field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 HPRE	Hold Timer Prescaler The HPRE field defines the rate of the Hold Timers on each Reaction Channel. If its value is zero the prescaler is bypassed thus the Hold timer operates at the module clock rate. If HPRE = 0x01 the module operates at module clock divide by two and so forth up to HPRE = 0xFFF which defines system clock divided by 4096.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 TPRE	Timer Prescaler The TPRE field defines the rate of the Timers on the Timer bank. If its value is zero the prescaler is bypassed thus the Timer operates at the module clock rate. If TPRE = 0x01 the Timer operates at module

Table continues on the next page...

REACM2_TCR field descriptions (continued)

Field	Description
	clock divide by two and so forth up to TPRE = 0xFF which defines module clock divided by 256 as the frequency of operation for the Timer.

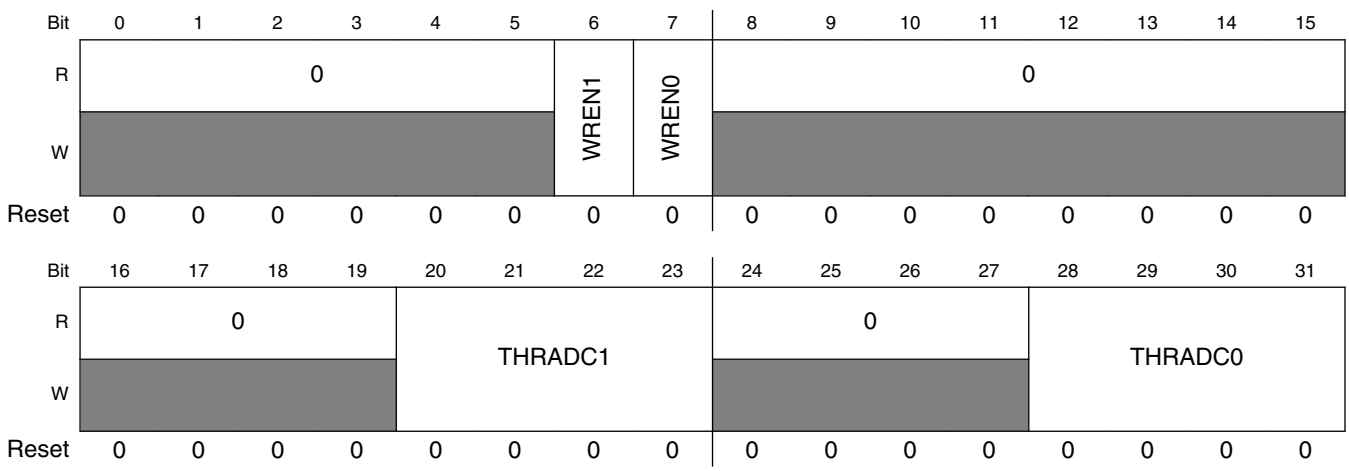
44.4.3 Threshold Router Register (REACM2_THRR)

The Reaction Module Threshold Router Register routes the ADC result to the Threshold Bank. Since a TAG is assigned for each ADC result coming from the on-chip ADC module, this TAG is specified in this register and used for the routing process. The ADC result is written to the Threshold Value Bank as soon as it is received by the Reaction Module. Note that THRADC0 and THRADC1 TAG values may also be used by the Reaction Channels. In this case the results are routed to both, the channel and the Threshold Bank.

NOTE

Due to the timing in which these parallel events occur, if the channel uses the same Threshold Bank address in which the incoming ADC result was written to, this ADC result is used for the channel modulation being executed.

Address: 0h base + 8h offset = 8h



REACM2_THRR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 WREN1	Write Enable Bit for THRADC1

Table continues on the next page...

REACM2_THRR field descriptions (continued)

Field	Description
	The WREN1 write enable bit 1 controls if the ADC result having a TAG matching to THRADC1 field will be written into address one of the Threshold Bank 0 Do not write received ADC result to Threshold bank 1 Write received ADC result to Threshold bank address one
7 WREN0	Write Enable Bit for THRADC0 The WREN0 write enable bit 0 controls if the ADC result having a TAG matching to THRADC0 field will be written into address zero of the Threshold Bank 0 Do not write received ADC result to Threshold bank 1 Write received ADC result to Threshold bank address zero
8–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 THRADC1	ADC result Router value for Threshold Bank address one The THRADC1[3:0] field controls the routing from the received ADC result to the Threshold Bank address one. Any ADC result which TAG matching THRADC1 will be routed to Threshold Bank.
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 THRADC0	ADC result Router for Threshold Bank address zero The THRADC0[3:0] field controls the routing from the received ADC result to the Threshold Bank address zero. Any ADC result which TAG matching THRADC0 will be routed to Threshold Bank.

44.4.4 ADC Sensor Input Register (REACM2_SINR)

The Reaction Module ADC Sensor Input Register is used to monitor the ADC Interface (see section ADC Interface) and allows the software to define the ADC result and TAG values for the Reaction Module. Also ADC values captured through DMA directly from the ADC interface or after filtering can be transferred to the Reaction Module by the CPU. Writes to this register may overwrite data coming from the ADC interface thus it should be used when no ADC conversion has the Reaction Module as the target. Writing to this register triggers the Reaction Channel selected by the TAG value to execute a comparison and to evaluate an eventually new value for the channel outputs.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												ADC_TAG				ADC_RESULT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

REACM2_SINR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 ADC_TAG	TAG value The ADC_TAG[3:0] represents the TAG for the ADC conversion which is used by the Reaction Module to select the Reaction Channel to execute the modulation.
16–31 ADC_RESULT	ADC conversion result value The ADC_RESULT represents the value resulting from an ADC conversion. This value is used for the Reaction Channel Modulation process or for capturing by the Threshold Bank.

44.4.5 Period Generator Configuration Register (REACM2_PCR)

The Reaction Module Period Generator Configuration Register is used to setup the period pulse generator sub-block. See section Period / Delay Generator. This configuration is necessary when running the modulation mode threshold/period.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								PERPRESC								PER															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

REACM2_PCR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 PERPRESC	Period prescaler rate value The PERPRESC field defines the rate of the prescaler to be used by the Period counter and Delay counter. If its value is 0, the rate is 1, and so forth up to PERPRESC= 0xFF which defines to divide the system clock by 256.
16–31 PER	Period time value The PER represents the value for the internal period generator counter. When it is configured to all 0, the counter is stopped. And if not all 0, the period time in clock cycles is given by multiplying (PER+1) by the prescaler value (PERPRESC+1).

44.4.6 Period Shift Delay Configuration Register (REACM2_PSCR)

The Reaction Module Period Shift Delay Configuration Register is used to configure the delay time between period pulses of two consecutive channels. See section Period / Delay Generator. The delay multiplied by the number of channels must be less than or equal to the period.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																DLY															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

REACM2_PSCR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 DLY	Delay value between two consecutive channel period pulses The DLY represents the value to configure the shift delay between the period pulses of two consecutive reaction channels (for example, CH0 and CH1). The correct time depends on the prescaler counter. Also, if DLY=0 then the period pulses of all reaction channels occur at the same time. If DLY is not all 0, then the shift delay time in clock cycles is given by multiplying DLY with the prescaler (PERPRESC+1).

44.4.7 Global Error Flag Register (REACM2_GEFR)

The Reaction Module Global Error Flag Register helps the software in the resolution of error conditions signaled by the Reaction Channels. This allows a faster service in error handling. This register mirrors the Error Flag in the Channel Configuration and Status registers.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OVR	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						EF9	EF8	EF7	EF6	EF5	EF4	EF3	EF2	EF1	EF0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

REACM2_GEFR field descriptions

Field	Description
0 OVR	Overrun Detection Flag The OVR flag is used to indicate that an overrun condition was detected at the ADC Interface. See section ADC Interface. 0 Overrun not detected 1 Overrun detected
1–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 EF9	Error Flag The EFn error flag bit indicates an error condition occurred in channel n. This bit is provided for a fast channel error resolution in the Reaction module. The condition could be any error indicated by MAXL, OCDF, SCDF, SQER, RAER or TAER error flags as described in section “REACM2 Channel Status Register (REACM2_CHSR)". The EFn bit is automatically cleared if the corresponding channel flags are all cleared.
23 EF8	Error Flag See description for EF9.
24 EF7	Error Flag See description for EF9.
25 EF6	Error Flag See description for EF9.

Table continues on the next page...

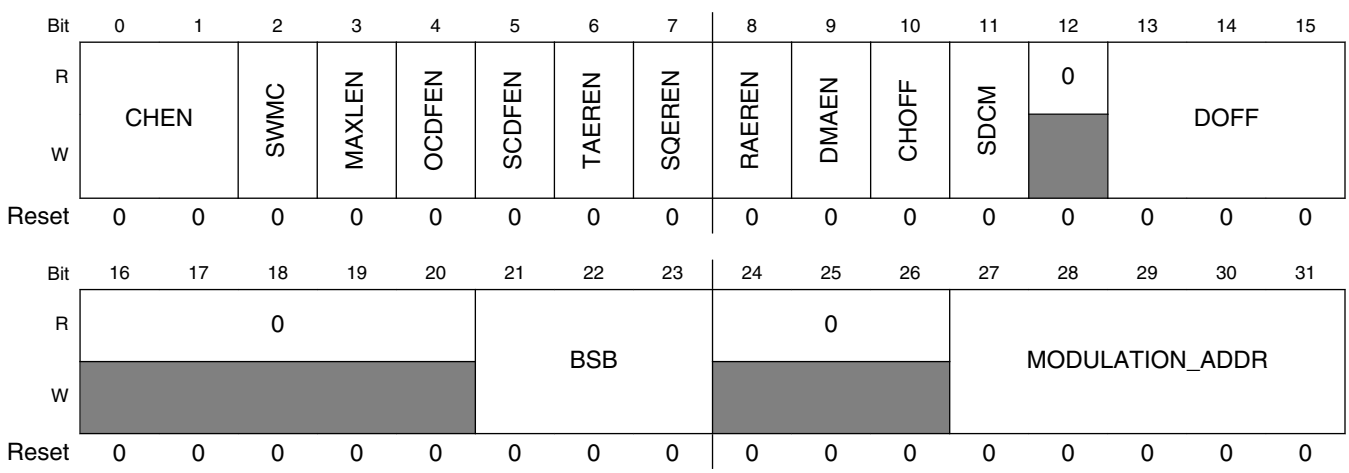
REACM2_GEFR field descriptions (continued)

Field	Description
26 EF5	Error Flag See description for EF9.
27 EF4	Error Flag See description for EF9.
28 EF3	Error Flag See description for EF9.
29 EF2	Error Flag See description for EF9.
30 EF1	Error Flag See description for EF9.
31 EF0	Error Flag See description for EF9.

44.4.8 Channel n Configuration Register (REACM2_CHCR*n*)

The Reaction Module Channel Configuration Register controls the channel behavior.

Address: 0h base + 100h offset + (16d × i), where i=0d to 9d



REACM2_CHCR*n* field descriptions

Field	Description
0–1 CHEN	Channel Enable Bits The CHEN[1:0] bits control the activation of the Reaction Channel. Once changed from disabled, CHEN = 00, to any enabled state, the channel reads the first Modulation Word and initializes itself in order to be

Table continues on the next page...

REACM2_CHCR*n* field descriptions (continued)

Field	Description
	ready to execute a modulation as soon as a timer control window is detected or SWMC=1, if software modulation control is selected by CHEN= 11. **NOTE:** If CHEN is programed with 01 or 11 enabling a channel, and the eTPU time window is already active (or SWMC=1), the Reaction Channel disregards this window and waits until the next window activation in order to start the modulation process by moving to the active state. In order to start a modulation controlled by software, it is required to first write CHEN=11 and then to write SWMC=1 only after that. 00 Channel disabled, meaning that it does not execute any modulation even if a timer window is detected or SWMC is made active. In this mode DOFF field in the Channel Configuration register defines the state of the channel outputs. 01 Channel is enabled for timer control only, meaning that as soon as a timer window is detected a modulation sequence starts. 10 Reserved 11 Channel enabled for software control only, meaning that as soon as SWMC bit is set a modulation sequence starts.
2 SWMC	Software Modulation Control bit The SWMC, Software Modulation Control bit, if set, initiates the channel modulation. It is equivalent to the assertion of an eTPU channel connected to the Reaction module. To use this functionality it is required that CHEN[1:0] = 11. 0 Channel does not perform modulation 1 Channel executes modulation
3 MAXLEN	ADC result Maximum Limit Interrupt Enable bit The MAXLEN enables the MAXL flag, when set, to generate an interrupt request for the CPU. 0 Interrupt disabled 1 Interrupt enabled
4 OCDFEN	OCDF Interrupt Enable bit The OCDFEN bit enables the OCDF bit to issue an interrupt request. 0 OCDF interrupt disabled 1 OCDF interrupt enabled
5 SCDFEN	SCDF Interrupt Enable bit The SCDFEN bit enables the SCDF bit to issue an interrupt request. 0 SCDF interrupt disabled 1 SCDF interrupt enabled
6 TAEREN	TAER Interrupt Enable bit The TAEREN bit enables the TAER bit to issue an interrupt request. 0 TAER interrupt disabled 1 TAER interrupt enabled
7 SQEREN	Modulation Word Sequence Error Interrupt Enable The SQEREN bit enables the SQER flag to generate an interrupt request.

Table continues on the next page...

REACM2_CHCR*n* field descriptions (continued)

Field	Description
	0 SQER interrupt disabled 1 SQER interrupt enabled
8 RAEREN	Resource Allocation Error Interrupt Enable The RAEREN bit enables the RAER flag to generate an interrupt request. 0 RAER interrupt disabled 1 RAER interrupt enabled
9 DMAEN	Direct Memory Access Enable bit The DMAEN bit enables a DMA request by the channel when an Modulation Word sequence advance occurs. 0 Disables channel DMA request 1 Enables channel DMA request
10 CHOFF	Output Disable bit The CHOFF bit allows disabling the output pin. DOFF[2:0] field is used as disable state for the channel outputs. The output pins will set to DOFF value if CHOFF is asserted. Note that disabling the channel through CHOFF does not disable the channel operation. Only the channel outputs are forced to DOFF state. 0 Output Disable Disabled 1 Output Disable Enabled
11 SDCM	Short Detection Counting Mode The SDCM bit configures the short detector to count one or two short detection events to set the SCDF field. 0 SCDF is set if two short detection events occur in a row 1 SCDF is set if a short detection occurs
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 DOFF	Drive Off Control field The DOFF[2:0] field defines the Reaction Channel output disabled state. This condition is achieved either when CHEN= 00, or after CHEN activation, or when Output is disabled by CHOFF in the Channel Configuration register. • DOFF[0] bit defines CHn_a output pin value • DOFF[1] bit defines CHn_b output pin value • DOFF[2] bit defines CHn_c output pin value **NOTE:** It is possible to control the channel outputs directly through the software by writing to the DOFF bits. In this case the channel should be in the disabled state, CHEN=00. **NOTE:** If the DOFF value is changed just after the channel is enabled, it is not assured the new DOFF value is immediately used in the channel output.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–23 BSB	Bank Support Bits

Table continues on the next page...

REACM2_CHCR*n* field descriptions (continued)

Field	Description
	The BSB[2:0] provides control for a banked mode operation of the Reaction Module. Each bit in this field controls the channel outputs chn_c, chn_b and chn_a respectively. When asserted the channel output implements a sharing function with the corresponding output of the subsequent channel.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 MODULATION_ADDR	Address for Modulation Control Bank The MODULATION_ADDR field has the address of the Modulation Word in the Modulation Control Bank. This address is used as a base address for the first modulation word. The Reaction Channel can access subsequent words in the Modulation Control Bank by incrementing the Modulation Address field. Note that this field is not modified by the Reaction Channel in this process. **NOTE:** The number of subsequent words is limited to 15. Therefore the channel can handle a sequence with as many as 16 modulation words. **NOTE:** If a sequence advance is requested in the 16th modulation word, the channel returns back and accesses the first modulation word that is indicated by the address field MODULATION_ADDR on the REACM2_CHCR*n* register.

44.4.9 Channel n Status Register (REACM2_CHSR*n*)

The Reaction Module Channel Status Register provides access to the channel flags and flag clear bits. It also provides access to the current values of the channel output being driven and the Modulation Word being accessed by the channel.

Address: 0h base + 104h offset + (16d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0		MODACT	MAXL	OCDF	SCDF	TAER	SQER	RAER	CHOUT			0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			0	0	0	0	0	0	0	MODULATION_POINTER					
W				MAXLC	OCDFC	SCDFC	TAERC	SQERC	RAERC							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

REACM2_CHSR*n* field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 MODACT	Modulation Active Flag The MODACT flag indicates that the channel is enabled and active thus executing a modulation cycle. 0 Channel is not active 1 Channel is active

Table continues on the next page...

REACM2_CHSR*n* field descriptions (continued)

Field	Description
3 MAXL	ADC Maximum Limit Detection Flag The MAXL flag indicates that the ADC result which TAG is addressing this channel achieved or passed the maximum allowed limit specified in the ADCMAX register. This flag is set when the channel is in the active state (see section Channel Modes) or in debug mode. This bit is not set in the off time of the Threshold/Hold-off modulation mode. 0 Normal operation 1 ADC Maximum Limit Detected
4 OCDF	Open Circuit Detection Flag The OCDF flag indicates that an open circuit was detected in the channel load using the PWM monitored modulation, as described in Monitored Modulation. This Flag is set only if the channel is in the active state. 0 Normal operation 1 Open Circuit Detected
5 SCDF	Short Circuit Detection Flag The SCDF flag indicates that a short circuit was detected in the channel load using the PWM monitored modulation, as described in Monitored Modulation. This flag is set only if the channels are in the active state. 0 Normal Operation 1 Short Circuit Detected
6 TAER	Timer Allocation Error The Timer Allocation Error bit indicates that the channel tried to allocate a timer counter in the Shared Timer Bank without success. This situation is an indication that the Timer resources available in the module are not sufficient to execute the required functionality. This error indication is used during software development and should not occur during normal use of the module since it may result in incorrect operation. This flag is set only if the channel is in the active state. 0 No error occurred during timer allocation. 1 Error occurred during timer allocation
7 SQER	Modulation Word Sequence Error Flag The SQER bit indicates a Modulation Sequence error occurred, meaning that the time window which defines a modulation cycle ended in a premature modulation phase. The correct modulation phase for the time window to close is when SM field in the Modulation Word indicates no advance, or SM= 00. This Flag is set only if the channel is in the active state. 0 Modulation Sequence Error did not occur 1 Modulation Sequence Error occurred
8 RAER	Resource Allocation Error Flag The RAER bit indicates that a resource allocation error occurred. The possible allocation errors are: Modulation Control Word address is out of available range (including if the MODULATION_POINTER increments to an inexistent MCW address or wraps to 0x0), Threshold Value Bank address is out of available range, Hold Timer Bank address is out of available range, Shared Timer Bank address is out of available range, Channel Input Router points to an inexistent eTPU channel, and when the Hold timer is select for both modulation and sequence advance at the same time (i.e, SM=10 and MM=01). Note that the amount of hardware resources is configuration dependent thus may vary according to module

Table continues on the next page...

REACM2_CHSR*n* field descriptions (continued)

Field	Description
	parameterization in the SoC. This flag is intended to help on the debug of the Reaction Module during software development and can be set only if the channel is in the enabled state. **NOTE:** The condition that sets the RAER bit must be resolved prior to clear the bit, otherwise the bit can be set again. 0 No allocation error occurred 1 Allocation error occurred
9–11 CHOUT	Channel Output Monitoring bits The CHOUT[2:0] Channel Output Monitoring bits provides for the software the ability to monitor the output provided by the channel. This data is not buffered thus represents the channel output at the time the CPU read access is done. CHOUT[0] corresponds to the chn_a output, CHOUT[1] corresponds to the chn_b output and CHOUT[2] corresponds to the chn_c output pin. These bits are available independent of the channel mode or state.
12–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 MAXLC	ADC Maximum Limit Flag Clear The MAXLC clears the MAXL flag if "1" is written. This bit reads always as 0x0. If a set event occurs at the same time a flag clear is done, the set event has precedence over the clear thus the flag remains set. 0 No action 1 Clears MAXL bit
20 OCDFC	Open Circuit Detection Flag Clear The OCDFC clears the OCDF flag if "1" is written. This bit reads always as 0x0. If a set event occurs at the same time a flag clear is done, the set event has precedence over the clear thus the flag remains set. 0 No action 1 Clears OCDF bit
21 SCDFC	Short Circuit Detection Flag clear The SCDFC bit clears the SCDF flag if "1" is written. This bit is self negated thus read always as ‘0’. If a set event occurs at the same time a flag clear is done, the set event has precedence over the clear thus the flag remains set. 0 No action 1 Clears SCDF
22 TAERC	TAER Clear bit The TAERC bit clears the TAER bit if "1" is written. This bit is self-negated thus reads always as ‘0’. If a set event occurs at the same time a flag clear is done, the set event has precedence over the clear thus the flag remains set. 0 No action 1 Clears TAER
23 SQERC	Modulation Word Sequence Error Flag clear The SQERC bit clears the SQER flag in the Channel Status register. 0 SQER flag is not cleared 1 SQER flag is cleared

Table continues on the next page...

REACM2_CHSR*n* field descriptions (continued)

Field	Description
24 RAERC	Resource Allocation Error Flag clear The RAERC bit clears the RAER flag in the Channel Status register. 0 RAER flag is not cleared 1 RAER flag is cleared
25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 MODULATION_ POINTER	Modulation bank address generated by the channel The MODULATION_POINTER[5:0] gives to the software the information of the current address being generated by the channel to access the Modulation Word. Note that the address increments from a base address stored in the channel, MODULATION_ADDR[5:0]. This register is not buffered thus represents the current address being generated.

44.4.10 Channel n Router Register (REACM2_CHRR*n*)

The Reaction Module Channel Router Register controls the channel connection to external timers, such as eTPU and eMIOS and ADC result data. The channels have access to any 16 timer inputs and any ADC sample tag from 0 to 15. Note that this architecture allows several Reaction Channels to point to the same timer channel or to the same ADC result.

Address: 0h base + 108h offset + (16d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												ADCR				0												CHIR			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

REACM2_CHRR*n* field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 ADCR	ADC result router field The ADCR[3:0] field selects which ADC result is used by the Reaction Channel for the modulation. The TAG[3:0] received along with the ADC result is used for comparison with the ADCR[3:0] field in order to define if the received result is used by the Reaction Channel.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 CHIR	Channel Input router field The CHIR field selects which eTPU channel is used by the Reaction Channel for the modulation.

44.4.11 Shared Timer Bank (REACM2_STBK*n*)

The Reaction Modulation Shared Timer Bank is a set of registers which define the values used by the Reaction Module Timer. The timer values are programmed by the Host CPU during the configuration of the Reaction Module. Modulation Word accessed by the Reaction Channel contains the address of a specific Timer value stored in the Shared Timer bank. By selecting a Timer value the Reaction Channel also selects and enables a counter. When this counter reaches the selected Timer value a time-out indication is generated for the Reaction Channel that initiated the counter. This event is used for example to indicate that a next Modulation Word should be used for the modulation.

Address: 0h base + 300h offset + (4d × i), where i=0d to 4d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																SHARED_TIMER															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

REACM2_STBK*n* field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 SHARED_TIMER	Timer Value The SHARED_TIMER value is one element of the Timer Register Bank. Up to 5timer values can be stored within the Timer Bank. **NOTE:** When using the shared timer for sequence advance, the counted time (considering prescaler) must be greater than 64 clock cycles.

44.4.12 Hold Timer Bank (REACM2_HOTBK*n*)

The Reaction Module Hold Timer Bank is a set of registers that defines the values used by the Reaction Channels to measure hold time on certain modulation schemes. The timer values are programmed by software and addressed by the Reaction Channel based on the data read from a Modulation Word.

Address: 0h base + 380h offset + (4d × i), where i=0d to 4d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																				HOLD_TIM											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

REACM2_HOTBK*n* field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 HOLD_TIM	Hold Timer Value The HOLD_TIM value is one element of the Hold Timer Register Bank. Up to 5timer values can be stored within the Hold Timer Bank. **NOTE:** When using the hold timer for sequence advance, the counted time (considering prescaler) must greater than 64 clock cycles.

44.4.13 Threshold Bank (REACM2_THBK*n*)

The Reaction Module Threshold Bank holds the value to be used for comparison against results received from the ADC. Based on that comparison the Reaction Channel decides the Channel output value to be either HOD or LOD.

Address: 0h base + 400h offset + (4d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																THRESHOLD_VALUE																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

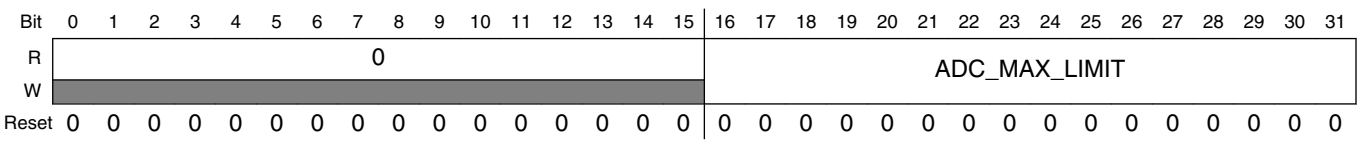
REACM2_THBK*n* field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 THRESHOLD_ VALUE	Threshold Value The THRESHOLD_VALUE unsigned value is one element of the Threshold register used for a threshold modulation. Up to 32timer values can be stored within the Threshold Bank.

44.4.14 ADC Result Maximum Limit Check (REACM2_ADCMAX)

The Reaction Module ADC Maximum Limit Check register holds the maximum expected value of the ADC result. If a value greater than or equal to ADC_MAX_LIMIT is received the MAXL bit in the corresponding Channel Status Register Error is asserted.

Address: 0h base + 600h offset = 600h



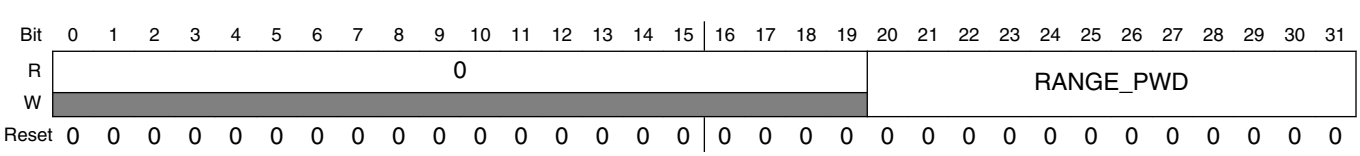
REACM2_ADCMAX field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 ADC_MAX_ LIMIT	Maximum Limit allowed for the ADC result The ADC_MAX_LIMIT[15:0] value indicates the maximum expected value for the ADC result. The MAXL bit in the corresponding channel Status register is set in case a greater or equal value is received from the ADC. If set to zero no limit checking is performed, thus MAXL bit will not be set anyway. **NOTE:** ADC results are always considered unsigned unless specific note states the contrary. Since ADC received results can be 14-bit values a two bit sign extension must be performed before any comparison is executed.

44.4.15 Modulation Range Pulse Width (REACM2_RANGE_PWD)

The Reaction Module Range PulseWidth Detection Check register provides the value used to check if the PWM pulse width generated during the modulation process is larger than a maximum pulse width specified by (MIN_PWD + RANGE_PWD). The checking is performed by the channel logic during appropriate time intervals. See section Monitored Modulation. This function is implemented by sharing the use of Reaction Channel Hold counter, thus if the Hold timer is used by the channel sequence mode, SM in the Channel Configuration register, this checking function is not active.

Address: 0h base + 680h offset = 680h



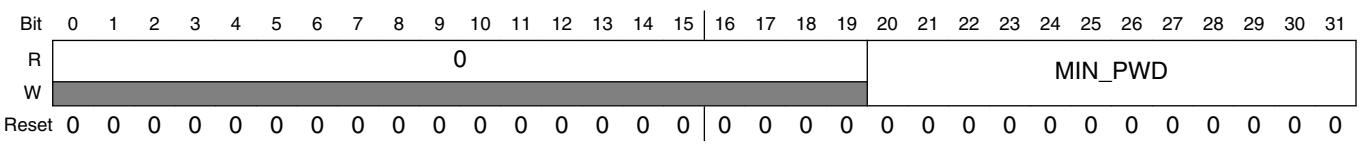
REACM2_RANGE_PWD field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 RANGE_PWD	Range Pulse Width The RANGE_PWD[11:0] field defines the maximum pulse width allowed by the channel PWM generation. The checking is performed by the channel internal Hold Timer at appropriate times in the PWM modulation process. The maximum pulse is defined as (RANGE_PWD + MIN_PWD). See “REACM2 Minimum Pulse Width Check (REACM2_MIN_PWD)" for MIN_PWD description. If RANGE_PWD = 0x00 then no maximum pulse width checking is performed. **NOTE:** The RANGE_PWD value should be calculated considering the prescaler settings used for the HOLD counter. For a programmed (RANGE_PWD + MIN_PWD) value, a pulse narrower than or equal to (RANGE_PWD + MIN_PWD + 1) does not set the OCDF flag.

44.4.16 Modulation Minimum Pulse Width (REACM2_MIN_PWD)

The Reaction Module Minimum PulseWidth Check register provides the value used to check if the pulse width generated during the PWM modulation process on the channel outputs is shorter than a minimum pulse width specified by this register. The checking is performed by using the channel internal Hold timer during appropriate times on the channel operation when this counter is not being used for the hold modulation cycle.

Address: 0h base + 6C0h offset = 6C0h



REACM2_MIN_PWD field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 MIN_PWD	Minimum Pulse Width The MIN_PWD[11:0] field defines the minimum pulse width allowed by the channel PWM generation. The checking is performed by the channel hold timer. If MIN_PWD[11:0] = 0x00 then no checking is done even for maximum pulse width or minimum pulse width. **NOTE:** The MIN_PWD value should be calculated considering the prescaler settings used for the HOLD counter. For a programmed MIN_PWD value, a pulse wider than (MIN_PWD+1) does not set the SCDF flag.

44.4.17 Modulation Control Word Bank (REACM2_MWBK*n*)

The Reaction Module Control Word Bank is a set of registers that controls the Reaction Channel Modulation scheme. These registers are programmed by software and read by the Reaction Channels. All the information required by the Reaction Channels to perform a modulation is stored in these words. All channels have access to the same words thus sharing of Modulation Words among channels is possible in this architecture.

There are 32 modulation control words in this device.

Address: 0h base + 700h offset + (4d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LOOP	IOSS	DMA	MM		0	SM		0	HOD			0	LOD		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		THRESPT					0	STPT			0		HDOFFTPT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

REACM2_MWBK*n* field descriptions

Field	Description
0 LOOP	Loop Control Bit The LOOP Control bit indicates that the next Modulation Control Word accessed by the Reaction Channel when a Modulation Word address increment event occurs should be the one indicated by the Channel MODULATION_ADDR[5:0] register with no offset. 0 Increment to the next modulation word 1 Loop back to initial channel modulation word
1 IOSS	Initial Output State Selection The IOSS bit defines the state of the channel output pins when a modulation phase is started by the Reaction Channel. This bit is important in the Modulation Word first access. This first access occurs after the channel is enabled by CHEN bit field in the Channel Configuration register, or when the modulation phase advances, e.g. due to a timer event. This bit also defines the modulation initial behavior for some modulation modes. See Modulation Mode MM[1:0] field for more details. 0 LOD[2:0] is initially used for chn_c, chn_b and chn_a respectively 1 HOD[2:0] is initially used for chn_c, chn_b and chn_a respectively
2 DMA	DMA bit

Table continues on the next page...

REACM2_MWBK*n* field descriptions (continued)

Field	Description
3–4 MM	Modulation Mode The MM[1:0] Modulation Mode field indicates the type of the modulation that is executed by the channel while the IOSS bit determines the modulation mode initial state. The initial behavior of the modulation modes (or the threshold comparison level performed in threshold/Period modulation) is selected by the IOSS bit. 00 Modulation Type: threshold/threshold Modulation occurs between two threshold levels defined by THRESPT upper level threshold pointer and THRESPT +1 which corresponds to a lower level threshold pointer. IOSS selects the initial output state. Output ON state is defined by HOD (IOSS=1) and Output OFF state is defined by LOD (IOSS=0). 01 Modulation Type: threshold/Hold-off • IOSS="0": Modulation starts with channel outputs set to LOD for an amount of time defined by the hold timer. After that outputs are set to HOD until an upper threshold defined by THRESHPT is reached. • IOSS="1": Modulation starts with channel outputs set to HOD until an upper threshold defined by THRESHPT is reached. After that level is achieved outputs are set to LOD for an amount of time defined by the hold timer. 10 Modulation Type: Threshold/Hold-on • IOSS="0": Modulation starts with channel outputs set to LOD until a lower threshold defined by THRESHPT is reached. After that level is achieved outputs are set to HOD for an amount of time defined by the hold timer. • IOSS="1": Modulation starts with channel outputs set to HOD for an amount of time defined by the hold timer. After that outputs are set to LOD until a low threshold is reached. 11 • IOSS="0": Modulation Type: threshold-low/Period. In this modulation mode channel outputs starts in LOD and a low threshold comparison is performed. When the threshold level defined by THRESPT is achieved, the channel outputs state changes to HOD until the modulation period expires. • IOSS="1": Modulation Type: threshold-high/Period. In this modulation mode channel outputs starts in HOD and a high threshold comparison is performed. When the threshold level defined by THRESPT is achieved, the channel outputs state changes to LOD until the modulation period expires.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 SM	Sequencer Mode The SM[1:0] field defines the event that makes the channel address the next modulation word. This event can be a timer event or a threshold level event. Note that the channel does not necessarily increments the Modulation Word address thus the same Modulation Word is executed again in the new modulation sequence. **NOTE:** For time-out event selections, it is required the related time should be greater than 64 clock cycles. 00 Current modulation word is used, no advance is performed on any event. No timer is activated by this modulation word. 01 Advance to the next modulation word when a Timer time-out event is detected. The Timer is activated when the modulation word is accessed by the channel.

Table continues on the next page...

REACM2_MWBK*n* field descriptions (continued)

Field	Description
	10 Advance to the next modulation word when a Hold Timer time-out event is detected. The Hold timer is activated when the modulation word is accessed by the channel. This is only for MM=00 (THR-THR). 11 Advance to the next modulation word when a threshold level is achieved (greater or equal). The threshold level is defined by the THRESPT pointer.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–11 HOD	High Output Drive The HOD[2:0] field defines the values driven on the chn_c, chn_b and chn_a channel output pins, respectively, when the channel is at ON state.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 LOD	Low Output Drive The LOD[2:0] field defines the values driven on the chn_c, chn_b and chn_a channel output pins, respectively, when the channel is at OFF state.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–22 THRESPT	Threshold Pointer The THRESPT Threshold Pointer is the address of the Threshold Bank that holds values to be used for ADC result comparisons in the modulation process. This pointer has the resolution for a 16-bit data stored in the register described in section “REACM Threshold Bank Register (REACM_THBK)".
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–26 STPT	Shared Timer Pointer The STPT Shared Timer Pointer field is the pointer for the Shared Timer Bank. The Shared Timer is used as timer sequencer defining the advance of Modulation Words.
27–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 HDOFFTPT	Hold on/off Pointer The HDOFFPT field is the address of the Hold timer value in the Hold Timer Bank that is used in the modulation cycle. Note that there are modulation sequences that does not require hold on/off measurement such as threshold/threshold modulations.

44.5 Functional Description

The following sections describe the Reaction Module functionality.

The Reaction Module is designed to allow a closed feedback loop control over the driver load currents. The load can be an injector for direct injection system or an electromagnetic actuator for robotized transmission. In both cases a solenoid is expected to be the actual load. The module architecture is based on shared resources sub-modules

that can be used by all 10 Reaction Channels. Please refer to Figure 44-4 for details about the module architecture. The Reaction Module is composed of the following internal sub-modules:

- Reaction Channel
- Modulation Control Word Bank
- Shared Timer Bank
- Hold Time Bank
- Threshold Bank
- Period Pulse Generator

The following sections describe each one of these sub-modules and their combined operation.

44.5.1 Reaction Channel

The Reaction Channel is the core of the Reaction Module. Each channel controls three output pins and it is composed of a Control FSM which receives parameters to generate a modulated waveform as well as the timer control window usually provided by eTPU. The ADC interface sub-module indicates to the channel that a new ADC result is available. If *activated* by the eTPU channel, the Reaction Channel reads the Modulation Word which provides addresses for the Threshold Value bank, Shared Timer Bank and Hold Time Bank. The date provided by the Threshold Bank is compared with the incoming ADC result. Based on that comparison the Reaction Channel state machine selects the values for the output pin registers. The modulation only occurs if *activated* by the timer control signal, which can be generated by eTPU, or can be controlled by software by writing to the SWMC, software modulation control bit, in the Channel Configuration register. The MM and SM fields in the Modulation Word provide the Modulation Mode and Sequencer Mode control respectively. The following figure describes the internal architecture of the Reaction Channel and its interconnection with other sub-modules.

NOTE

The indexes corresponding to the register bit widths in the figure below will vary depending on the chip implementation. See the register descriptions for the correct bit width in your chip.

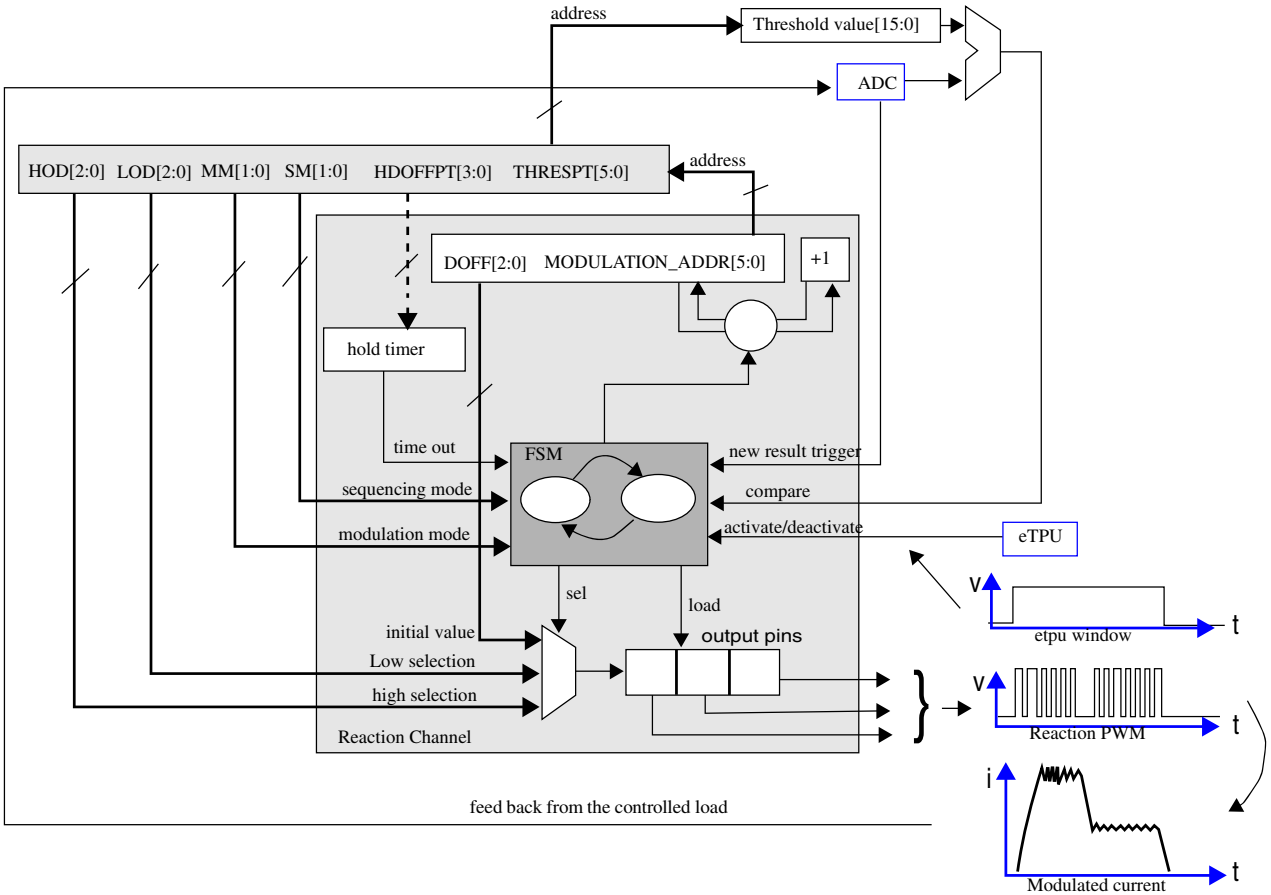


Figure 44-126. Reaction Channel Architecture Simplified Diagram

The addresses stored in the Modulation Word are pointers to timers and threshold banks. HOD and LOD provide the 3-bit field which controls the states, HIGH and LOW, respectively for the Reaction Channel output pins. The Reaction Channel FSM selects the values for the output pins as well as the load signal used to update those values. The modulation executed by the Reaction Channel is based on the feedback loop provided by the ADC results. These results are routed to one side of a comparator while the other comparator input receives the selected Threshold value from the threshold bank. The comparison result indicates for the Reaction Channel FSM if the output pin value should be defined either by HOD or by LOD. As a result a PWM is generated on the output pins. Note that since there are three independent outputs the PWM signal can be generated on any output or even on several outputs at the same time. Usually a PWM signal is generated in one of the outputs only, while the others outputs are used to control power supply switching, for example.

The preceding figure shows three waveforms at the right side of the figure. On the top there is the control signal from eTPU which defines the boundaries of the modulated waveform. In the middle there is the PWM waveform which is used to drive the power ON and OFF at the load, thus generating the third waveform which represents the current passing through the load.

NOTE

The modulation word sequence is limited to 16 words because the address value to the modulation control word bank can vary only from the configured initial value up to a maximum value given by (MODULATION_ADDR + 15).

44.5.2 Modulation Control Words Bank

The Modulation Control Word provides information about the modulation type to be performed and the addresses for several address banks. REACM2_MWBKn describes the Modulation Word register and Figure 44-127 describes the interfaces of this sub-module with some other sub-modules in the Reaction Module. The information stored in the Modulation Word are:

- Modulation control parameters for the Reaction Channel
- Threshold Value Register Bank address
- Hold time bank address
- Shared Timer bank address
- DMA support

All channels share the information stored in the Modulation Control Word Bank, which provides a size-effective implementation avoiding the duplication of information and allowing flexible implementation. The sharing of modulation control words allows several channels to execute the same modulation sequence.

The Modulation Control is designed to be used by all reaction channels as a centralized resource. However, only one channel is able to access the Modulation bank at a given time. Therefore, there is a priority in the selection of the channel that will have the access granted, but note that this condition does not occur too often since ADC results are provided for one channel at a time.

An arbiter in the Modulation Control Word bank selects one of the channels which are requesting access to a modulation word. The priority criteria is fixed and based on the channel number, considering Channel 0 the highest priority channel. The channel selected by the arbiter receives an acknowledge signal which indicates that channel was selected and therefore can access the modulation word.

Note

In order to avoid an initial delay when processing a timer window start event, the channel performs an speculative read operation of the first modulation word when it is enabled (CHEN is configured). Therefore, when the modulation cycle is triggered by the timer window start event, all needed information for the modulation is already stored inside the channel.

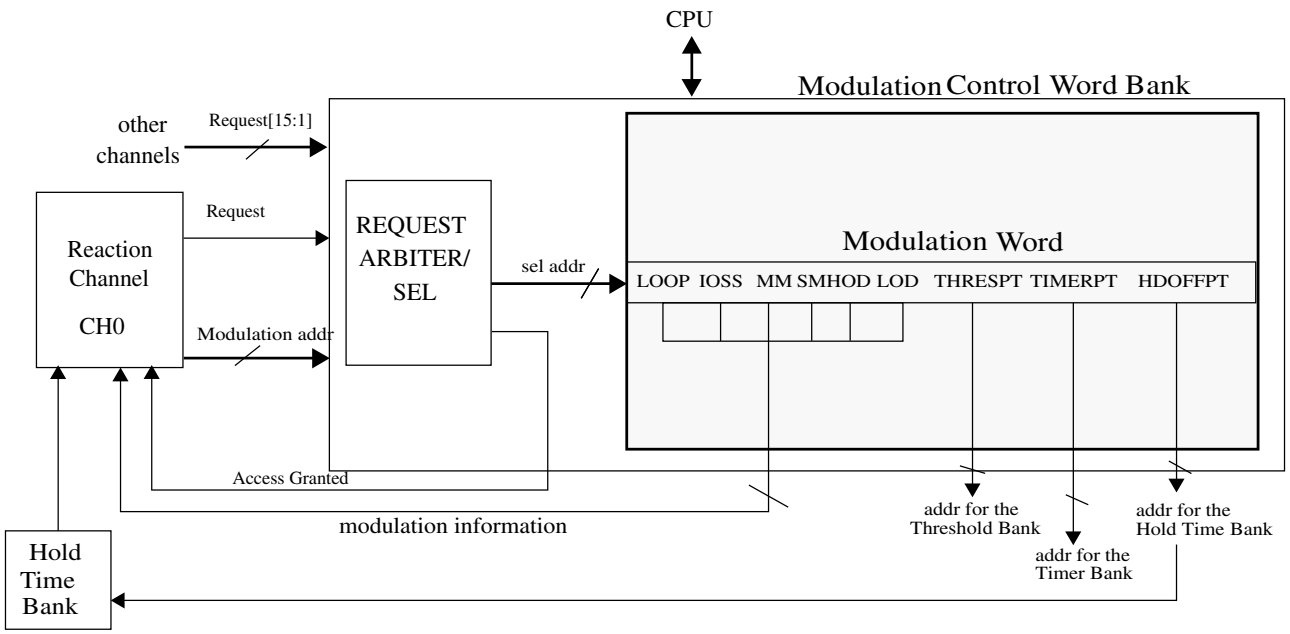


Figure 44-127. Modulation Control Word Bank Interfaces

44.5.3 Shared Timer Bank

The Shared Timer bank is an innovative concept of dynamic timer allocation. Since the number of timers can be smaller than the number of channels in the Reaction Module, there is a possibility that all timers are allocated at a certain time. This architecture is based on the low probability of such scenario since the timer allocation is a sporadic event. The timers in the Shared Timer Bank are usually in IDLE state until they are allocated by a channel. Any timer can be allocated as soon as it is in IDLE. The shared timers work in conjunction with a timer bank which stores values to be used by the timing measurement.

The Shared Timer Bank block is composed of two sub-modules:

1. Four 16-bit counters.

2. A bank with five selectable 16-bit time values.

If there is an attempt to allocate more than existing timers then an error flag TAER (see Figure 44-134) is set and no timer is allocated by the requesting channel. As a general guideline the system should be dimensioned in such a way that the timer allocation is always possible.

Note

In case of an allocation error the channel forces DOFF to its output pin preventing any damage to occur to the actuator being controlled. This state will not change until TAER bit is cleared.

During the timer allocation the channel also provides the TIMERPT pointer which selects a timing value. The value pointed by TIMERPT is loaded into one of the 4 counters which counts down until reaching zero. At this time a time-out indication is sent to the requesting channel and the timer is deallocated, moving back to IDLE state.

In case of several timer activation requests being issued at the same time, the logic in the Timer Bank will prioritize giving higher priority to the channel with lower number, thus channel zero has higher priority than the others. No flags are set in case several requests are issued simultaneously unless there are more requests than the number of available timers. In this case the TAER error flag is set in the requesting channel status register. The following figure presents a block diagram of the Timer Bank.

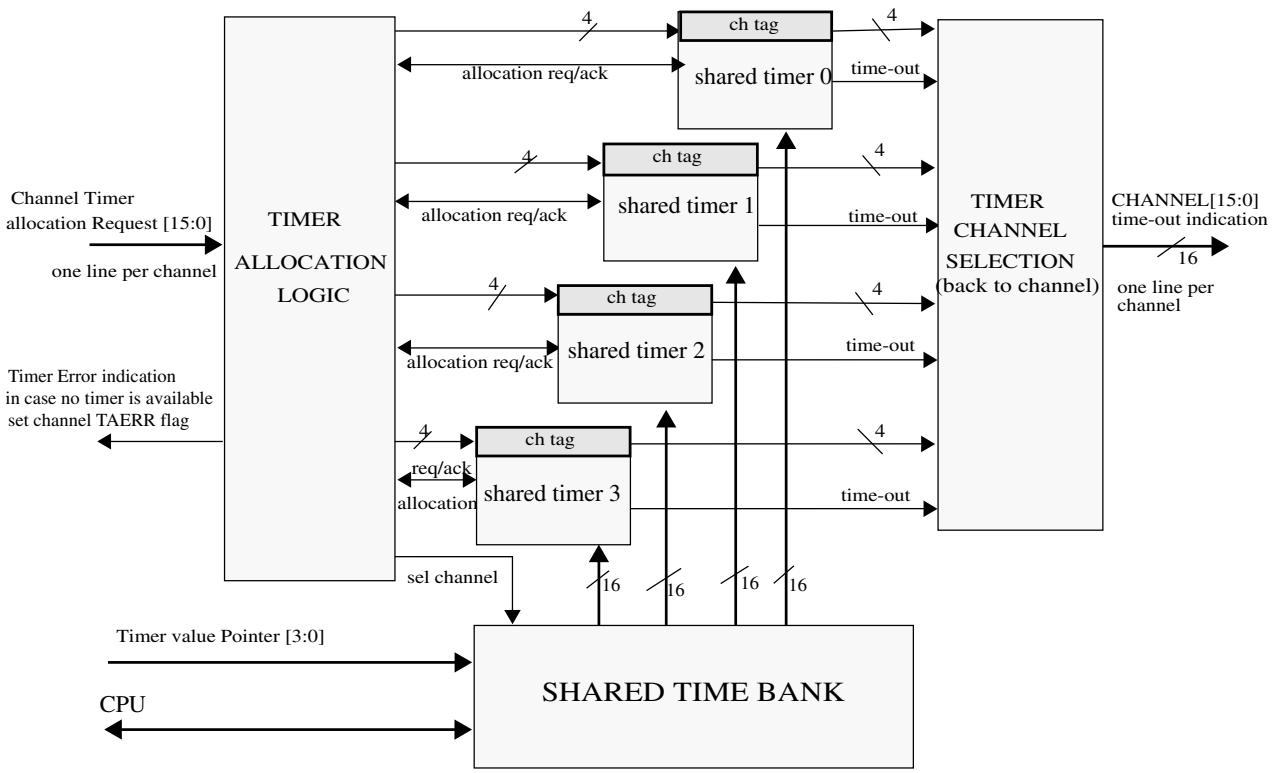


Figure 44-128. Shared Timer Bank Block Diagram with 4 Timers

44.5.4 Hold Time Bank

The Hold Time Bank stores time values that are used by the hold-on/off periods during the modulation process. This bank is shared among all channels and is addressed based on the HDOFFPT pointer in the Modulation Control Word. The following figure shows a block diagram of this bank and its interconnections.

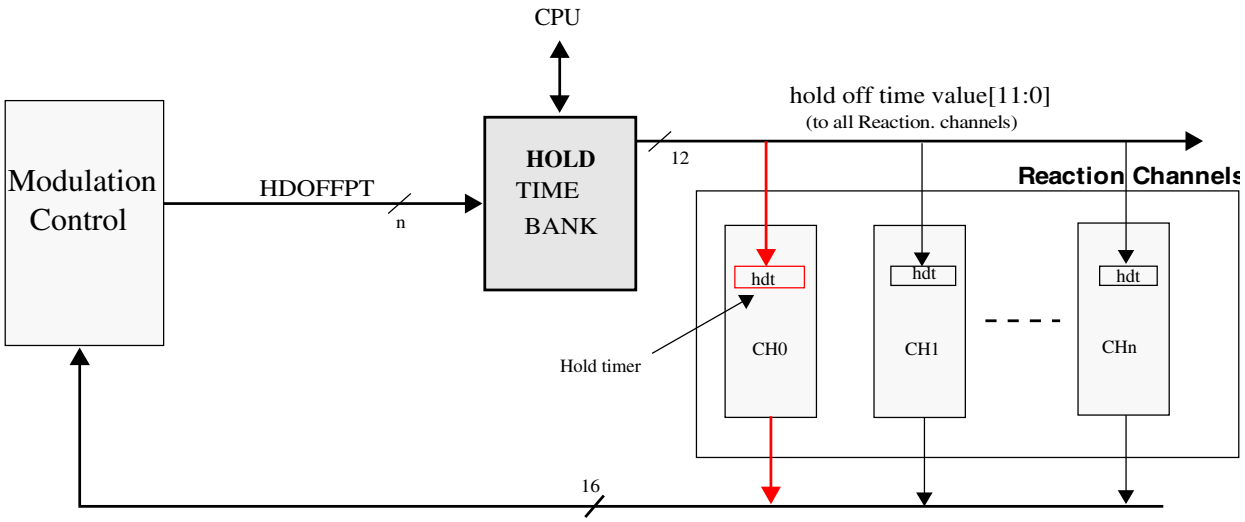


Figure 44-129. Hold Time Bank Block Diagram

The Modulation Control Word generates the address for the Hold Time bank which then generates the hold-on/off time value for the channels. The channel that addressed the Modulation Word captures the 12-bit value storing it in the channel Hold timer. In The preceding figure channel CH0 is requesting access to the Modulation Word thus receiving hold-on/off value from the Hold Time Bank. Each Reaction Channel has its own internal hold timer.

44.5.5 Threshold Bank and Comparator

The Threshold Bank and Comparator contains the threshold levels to be compared against the ADC results. This sub-module also performs the comparison between the ADC result and the selected threshold value.

The Threshold bank can be programmed with up-to 32 threshold values.

An ADC result received from the on-chip ADC is connected to the ADC router and then to the comparator (see the following figure). This result stays at the comparator input until the Reaction Channel and Modulation Control Word selects the address of the Threshold value. The THRESPT field in the Modulation Control Word sub-module is used as the address for the Threshold Bank. Note that if Threshold-Threshold modulation mode is used two comparisons need to be executed with two different Threshold values. After the first comparison is made the THRESPT pointer is incremented and a new comparison is done:

- First comparison: COMP = ADC_DATA >= THRESHOLD_VALUE[THRESPT]
- Second comparison: COMP = ADC_DATA >= THRESHOLD_VALUE[THRESPT +1]

The COMP, comparison result, is routed to the channel selected by the received TAG. Once receiving the comparison result the channel takes the appropriate actions in order to executed the modulation mode as defined by the Modulation Control Word.

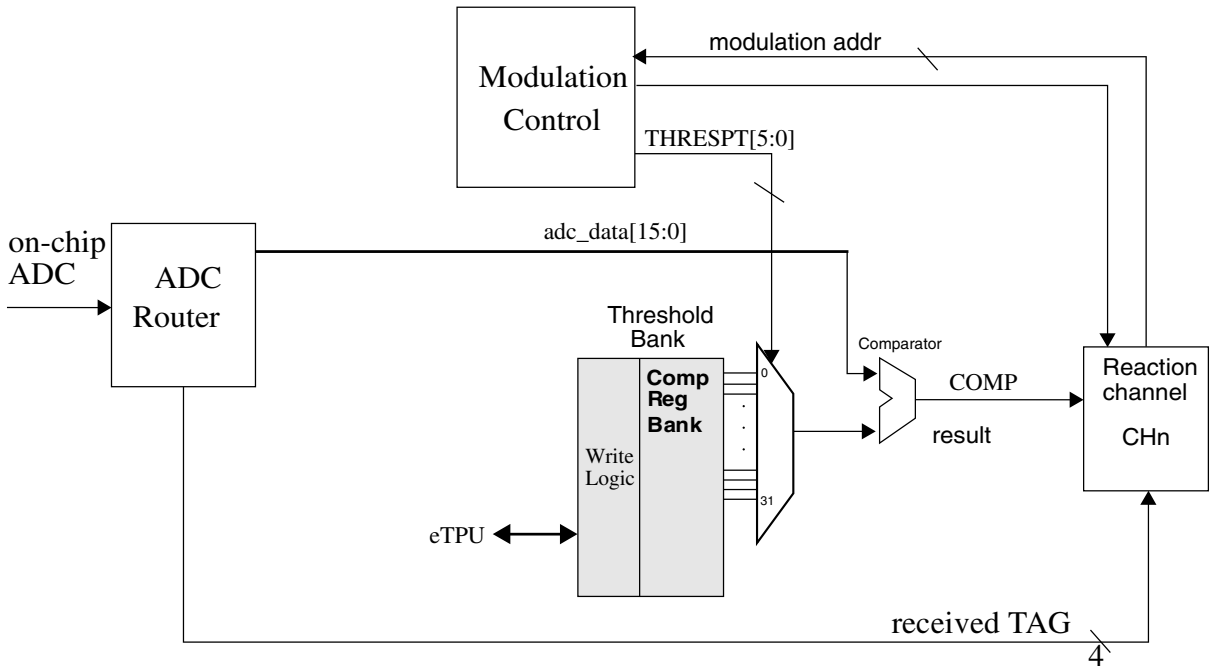


Figure 44-130. Threshold Bank and Comparator Block Diagram

44.5.6 ADC Interface

The ADC Interface connects to the BCTU module through a Parallel Side Port (PSI). Please refer to the BCTU chapter for more information about this interface. The BCTU sends conversion results to the Reaction Module which are received by the ADC Interface sub-module. The ADC Interface is capable of distinguishing between ADC results, Time stamps and *Prefill* information from BCTU. Only ADC results with no *Prefill* are considered by the Reaction Module as valid ADC samples. The TAG value received with the conversion result indicates which channel the result is addressed to. The ADC interface is also responsible to indicate for the selected Reaction Channel that a new ADC result is available. It is also possible to access the ADC Interface input buffer through the REACM_SINR register.

44.5.6.1 Input Buffer Overrun

The OVR flag indicates than an input buffer overrun occurred. This buffer is normally written by the ADC interface (PSI) but can be written also by the CPU using the REACM_SINR register.

The ADC Interface data (ADC_TAG and ADC_RESULT) should not be updated until all Reaction Channels (with ADCR = ADC_TAG) process the received data. The OVR flag is set in the case of an overrun condition occurs, indicating that at least one ADC data was lost (the input data that caused the overrun is lost).

The following situations can generate an input overrun:

- Two consecutive sample data are received from CPU.
- ADC and CPU sending sample data at the same time (asynchronous events).
- Input buffer is holding a sample data that is being processed by some channel and a new sample is received from CPU (normal mode of operation)

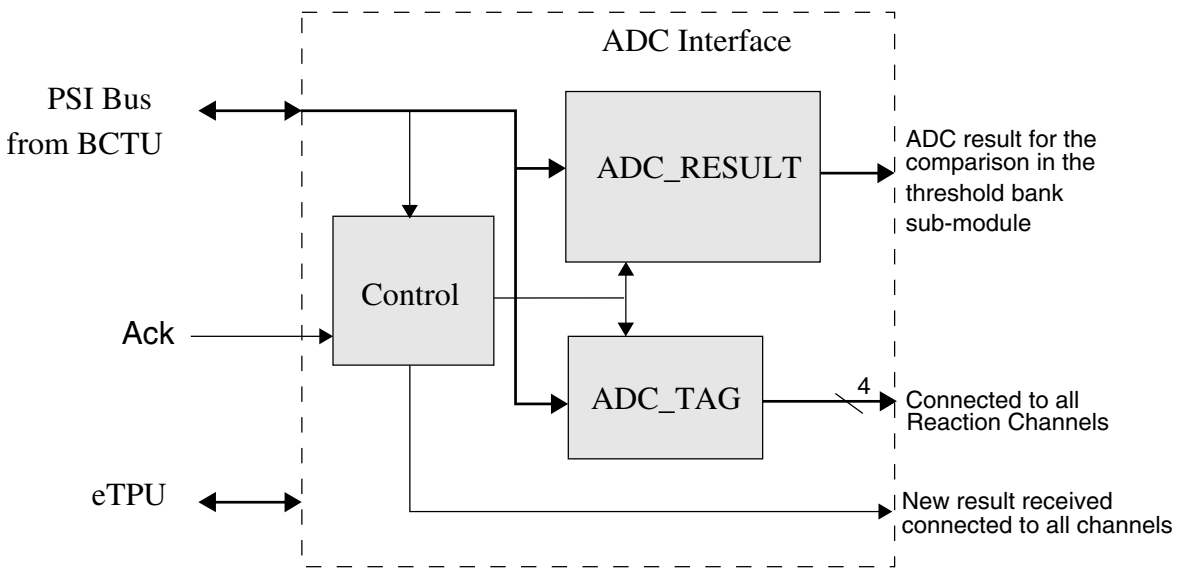


Figure 44-131. ADC Interface Block Diagram

The maximum throughput supported by the Reaction Module depends upon the TAG of the incoming ADC data. If only one Reaction Channel is addressed by the ADC TAG then the maximum supported ADC data rate is 1 sample for each 5 clock cycles. If two Reaction Channels are addressed by the same ADC TAG, thus having the same CHRR ADCR field and are active at the same time, the maximum supported ADC data rate is 1 sample for each 10 clock cycles. In general if (n) channels share the same TAG and are active at the same time the maximum supported ADC data rate is 5 x (n) clock cycles.

These limitations are related to the sharing of internal Reaction Module resources such as the Modulation Word bank. The ADC conversion data should remain stable in the ADC interface until it is used by all channels with matching TAG and CHRR ADCR fields.

Note that if all active channels have different CHRR ADCR fields, i.e. are assigned to different TAG, the maximum supported ADC rate is 5 clock cycles. If multiple channels have the same CHRR ADCR field but only one is active at a time, then the maximum supported ADC data rate is also 5 clock cycles.

44.5.6.2 On-the-fly ADC data Acquisition

The ADC Interface and the Threshold bank can operate in a learn mode, meaning that the received ADC result can be stored in the Threshold Bank and used for comparisons. This functionality allows the user to interactively calibrate the modulation levels or capture on-the-fly levels from the external application board for debug purposes. The THRR register is used for that purpose. See Figure 44-128 for more details about the register architecture. Figure 44-132 describes the main connections between the ADC interface and Threshold Bank sub-modules.

The routing of received ADC results to be stored in the Threshold Bank is independent from the routing of the same result to the Reaction Channels (see Figure 44-135). Thus the same ADC result can be used in a comparison and stored in the Threshold Bank.

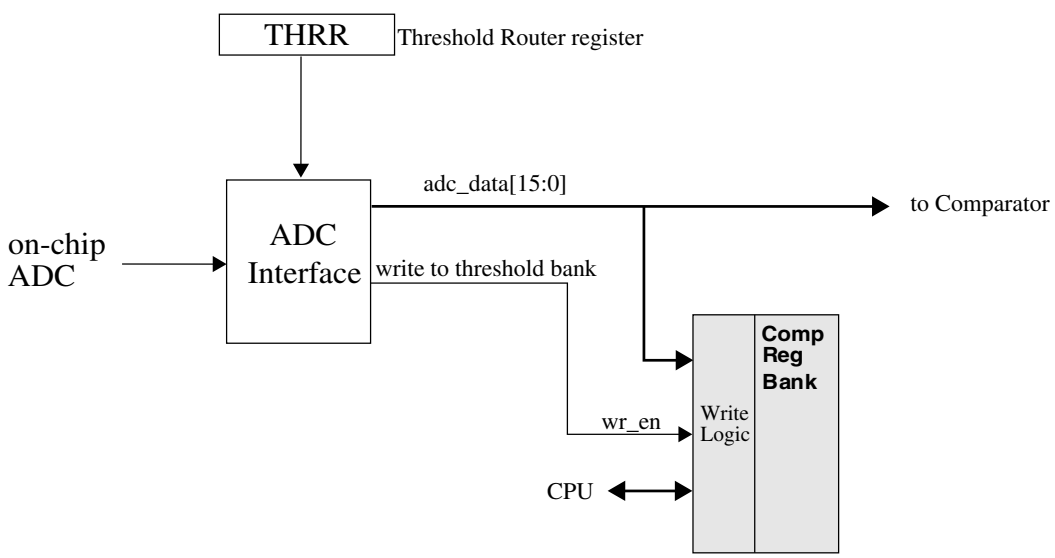


Figure 44-132. ADC interface and Threshold Bank interconnections

44.5.7 Prescalers

The Prescalers provide internal system clock divided signals to be used by internal timers. There are 2 prescalers in the Reaction Module. A 12-bit prescaler HPRE[11:0] and an 8-bit prescaler TPRE[7:0]. Both are defined in the Timer Configuration register TCR. The HPRE[11:0] is dedicated to the Hold timers within the Reaction Channels. The

TPRE[7:0] is used by the Shared Timer Bank counters. The HPRE[11:0] and TPRE[7:0] prescalers are enabled by HPREN and TPREN bits, respectively, in the Module Configuration register. Note that prescalers operates in a similar way regarding its activation. Once the prescaler is enabled by HPREN or TPREN bits in the Module Configuration Register it starts a new count sequence meaning that it is put in reset state and will generate the first prescaler tick after it reaches the programmed value defined by HPRE or TPRE fields.

44.5.8 Period / Delay Generator

This signal generator provides periodic pulses to support multi-channels operating in the threshold-period modulation modes (see Threshold low/Period Mode and Threshold high/Period Mode). There is an exclusive prescaler counter for period / delay counters. The modulation period is defined by the PER[15:0] bits in the Period Configuration register and supplied by internal period counter. This counter can be disabled if the field PER[15:0] is all 0.

Each channel receives a period pulse signal delayed from the others. The delay of the period signals between consecutive channels is defined by the field DLY[15:0] in the Period Shift Delay Configuration register. Channel 0 receives the period signal without delay. Channel 1 gets the channel 0 period signal delayed by DLY[15:0], channel 2 gets channel 1 signal delayed by the same amount, and so on. However, if the delay is zero, all period pulses occur at the same time.

Note

The correct period and delay times depend on the period prescaler setting.

The period pulses are generated for all available channels regardless if the threshold/period modulation mode is being executed. For instance, if channel 0 and 2 have Threshold/Period and channel 1 doesn't, the delay from 0 to 2 is "2 x DLY".

The following figure shows the architecture of the Period / Delay generator sub-block.

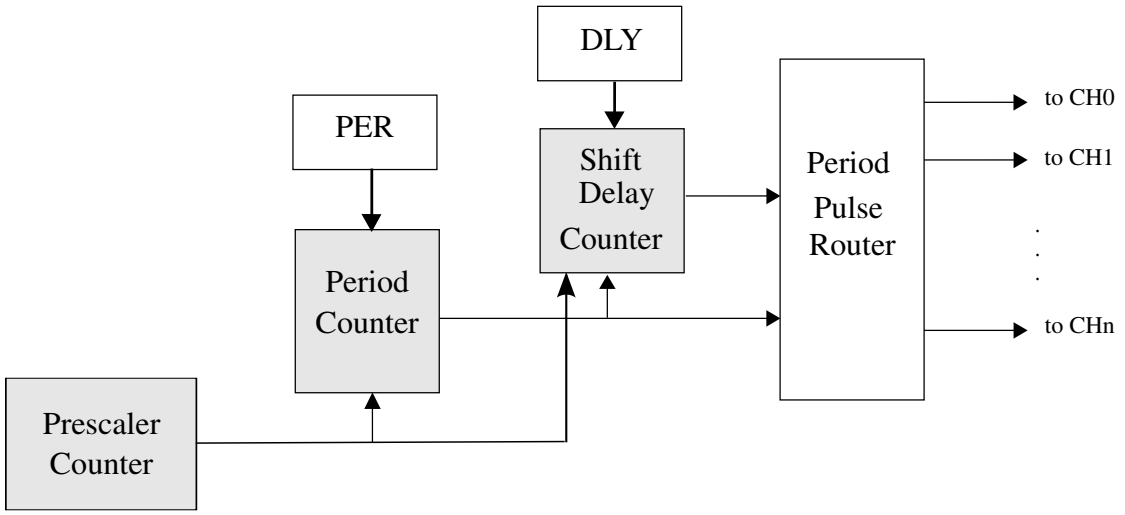


Figure 44-133. Period / Delay generator sub-block

44.5.9 Banked Mode Support

Banked Mode is a Reaction Module hardware configuration which allows the sharing of Reaction Channel output pins at the device I/O level. The banked mode architecture allows the stacking of up to four Reaction channels. The following figure shows the connection between two adjacent channels, CH0 and CH1. The CCR BSB bits are used to control the configuration of channel output logic. Thus if BSB[0] in Reaction Channel [0] is asserted to 1 and the channel is not in the active state, ch0_a output will be switched from CH[0] OUT[0] to CH[1] OUT[0]. If a channel is active, i.e. executing a modulation, it takes control over its outputs independent of BSB bits setting. The banked mode logic is extended to Reaction Channels CH[2] and CH[3] thus defining a group of 4 channels. For simplification the following figure shows only channels CH[0] and CH[1] logic.

Note

When CHEN=00 for CH0 (channel disabled), the BSB bits don't influence the channel output, which is driven by CH0's DOFF. Therefore, a banked injector driven by CH1 will have part of its controls off, even if CH1 is enabled.

To use BSB of CH0 in this case, an option is to program channel CH0 with CHEN=11 (channel enabled) and with SWMC=0 (modulation OFF).

Note

In case of using CH2 in banked mode with CH0, the intermediate channel CH1 should also be configured with CHEN different from 00.

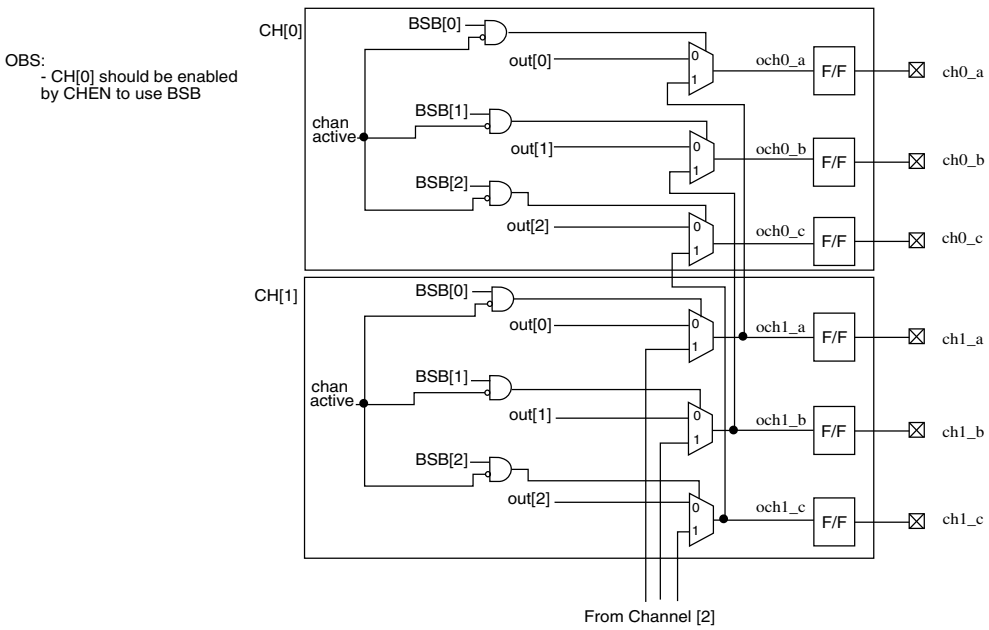


Figure 44-134. Banked Mode showing stacking of channels [0] and [1]

The Banked Mode support hardware is implemented on groups of up to 4 channels. The groups are defined as CH[3:0], CH[7:4], and CH[9:8].

Table 44-125. Banked Mode groups

CH3 → CH2 → CH1 → CH0
CH7 → CH6 → CH5 → CH4
CH9 → CH8

Thus CH[3], CH[7], and CH[9] do not connect to the subsequent channel which are CH[4], CH[8], and CH[0] respectively.

44.6 Modulation Modes

This section describes the modulation modes provided by the Reaction Module.

44.6.1 Threshold/Threshold Mode

This mode is programmed by using REACM2_MWBKx[MM] = 00. In this modulation mode the Reaction Channel tries to maintain the ADC results between two Threshold limits. The modulation actions are as follows:

- ADC result >= [THRESPT]: The Reaction Channel turns the outputs off by loading LOD[2:0] to the channel outputs.
- ADC result < [THRESPT + 1]: The Reaction Channel turns the outputs on by loading HOD[2:0] to the channel outputs.
- [THRESPT+1] <= ADC result < [THRESPT]: The Reaction Channel keeps the outputs unchanged.

The following figure indicates the threshold/threshold modulation mode.

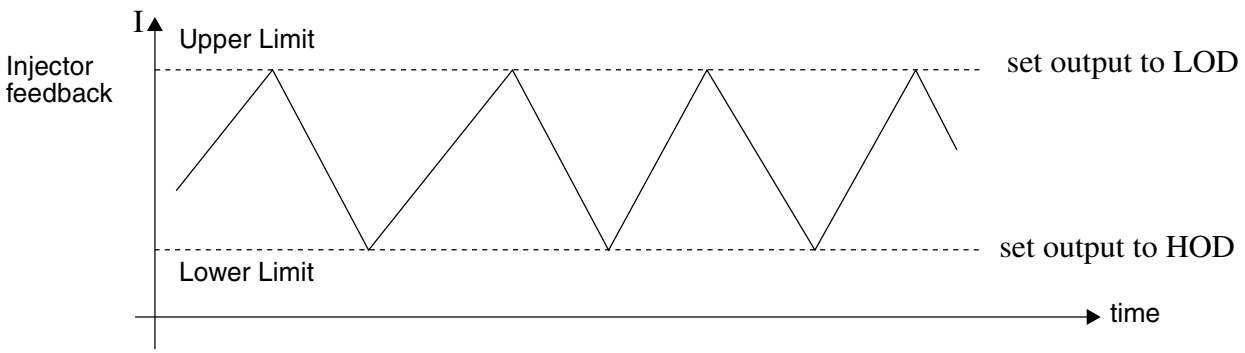


Figure 44-135. Threshold/Threshold Modulation Mode

There can be overshoots or undershoots in the real application related to the threshold limits. This occurs due to following:

- Feedback values are periodically sampled thus can present gaps on the measured values. For example when the sampling has been executed the value may have already passed the range defined by the limits;
- There is a delay between the sampling of feedback value and the reaction of the Reaction Channel. This is mainly caused by the ADC conversion time and its maximum sampling frequency.

The short circuit and open circuit detection are performed by monitoring the width of the ON pulse. For more details refer to Monitored Modulation.

44.6.2 Threshold/Hold-off Mode

This modulation mode is programmed by using REACM2_MWBKx[MM] = 01. In this mode the output pins are driven with HOD[2:0] until the ADC results matches or is greater than the programmed Threshold. The output controls are then driven with LOD[2:0] for a fixed amount of time defined by the hold timer. The initial modulation behavior can be controlled by using the REACM2_MWBKx[IOSS] bit as follows:

- When REACM2_MWBKx[IOSS] = 0, modulation starts with outputs driven with LOD until the hold timer expires.
- When REACM2_MWBKx[IOSS] = 1, modulation starts with outputs driven with HOD until the upper threshold level is reached.

Note that the upper threshold limit and the hold timer pointers are indicated by the Modulation Word in the REACM2_MWBK register.

Note

It is considered that in hold-off time the ADC samples are not reliable, therefore no comparison for MAXL or for output definition are performed. If these comparisons are necessary, the Threshold / Threshold modulation mode can be used.

The short circuit and open circuit detection are performed by monitoring the width of the ON pulse. For more details refer to Monitored Modulation

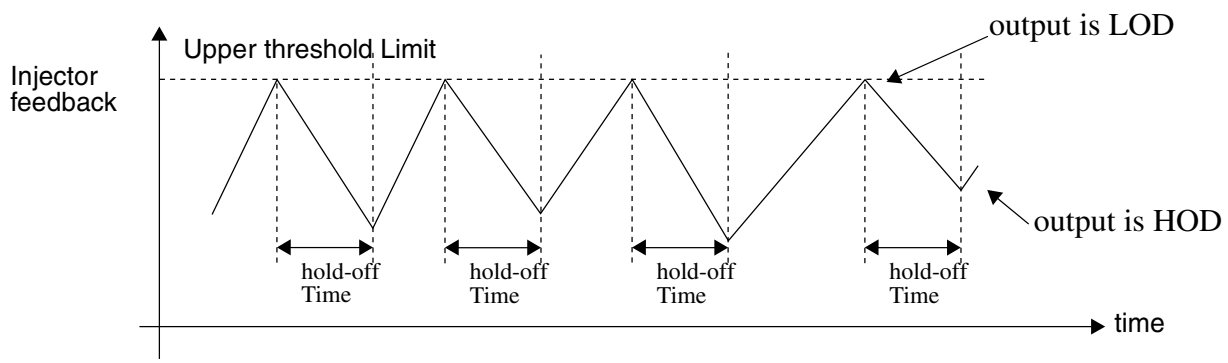


Figure 44-136. Threshold/hold-off Modulation Mode

44.6.3 Threshold/Hold-on Mode

This modulation mode is programmed by using REACM2_MWBKx[MM] = 10. In this mode the output pins are driven with LOD[2:0] until the ADC results is smaller than the programmed Threshold. The output controls are then driven with HOD[2:0] for a fixed

amount of time defined by the hold time. The exception for this defined ON time occurs if the value of an input sample surpasses the MAXL. In this case the output pins change to LOD[2:0] immediately.

The initial modulation behavior can be controlled by using the REACM2_MWBKx[IOSS] bit as follows:

- When REACM2_MWBKx[IOSS] = 0, modulation starts with outputs driven with LOD until the lower threshold level is reached.
- When REACM2_MWBKx[IOSS] = 1, modulation starts with outputs driven with HOD until the hold timer expires.

Note that the lower threshold limit and the hold timer pointers are indicated by the Modulation Word in the REACM2_MWBK registers.

The open circuit detection is performed by monitoring the minimum width of the OFF pulse. The short circuit is indicated by monitoring the ADC samples if the input sample surpasses the MAXL value. For more details refer to Monitored Modulation.

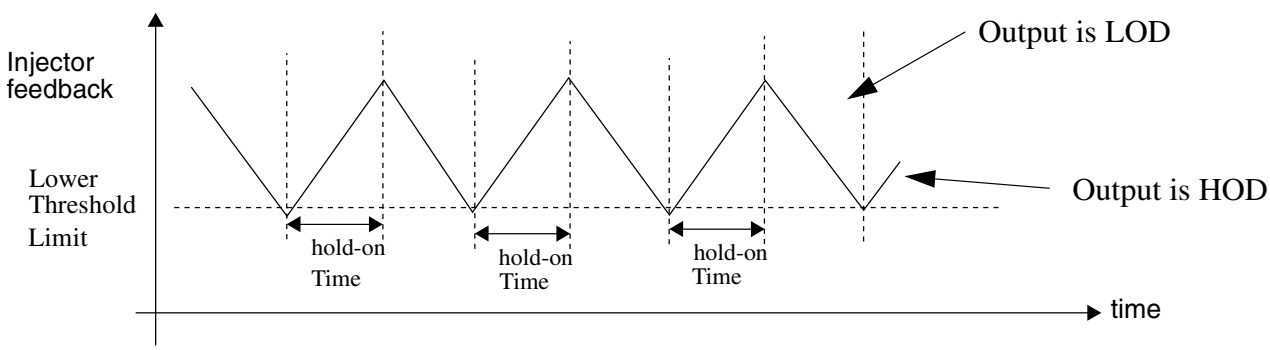


Figure 44-137. Threshold/Hold-on modulation mode

44.6.4 Threshold low/Period Mode

This modulation mode is programmed by using REACM2_MWBKx[MM] = 11 and REACM2_MWBKx[IOSS] = 0. In this modulation mode channel outputs start with LOD[2:0] and a low threshold comparison is performed until the ADC result is smaller than the programmed threshold. When the threshold level defined by THRESPT is achieved, the channel outputs states change to HOD[2:0] until the modulation period expires. The exception for this defined event occurs if the value of an input sample surpasses the MAXL. In this case the output pins change to LOD[2:0] immediately. Note that the lower threshold limit is indicated by the Modulation Word in the REACM2_MWBK registers, but the period time is given by the Period / Delay generator sub-block.

The following figure indicates the threshold low/period modulation mode.

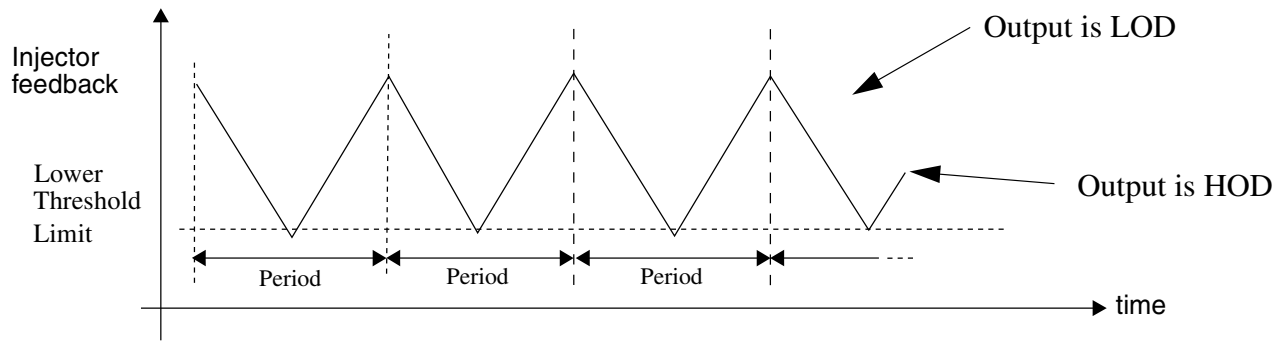


Figure 44-138. Threshold low/Period modulation mode

The open circuit detection is performed by monitoring the minimum width of the OFF pulse. The short circuit is indicated by monitoring the ADC samples if the input sample surpasses the MAXL value. For more details refer to Monitored Modulation.

44.6.5 Threshold high/Period Mode

This modulation mode is programmed by using REACM2_MWBKx[MM] = 11 and REACM2_MWBKx[IOSS] = 1. In this modulation mode channel outputs starts with HOD[2:0] and a high threshold comparison is performed until the ADC results matches or is greater than or equal to the programmed threshold. When the threshold level defined by THRESPT is achieved, the channel outputs state change to LOD[2:0] until the modulation period expires. Note that the high threshold limit is indicated by the Modulation Word in the REACM2_MWBK registers, but the period time is given by the Period / Delay generator sub-block.

The following figure indicates the threshold high/period modulation mode.

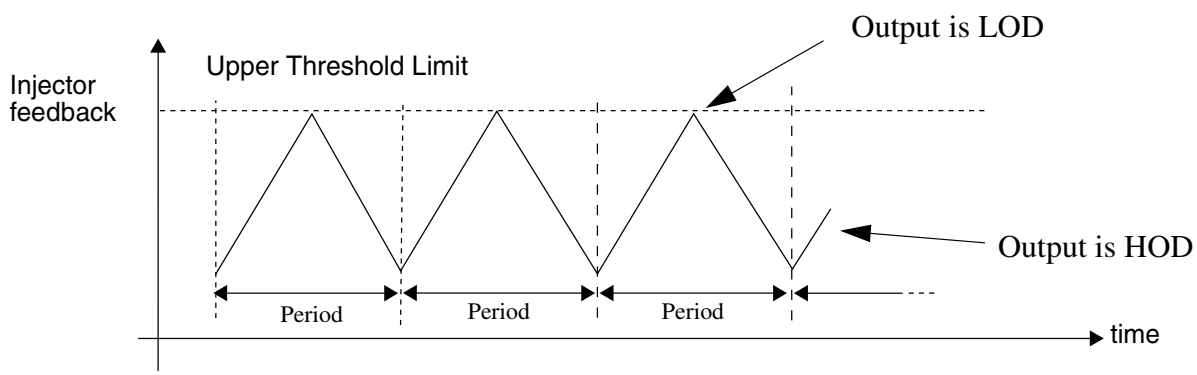


Figure 44-139. Threshold high/Period modulation mode

The short circuit and open circuit detection are performed by monitoring the width of the ON pulse. For more details refer to Monitored Modulation.

44.6.6 Limitations on the Modulation Process

This section describes the Channel limitations on the modulation process such as the width and distance between consecutive modulation pulses.

44.6.6.1 Minimum distance between consecutive Timer Control Pulses

The control signal generated by the external timer which controls the modulation process in a worst case scenario has a minimum distance of 40 system clock periods. This is required for the Reaction Channel state machine to properly re-start the modulation process and address the correct modulation word. The worst case scenario considers a Reaction Module with 10 channels and all channels being activated at the same time. If this minimum distance is violated the modulation on the second pulse will not be executed, meaning that the channel output will be defined by the DOFF field.

Note

No error flag is set if this violation occurs.

The following figure describes the channel behavior if the minimum time between two consecutive timer control pulse is violated.

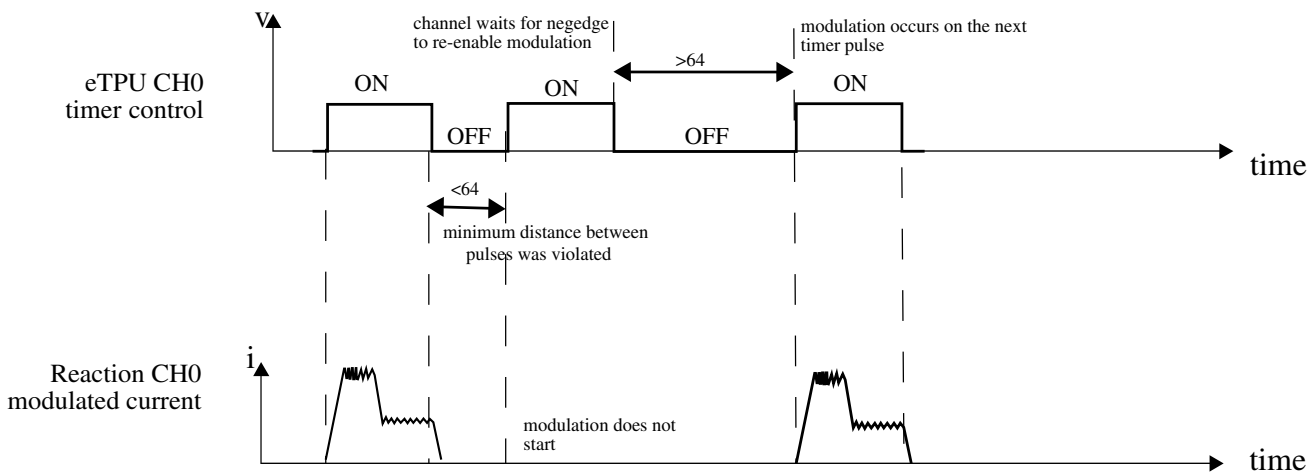


Figure 44-140. Limitation on the OFF modulation timing

44.6.6.2 Minimum Timer Control Pulse Width

There is no limitation on the Timer Control pulse minimum width. The channel just finishes the modulation process when the pulse ends even if the Modulation Words did not complete the expected sequence. Thus if a modulation phase is designed to have four Modulation Words starting from word 0, and the Timer Pulse ends at the second word (word 1) then the modulation process finishes at this point in time. The channel output is set to DOFF one clock cycle after the Timer Pulse ends. And the flag SQER is set because the SM bit field for the second modulation word is not 00. This flag occurs to indicate that the modulation was ended before the last phase of the sequence that uses the modulation word 3 and has SM=00.

The following figure describes the channel behavior when an early end of the Timer Control pulse occurs. Note that the channel output is driven to DOFF which causes the modulation to end. Note that an early end of pulse only affects the current modulation cycle. Meaning that on the next modulation cycle the modulation word 0 is executed first and all subsequent words are executed in the appropriate sequence.

Note

In some applications the modulation runs continuously and the input timer control signal is not used as a modulation pulse control but only as an enabling signal. Therefore, if the modulation is turned off, the REACM2 can issue a Modulation Word Sequence Error by setting SQER flag. In this case, this SQER flag can be ignored (masked) without prejudice.

An option, if the application permits, is to disable the REACM2 channel (CHEN=00), thus avoiding unwanted SQER error.

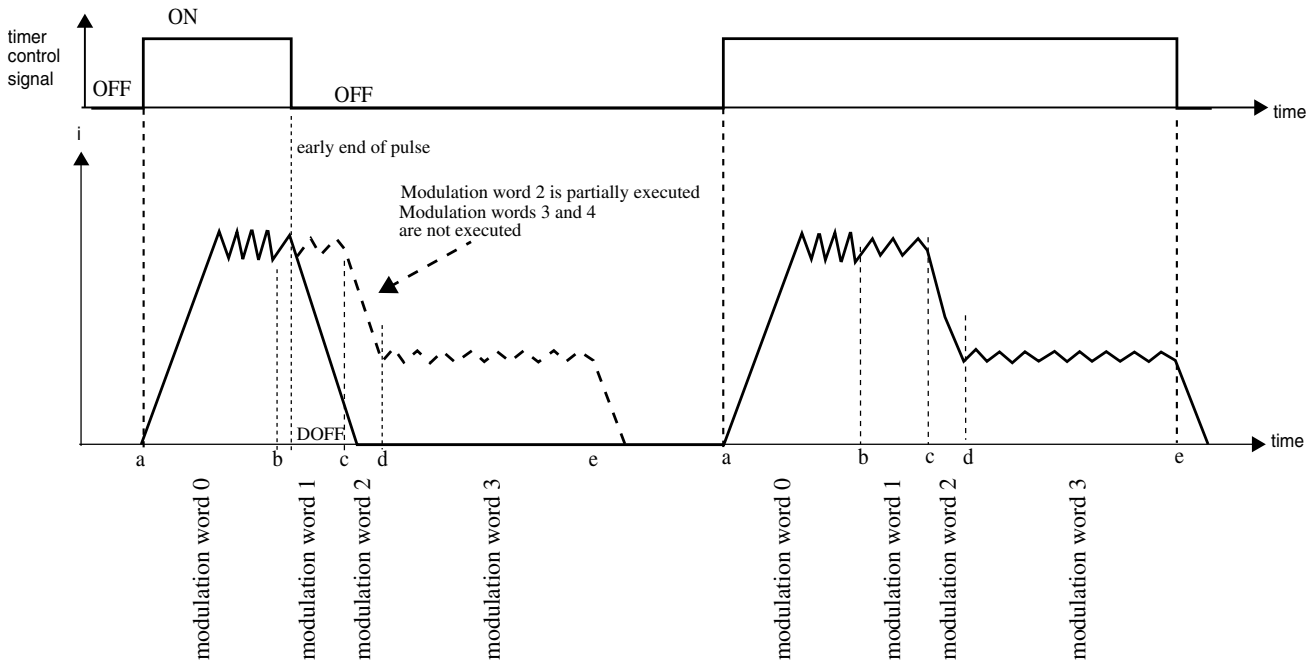


Figure 44-141. Early end of Timer Control pulse

44.6.6.3 CHOFF behavior during modulation

The CHOFF, channel output disable bit in REACM2_CHCRn, is intended to disable the channel immediately before a modulation cycle had ended. This is important on error conditions detected by the software, thus setting the channel outputs to a safe state defined by DOFF. It is important to note that re-activating the channel by setting CHOFF=0 does not imply that the channel outputs immediately return to the state defined by HOD and LOD on the modulation word being executed. Instead, the HOD and LOD are driven to the channel outputs only when one of the following conditions occur:

- A sequence advance event (time-out or threshold, depending on SM)
- A new sample is received (no matter whether the comparison matches or not)
- A hold-on/off time-out
- A period pulse occurs for the related modulation mode

44.6.6.4 Module initialization

In order to properly execute the modulation process the Reaction Module must be initialized with a correct sequence. One alternative is described as follows:

1. Make sure eTPU channels are at zero, inactive state
2. Setup BCTU module
3. Program the Modulation Words
4. Program the Threshold Bank
5. Program the Shared Timer Bank
6. Program the Hold Time Bank
7. Program Timer Configuration register
8. Program the Module Configuration register
9. Program the Period Shift Delay Configuration register
10. Program the Period Generator Configuration register
11. Program the Channel Router register
12. Program the Channel Configuration register
13. Start eTPU channels

It is important to note that the channel activation, by setting CHEN=01, should be done after all other registers were configured and before the input timer control signals are active. Violating this order may lead to errors when the modulation cycle is executed by the channel.

Note

If a glitch is introduced in the input timer control pulse, the channel stops to modulate and does not operate during the pulse just after the glitch. The glitch value for wrong operation ranges from 1 system clock to about 5 times the number of channels.

44.6.6.5 Potential glitches on the reaction channel outputs

Reaction channels outputs are digital signals that are controlled by several events that are not synchronized to each other. Due to this nature, there are a few cases when a glitch can occur at the channel outputs. If the channels outputs are going to be used by systems that are very slow compared to the generated PWM, these cases of glitches are not a problem.

The known cases of potential glitches are listed below:

- At the end of a modulation phase when a new modulation word will be used, a glitch can occur if the output has just changed prior to the end of the phase and the IOSS of the next phase forces a value that is the opposite of the last value.
- At the end of the whole modulation when the modulation is going to be disabled by the timer channel, a glitch can occur if the output has just changed to HOD prior to the disabling.
- If the threshold / threshold modulation mode is used with SM set to threshold (high) mode, at the end of the phase the output is switched off to LOD. If the modulation word for the next modulation phase has IOSS set to 1, a LOW glitch is generated. This can be avoided by setting HOD=LOD in the described previous modulation phase.

44.7 Monitored Modulation

The modulation executed by the Reaction Channel can be monitored by measuring the width of the PWM ON pulse or the OFF state.

44.7.1 Monitoring the ON Pulse Width

The modulation executed by the Reaction Channel can be monitored by measuring the width of the PWM pulses provided by the channel. If the pulse becomes too narrow it means that the load impedance is probably too low, thus indicating a possible short circuit. If the PWM pulses become too wide it may indicate an open circuit on the solenoid. These criteria can be used for threshold-threshold, threshold-hold-off, and threshold high-period modulation modes.

The PWM pulse is considered to start (rising edge) in the following cases:

- in the start of the modulation phase with IOSS=1 (threshold-threshold, threshold-hold-off, or threshold high-period modulation modes);
- when the hold-off time is over in the threshold-hold-off modulation mode;
- when the received sample is lower than the threshold LOW value (threshold-threshold, or threshold-hold-off modulation modes);

The PWM pulse to be monitored ends (falling edge) when the input sample value surpasses the threshold HIGH in the above modulation modes.

NOTE

When modulation word progresses to new modulation phase the current pulse width monitoring is stopped and a new one is started if necessary. Therefore there is no pulse width extension nor narrow pulse occurrence due to modulation phase progress.

The short detection has an additional configuration to set the SCDF flag. The SCDF flag is set if a single detection event occurs (REACM2_CHCR register field SDCM=0) or if two short detection events occur in a row (SDCM=1). For simplicity, the following descriptions are for single short detection case.

The limits for narrow and wide pulses are defined by RANGE_PWD and MIN_PWD registers (see Figure 44-140 and Figure 44-141). These values apply to all Reaction Channels. The PWM pulses are measured by the Hold timers within the Reaction Channels. This is possible only during idle periods of this timer, for example from the moment a hold-off time-out occurs until the maximum threshold is reached when HOD is being used for the channel output pins. During this period the hold counter is not used and thus it can measure pulse widths and compare them against pre-defined limits defined by RANGE_PWD and MIN_PWD registers, as shown in Figure 44-142.

Note

Consider an uncertainty of (+1) in the value MIN_PWD and (MIN_PWD + RANGE_PWD) when calculating the pulse width limits. The Hold timer prescaler contributes to this uncertainty.

For a programmed MIN_PWD value, a pulse wider than (MIN_PWD+1) does not set the SCDF flag.

For a programmed (RANGE_PWD + MIN_PWD) value, a pulse narrower than or equal to (RANGE_PWD + MIN_PWD + 1) does not set the OCDF flag.

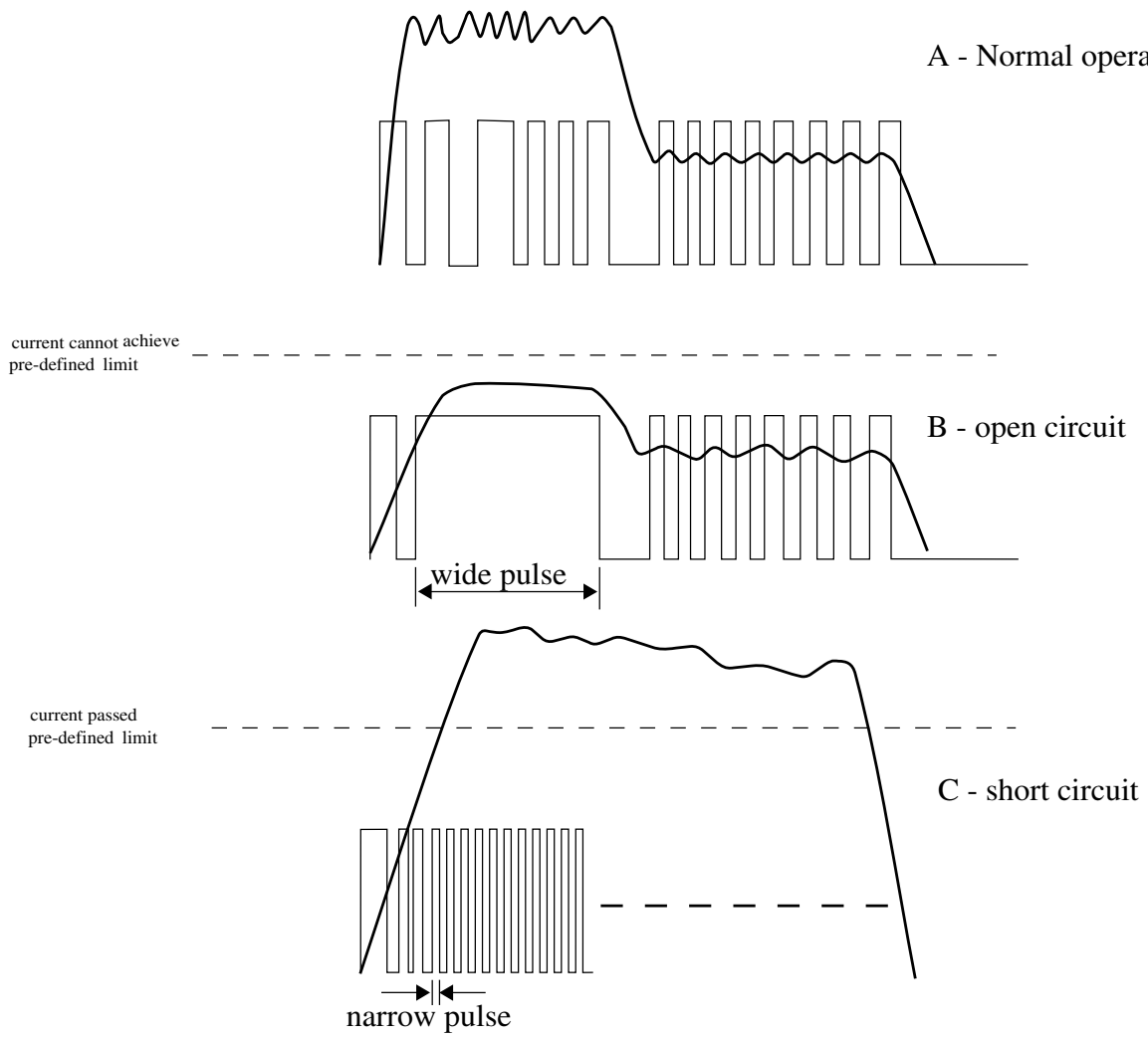


Figure 44-142. Fails detected by the pulse width monitoring

Figure 44-143 describes in more detail an open circuit detection using the Hold timer. Note that the Hold timer is used when the output is at the ON state, which means HOD is used to drive the outputs, thus hold-off functionally is not required at this moment. The Hold timer is loaded with MIN_PWD and starts counting. After a time-out occurs RANGE_PWD value is loaded and the Hold counter starts counting again. An open circuit is detected if the timer times-out on the second RANGE_PWD counting and the PWM pulse is still high, meaning that the outputs did not switch to LOD. In this case the OCDF flag is set in the Channel Status Register (REACM2_CHSR), of the corresponding channel.

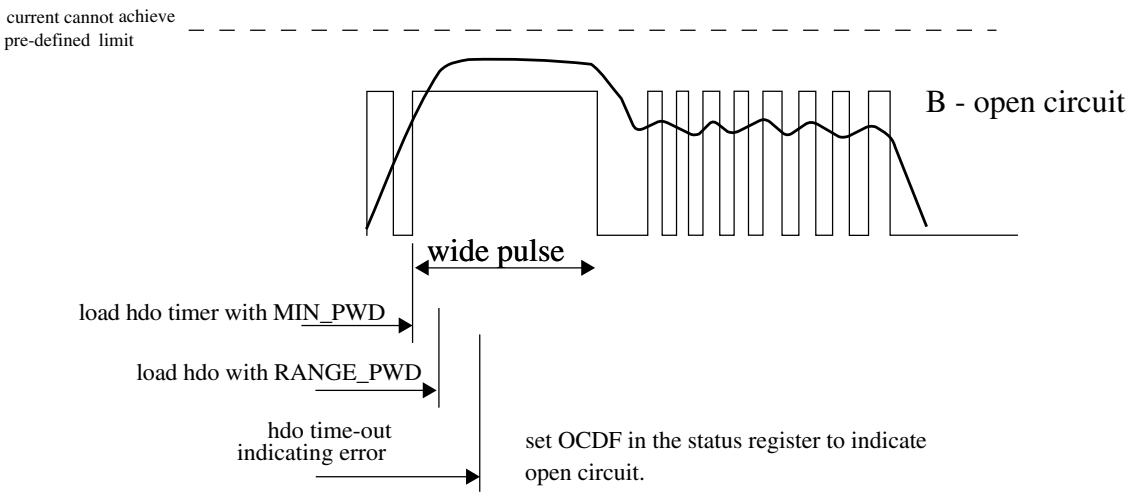


Figure 44-143. Open Circuit detection using Hold timer

Figure 44-144 describes how the Hold timer detects a short circuit. The Hold timer is loaded with the MIN_PWD value and enabled. The condition indicating that a short circuit occurred is that the Hold timer is still running and LOD is loaded to the channel output pins. That means the PWM pulse is too narrow and a short circuit has occurred. In this case the SCDF flag is set in the Channel Status Register (REACM2_CHSR) of the corresponding channel.

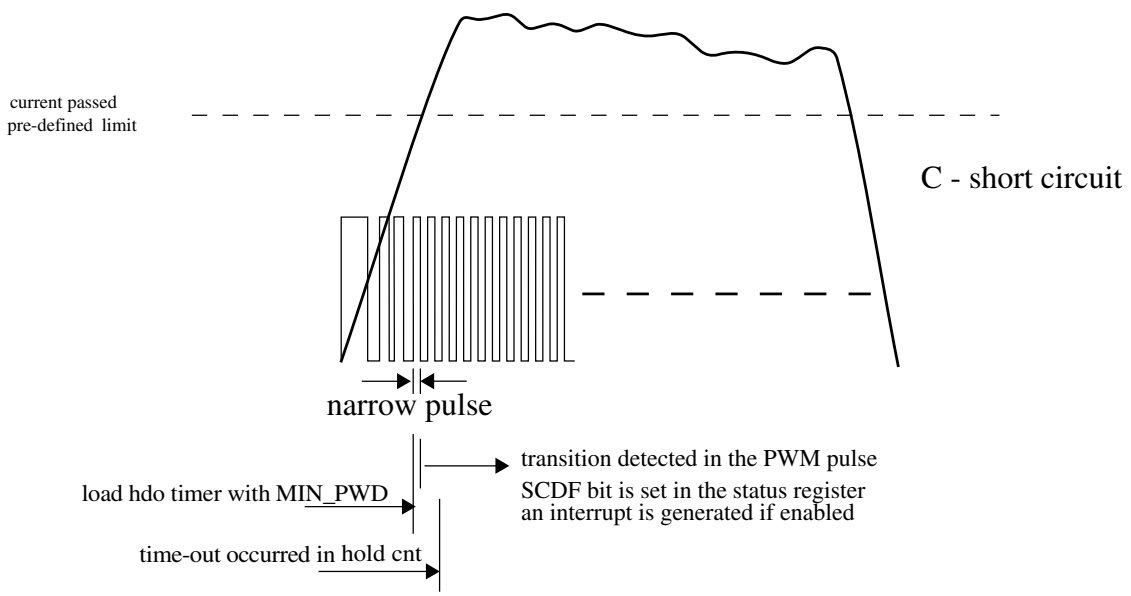


Figure 44-144. Short Circuit detection using Hold timer

Note

In order to define RANGE_PWD value it is required to consider that the Hold timer already measured MIN_PWD, thus actually the maximum allowed pulse width = (MIN_PWD + RANGE_PWD). In other words, RANGE_PWD = (maximum allowed pulse width - MIN_PWD).

IF MIN_PWD = 0x00 or RANGE_PWD = 0x00 no pulse width is performed.

The CHSR SCDF flag does not set if the pulse was finished by disabling the modulation (i.e. eTPU channel=0 or SWMC=0) or by disabling the channel, CHEN=00, even if it ended shorter than MIN_PWD. However this flag can set in some situations that really indicates a short pulse detection but it is the result of some internal condition of the reaction module. The known situations are listed below:

- When a modulation phase with MM=01 (threshold / holdoff) starts with IOSS=1 with the previous modulation phase set to SM=11 (advance on threshold).

The CHSR OCDF flag only sets when the channel is enabled (CHEN not null) and the eTPU channel signal or SWMC is active too. However, the OCDF flag can set in some cases when the CHOFF bit is set. In this case, this OCDF flag should be disregarded because it is a false indication of the detector.

There can be a conflict of resource allocation if the Hold timer is used as the timer for the sequencer mode (SM=10) because it is not possible to have the detection of minimum and maximum pulse widths. This configuration is only allowed to be used with the threshold-threshold modulation mode but the monitor of short / open circuit is not enabled. The RAER flag will be set for any other modulation mode.

NOTE

An application case is to use SM=10 with threshold / threshold modulation mode and IOSS=0. The channel output starts OFF and the SM hold timer can be configured to have a time-out before the threshold low is reached. If the threshold low is reached before the SM time-out event, the channel switches to ON but the short / open detection is not started.

44.7.2 Monitoring the OFF State Width and the ADC Maximum Value

The modulation executed by the Reaction Channel can be monitored by measuring the width of the OFF states of the PWM waveforms provided by the channel. If the OFF state becomes too narrow it means that the load impedance is probably too high, thus indicating a possible open circuit. And if the sample value surpasses the ADC_MAX_LIMIT value a short circuit condition is detected. These criteria can be used for threshold-hold-on, and threshold low-period modulation modes.

The open detection event should occur twice in a row to set the OCDF flag. This condition helps to avoid false set of the open circuit detection flag. However a single sample that surpasses the ADC_MAX_LIMIT is sufficient to set the MAXL flag

The time limit for narrow OFF state is defined by MIN_PWD register. This value applies to all Reaction Channels. The PWM OFF states are measured by the Hold timers within the Reaction Channels. This is possible only during idle periods of this timer, for example from the moment a hold-on time-out occurs until the minimum threshold is reached when LOD is being used for the channel output pins. During these periods the hold counter is not used and thus it can measure the OFF states widths and compare them against the pre-defined limit defined by MIN_PWD register, as shown in the figure below.

Note

Consider an uncertainty of (+1) in the value MIN_PWD when calculating the OFF state width limit. The Hold timer prescaler contributes to this uncertainty..

For a programmed MIN_PWD value, a pulse wider than (MIN_PWD+1) is not an open circuit event to set the SCDF flag.

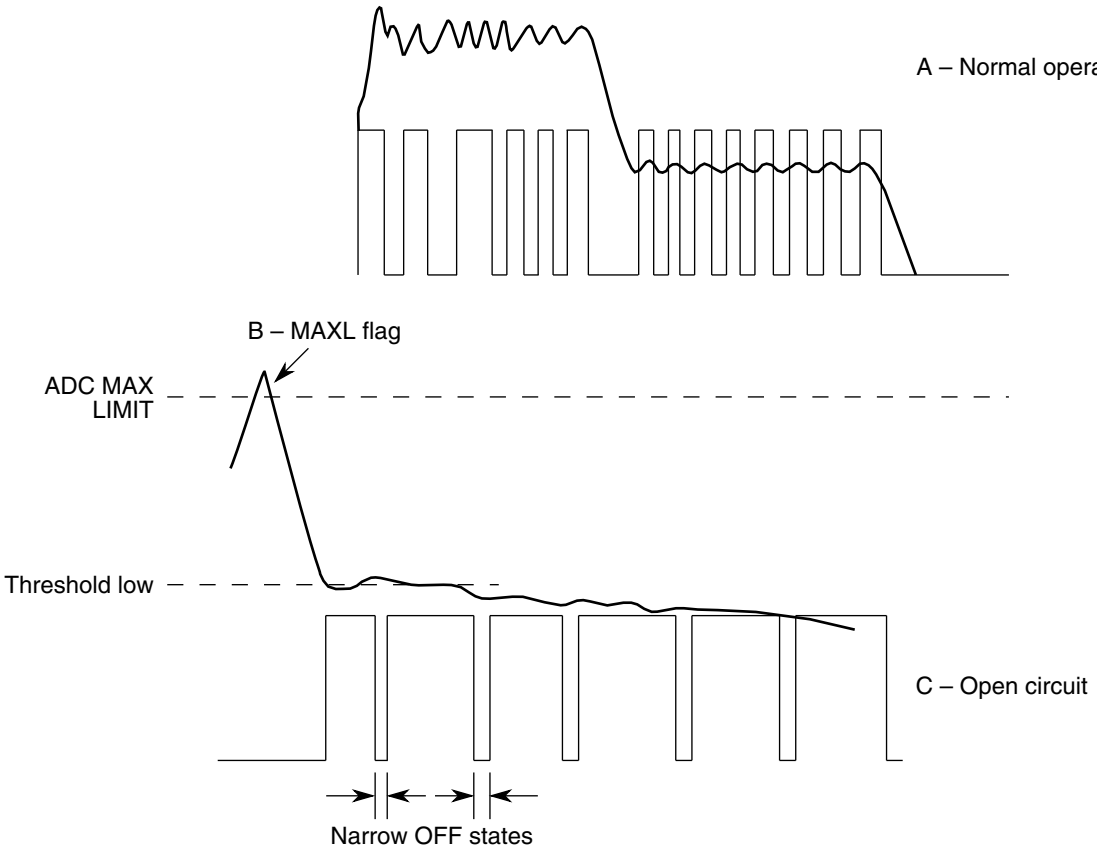


Figure 44-145. Fails detected by the OFF timing monitoring

The CHSR OCDF flag does not set if the OFF state was finished by disabling the modulation (i.e. eTPU channel=0 or SWMC=0) or by disabling the channel, CHEN=00, even if it ended shorter than MIN_PWD.

There can be a conflict of resource allocation if the Hold timer is used as the timer for the sequencer mode SM=10. In this case the RAER flag will be set to signal this conflict condition.

44.8 DMA support

The Reaction Module provides supports for one DMA channel per Reaction Channel. The DMA request signal is controlled by the DMAEN bit in the Channel Configuration Register (see Figure 44-133) and by the DMA bit in the Modulation Control Word. If the DMAEN=1 and the DMA = 1 then a DMA request is issued by the Reaction Channel. Note that the DMA request is deasserted if the DMA done signal is asserted even though the channel is still pointing to the same Modulation Control Word that generated the DMA request. In order for a new DMA request be issued after the DMA done is issued, the Reaction Channel must access a new Modulation Control Word or execute a new modulation cycle controlled by the timer input signal. The following figure shows the

DMA protocol executed by the Reaction Channel. The DMA request signal is asserted when the Modulation Word 1 is executed by the channel. This signal remains asserted until a DMA done signal is issued by the DMA controller.

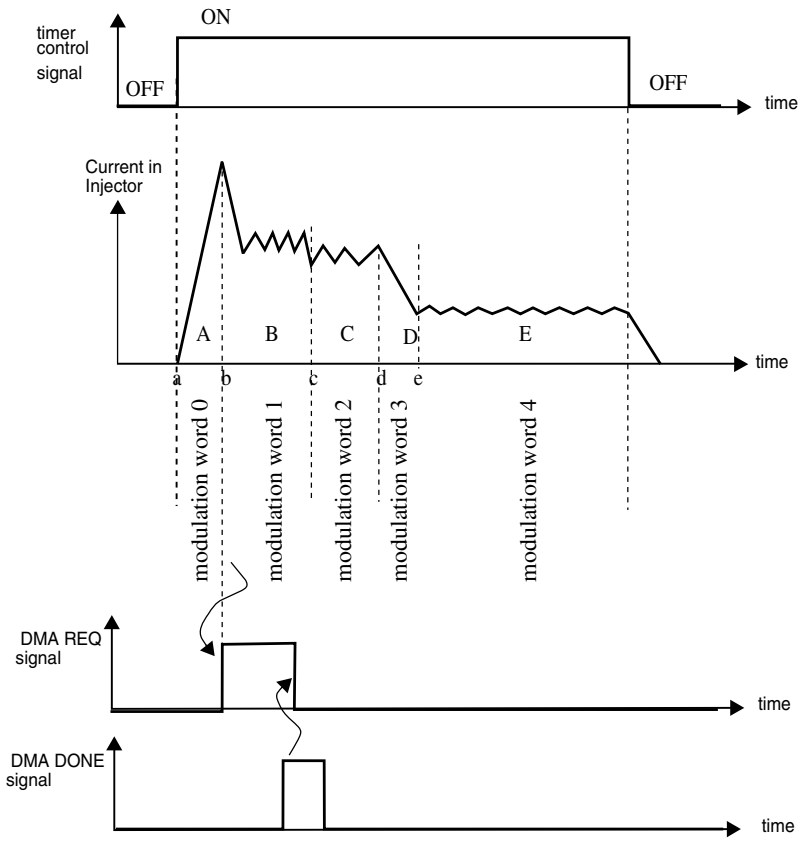


Figure 44-146. DMA Req/Done protocol

44.9 Reset Overview

The Reaction Module is reset whenever any MCU reset occurs. In order to re-initialize the Reaction Channels, the CHEN channel enable register should be used. Once disabled the channel output is set to DOFF state and the channel configuration can be changed safely. The disable/enable operation does not change the channel setup, thus the configuration registers remain at the state as before the channel was disabled. The Modulation Control word addressed by the channel after the enable bit is asserted is defined by the MODULATION ADDR field in the channel configuration register.

44.10 Reaction Module Interrupts

The Reaction Module issues one global interrupt signal and one interrupt signal per channel. If the Global interrupt signal is used, the resolution of the interrupt source needs to be performed by reading the Global Error Flag register to evaluate which channel issued the interrupt. After that the Channel Status register needs to be read to distinguish between the several interrupt sources by evaluating the flags: MAXL, OCDF, SCDF, TAER and SQER.

44.10.1 Interrupt sources

There are several sources of interrupts that indicates a faulty condition:

1. MAXL: maximum ADC result value was reached
2. OCDF: open circuit detected which indicates an open circuit, and thus, a potential malfunction in the circuitry controlled by the Reaction Module. Note that differently from the TAER, this flag does not indicate a faulty condition in the channel but in the circuit outside the SoC.
3. SCDF: short circuit detected which normally indicates a short circuit was detected on the off chip logic controlled by the reaction channel.
4. TAER: timer allocation error which indicates a required Timer resource was not allocated properly thus leading to faulty operation of the Reaction module.
5. SQER: sequencer error occurred meaning that the timer input signal was negated in a modulation phase with SM != 00.

44.10.2 Global Interrupt and Channel Interrupt

The following figure illustrates the relationship between the channel interrupts and the module global interrupt. The global interrupt summarizes the indication of error in the reaction module.

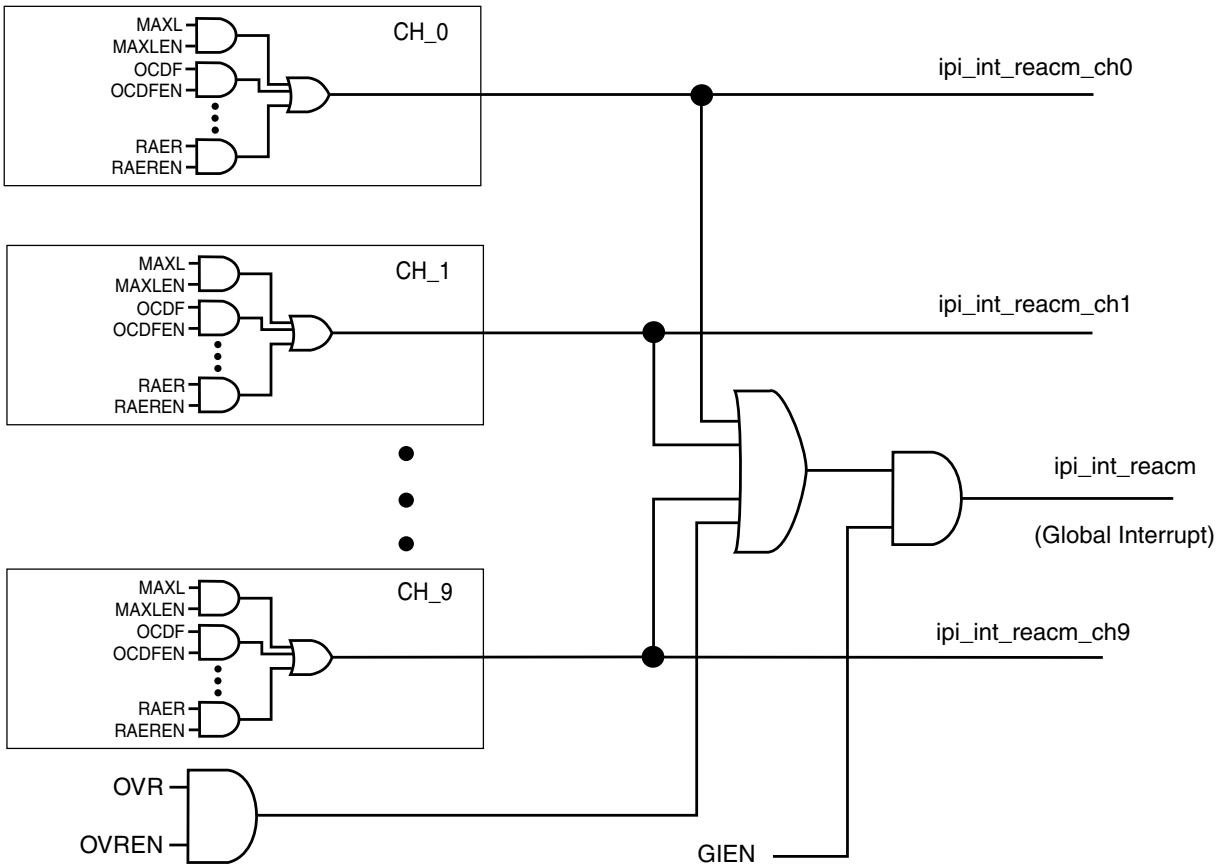


Figure 44-147. Channel Interrupt and Global Interrupt Diagram

44.11 Use cases

Figure 44-148 shows an example of the Reaction Module used to control an Injector solenoid. Note that this is a dual injector which is also called banked injector. Two Reaction channels are used to control this injector.

The injector Boost transistor on the top applies a higher voltage in order to minimize the time necessary for the injector to start injecting fuel. Transistors A and B control which injector is active in the injector bank. A sensor resistor is used to feed the current flowing through the solenoid back to the on-chip ADC. The current is sampled by the ADC and the result is sent to the Reaction Module, allowing closed loop control.

Note

This injector bank architecture does not allow both injectors to operate at the same time since the sensor in the feedback loop is shared by both injectors.

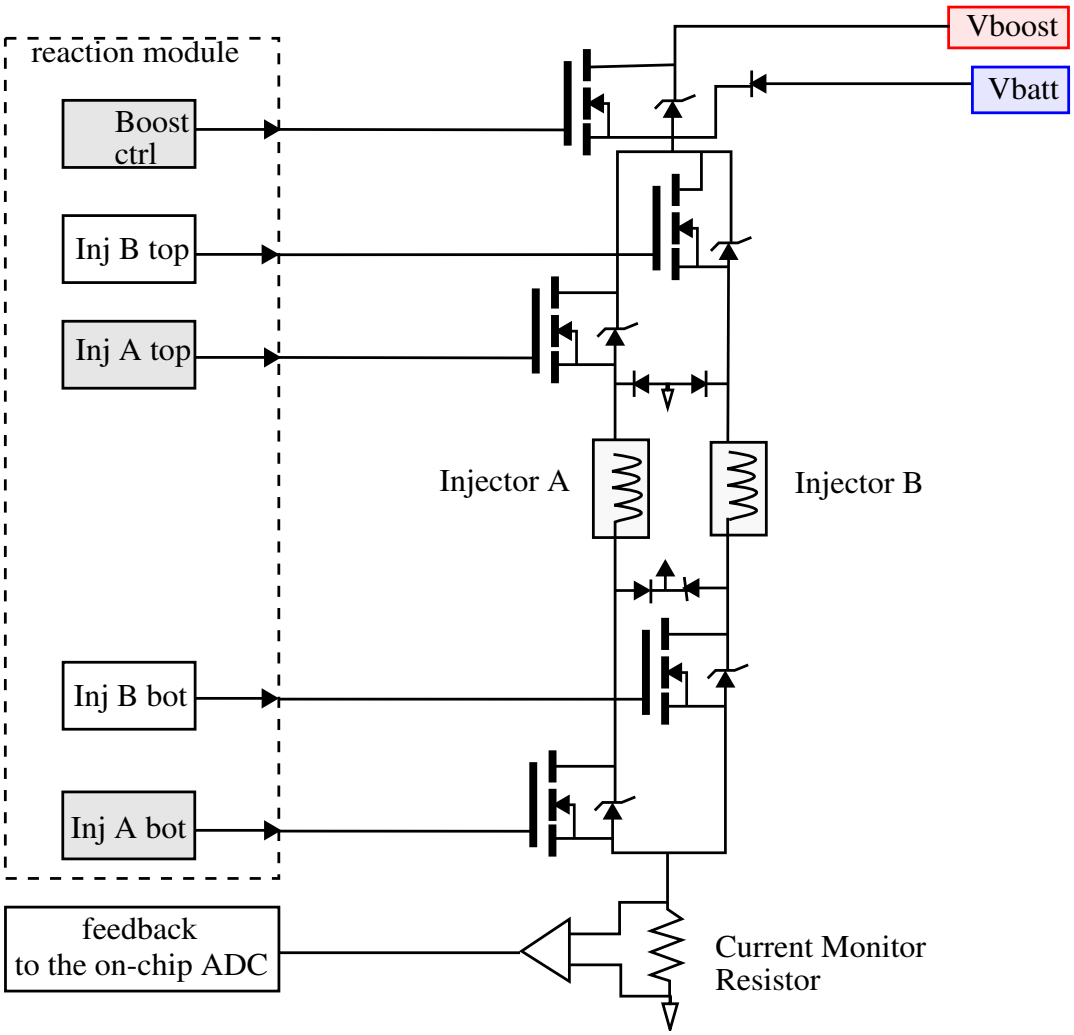


Figure 44-148. Boosted Banked Direct Injection with Passive Recirculation

Two eTPU channels are used to provide timing control signals, one for CH0 and one for CH1. The on-chip ADC monitors the sensor current periodically and sends the digitalized results to the Reaction Module. In a banked configuration as shown in this example one ADC channel is used to monitor both injectors current. Note that only one channel is *active* at a given time since the eTPU time windows are not *active* at the same time for both Reaction Channels. See Figure 44-149 for more details.

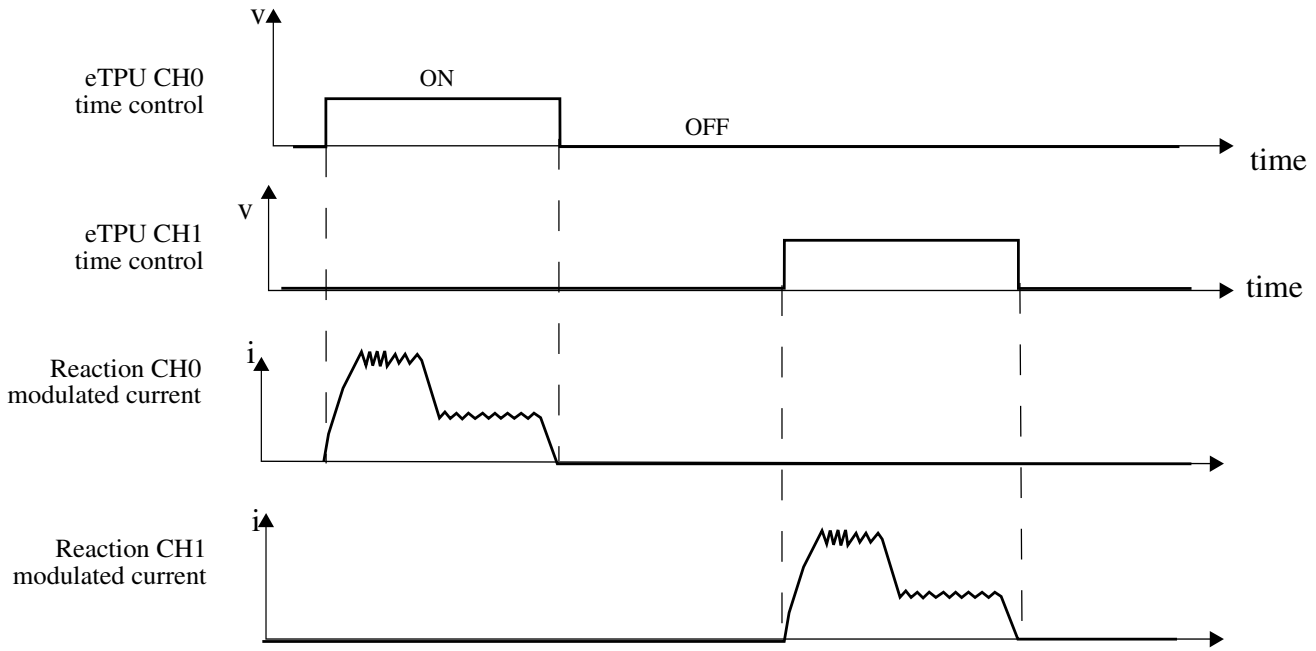


Figure 44-149. eTPU CH10/1 Controlling Reaction CH0/1

There are several possible configurations for a banked mode application. The objective on those configurations is to share hardware resources such as Reaction Module channels, ADC monitor inputs and MCU pins, among others related to the Injector driver which is not covered in detail in this document.

Figure 44-150 shows a more detailed diagram of the interconnection between the injector bank and the Reaction Module. Two reaction channels are used in this application. CH0 is used to control Injector A and CH1 is used to control Injector B. However, the Vboost/ Vbatt selection is controlled by CH0 ch0_a output only since, when VBoost driver is switched off, Vbatt power source is applied to the injectors by the direct bias of the diode connecting Vbatt to the injector bank.

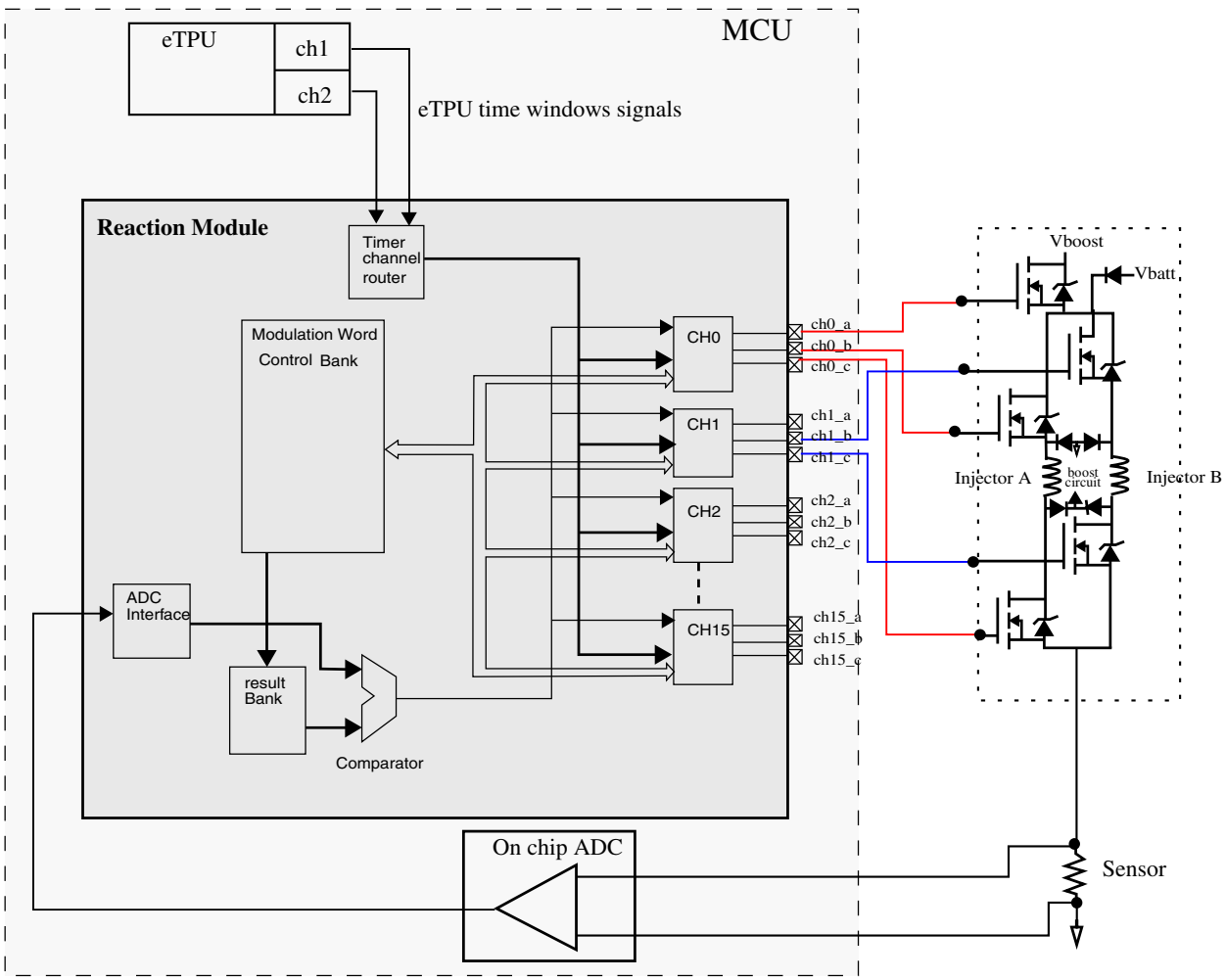


Figure 44-150. System Level Connection in a Banked Configuration

It is important to note that even if two reaction channels control different injectors they can share the data stored in the Modulation Word Control. In this case both channels should execute the same type of modulation and use the same threshold values. Note also that the data stored in the Threshold bank in this case is also shared between these channels. This is an important feature of the Reaction Module architecture since it allows the sharing of resources and therefore provides savings in size without compromising the module functionality.

Figure 44-151 shows an example of the required current levels through Injector A and B. In order to generate this waveform, the Reaction Module uses one Modulation Control Word for each one of the four phases of the waveform from A through D. In this example the Module should be configured in the following way:

1. Set the REACM2_CHRR0 CHIR[3:0]=0x0, thus routing eTPU channel 0 to Reaction Channel 0
2. Set the REACM2_CHRR1 CHIR[3:0]=0x1, thus routing eTPU channel 1 to Reaction Channel 1
3. Set the REACM2_CHRR0 ADCR[3:0]=0x0, thus routing ADC TAG 0 to Reaction Channel 0
4. Set the REACM2_CHRR1 ADCR[3:0]=0x0, thus routing ADC TAG 0 to Reaction Channel 1
5. Program Modulation Word Control bank according to Figure 44-152
6. Program Shared Timer Bank REACM2_STBK for addresses from 0 through 1 with timing intervals related to the duration of phases B and C respectively.
7. Program appropriate values in the Threshold Bank. Since threshold-threshold modulation is to be used in this example, 4 pairs of values should be provided for phases A,B,C and D respectively. Each pair corresponds to one address of the REACM2_THBK starting at address 0x0400.
8. Program configuration registers for both channels, REACM2_CHCR0/1. The parameters are DOFF[2:0] which defines the OFF state of the channel outputs and the MODULATION_ADDR = 0x0, which defines the address of the Modulation Control word. It is assumed that the Modulation Word zero is the first word to be accessed by both channels. Since 4 Modulation words will be used the addresses will be incremented by the Reaction Channel as needed, thus only the address for the first word is required. Note that MODULATION_ADDR=0x0 points to the first Modulation Word in the Modulation Word Bank.
9. Program the prescalers HPRE and TPRE in the REACM2_TCR register. Also enable the prescalers by setting the TPREN and HPREN bits in the REACM2_MCR register.
10. Enable channels CH0 and CH1 to start the modulation sequence by programing field CHEN=01 on REACM2_CHCR1/0 registers. At this time the Reaction channel CH0 accesses the Modulation Control word zero and switches to ON state as defined by the data stored in the HOD[2:0] field. Up to this point any activity in the eTPU channel or incoming ADC result is ignored by the Reaction module. After CHEN field is programed, the reaction channels wait until a timer window is initiated by eTPU for the modulation process to start.

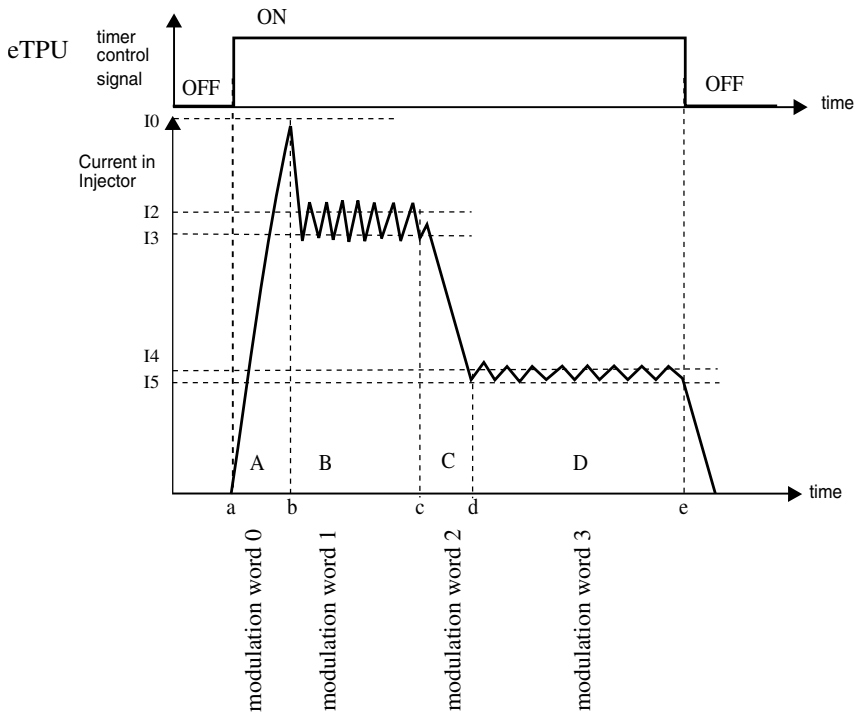


Figure 44-151. Modulation Phases1

The current through the injectors are initially zero since the reaction channel output is in the OFF state as configured by the DOFF[2:0] field. The eTPU timer window is at the OFF state as well. The modulation starts when the eTPU channel time window switches to ON state. Modulation *phase A* starts at this time as described in Figure 44-151. The modulation word 0 is used. In this application this phase corresponds to the *Boosted Peak phase*. This process allows the fuel injection to start sooner because of the sharp edge of the current which is important for a precise control of the fuel to be injected. *Phase A* is setup to execute a Threshold-Threshold modulation but note that only *I0* value is used since the phase advances when a certain current is achieved. Below is a description of the bit fields in the Modulation word in order to execute the modulation described in *phase A*:

- LOOP=0
- The initial value should be HOD (IOSS=1)
- Threshold-Threshold modulation mode is obtained with MM=00
- This phase ends when the threshold value *I0* is achieved (SM=11)
- Necessary to have HOD=111 for boosted operation and sensor active. LOD setting is not important in this case since it is not used
- *I0* is read from Threshold Bank by using THRESPT=0x0 that points to address 0 of this bank

- *I1* is read from Threshold Bank by using (THRESPT+1) = 0x1
- Hold timer is not used, therefore HDOFFPT can have any value (X).

At point b, *phase B* is initiated by the threshold being achieved from *phase A*. CH0 increments the Modulation Word address to MODILATION_ADDR = 0x1 and the second Modulation Word is read by the channel. As a result of the second Modulation Word decoding the *Vboost* voltage is disabled causing a *peak* modulation with *Vbatt*. The phase is called the *Peak Vbatt phase*. For *phase B* a Threshold-Threshold modulation is used with levels *I2* and *I3* during a period defined by *TB*. See Figure 44-152

A time-out event is received from the Shared Timer sub-module and a new Modulation Word is read. *Phase C* corresponds to the *re-circulation phase*. Energy from the injector is transferred back to the boost circuitry. In this phase the current cannot be measured because there is no current flowing through the sensor resistor. A Threshold-Threshold modulation mode is used. The Shared Timer is started at the beginning of this phase and *TC* delay is measured. The channel outputs are kept in the OFF state.

When the Shared Timer times out after *TC* delay the Modulation Word address is incremented, MODULATION_ADDR = 0x4 and the *Hold phase* is initiated. This phase is typically longer compared to the other phases and defines the amount of fuel that will be injected. Threshold-Threshold modulation mode is used between levels *I4* and *I5* and *Vbatt* is selected as the power supply. The phase ends based on the eTPU time window switching to *off* at point *e*. At this time the channel outputs are set to OFF and the channel points to MODULATION_ADDR = 0x0 which is the address of the first Modulation Word. Note that this address is not necessarily 0x0.

This modulation process is executed by a sequence of Modulation Words as described in Figure 44-152.

Modulation Control Word Bank (REACM2_MWBKn)

	31 0	30 1	29 2	28 3	27 4	26 5	25 6	24 7	23 8	22 9	21 10	20 11	19 12	18 13	17 14	16 15	15 16	14 17	13 18	12 19	11 20	10 21	9 22	8 23	7 24	6 25	5 26	4 27	3 28	2 29	1 30	0 31
Fields	L	I	DMA	MM			SM			HOD				LOD			THRESHPT								STPT				HDOFFPT			
Phase A	0	1	0	00			11			111				101			000000 (I0) 000001 (I1)								X				X			
Phase B	0	0	0	00			01			110				100			000010 (I2) 000011 (I3)								00000 (TB)				X			
Phase C	0	0	0	00			01			000				000			X [1]								0001 (TC)				X			
Phase D	0	0	0	00			00			110				100			000100 (I4) 000101 (I5)								X				X			

Figure 44-152. Modulation Words for Injector Application

Note

[1] Any value

44.11.1 Advancing Modulation Phase on a Threshold level

The Modulation Phase may be set to advance when a specific threshold value is reached. The use case for this scenario is described in the following figure. This functionality is used to assure a specific current level was achieved before the Reaction Channel advances to the next modulation phase, thus making sure that the solenoid had the fastest opening speed.

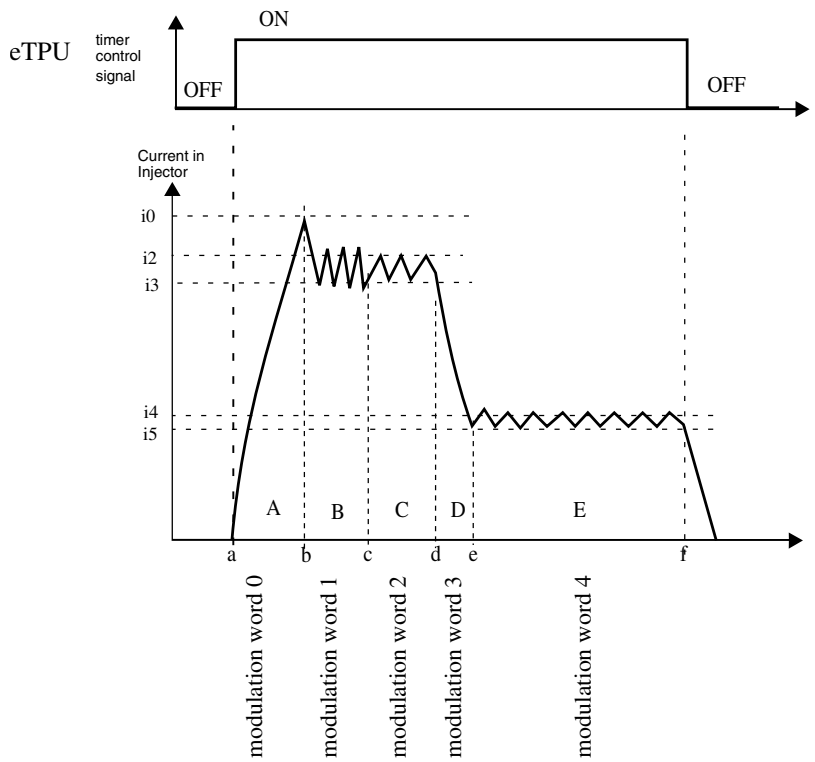


Figure 44-153. Advancing Modulation Phase on a Threshold level

In the example shown in the preceding figure the Modulation Word 0 has the field SM=11 meaning that it should advance if the ADC comparator, which compares ADC results and threshold values, is true. This means that the ADC result is greater or equal to the threshold value pointed by THRESPT Modulation Word 0 field. The advance from phase A to phase B in the preceding figure is described as follows:

- Initially at time *a* the Reaction Channel output is loaded with HOD[2:0] as indicated by IOSS value, both fields of modulation word 0.
- If ADC result >= [THRESPT] the Reaction Channel turns the outputs off by loading LOD[2:0] to the channel outputs and advance to the next modulation cycle at time *b*.

Note

The advance on threshold, SM=11, is intended to be used with IOSS=1, thus the advance occurs when the level from the ADC is greater or equal to the value pointed by THRESPT. IOSS=0 is a reserved value for this bit in this configuration and should not be used.

Note

In case of MM=01 (threshold/holdoff) on phase B, a glitch can occur in the beginning of this phase. To avoid this, on phase A it is necessary to set HOD=LOD. However, a false flag SCDF can be set at this time and should be discarded.

44.11.2 Controlling the LOOP function

The LOOP field in the Modulation Word controls the sequencing of Modulation Words to be executed by the channel. If LOOP = 1 and a phase is ended the next Modulation Word address returns to the initial modulation address programmed in the channel when the cycle was initiated. In this case a loop in the Modulation Bank is created. This loop ends when the cycle ends.

The following figure shows the sequence of Modulation Words within a Modulation Cycle if LOOP function is used. Note that an alternate modulated waveform is generated using only two Modulation Words, in this case Modulation Word 0 and Modulation Word 1.

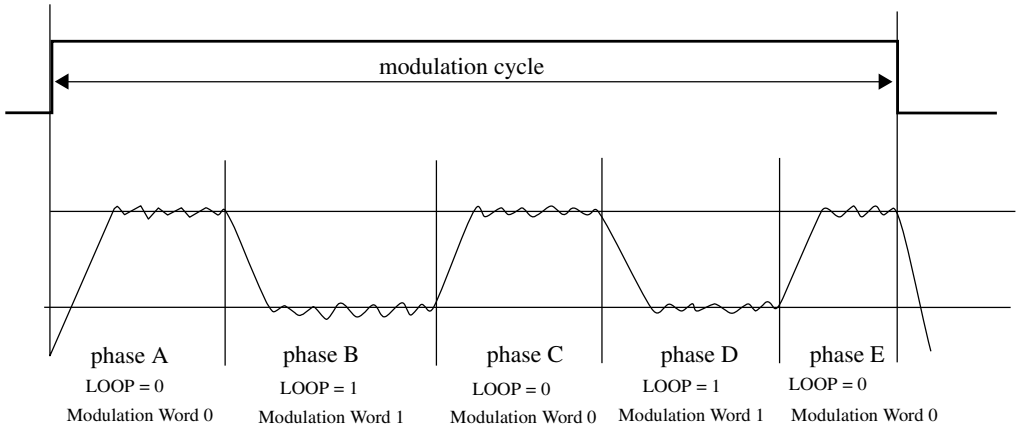


Figure 44-154. LOOP function used within a Modulation Cycle

NOTE

Due to the limit of 16 words for a channel modulation, after the 16th modulation word accesses, the sequence advance is executed by returning back to the first word of the sequence even with LOOP=0.

44.11.3 Banked Mode

The following figure describes the interconnection of four channels controlling two injector banks. Note that two output pins are not connected since the boost control is shared between channels [0] and [1] and channels [2] and [3].

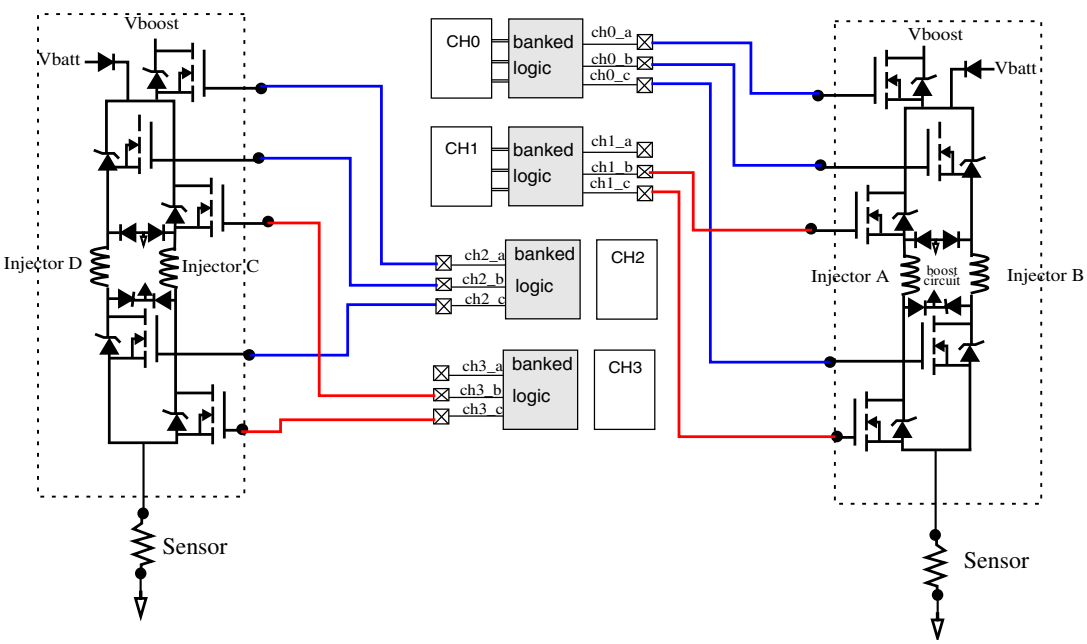


Figure 44-155. Four channels controlling two injector banks in banked mode.

Chapter 45
Input Glitch Filter (IGF)

45.1 Input Glitch Filter chip-specific module information

This device implements one cluster of Input Glitch Filter modules comprising of 32 input filter channels. (See the Filter Cluster section.)

- The inputs are connected to the pins through an input mux which is implemented in the IOMUX. See the Signal Multiplexing and Signal Description chapter.
- 32 filter channels
- 24-bit counters in each channel
- One 9-bit prescaler on channel 0
- The filtered output from Input Glitch Filter routes to both eTPU and eMIOS.

The following figure shows the interaction of the Input Glitch Filter with other modules in the device.

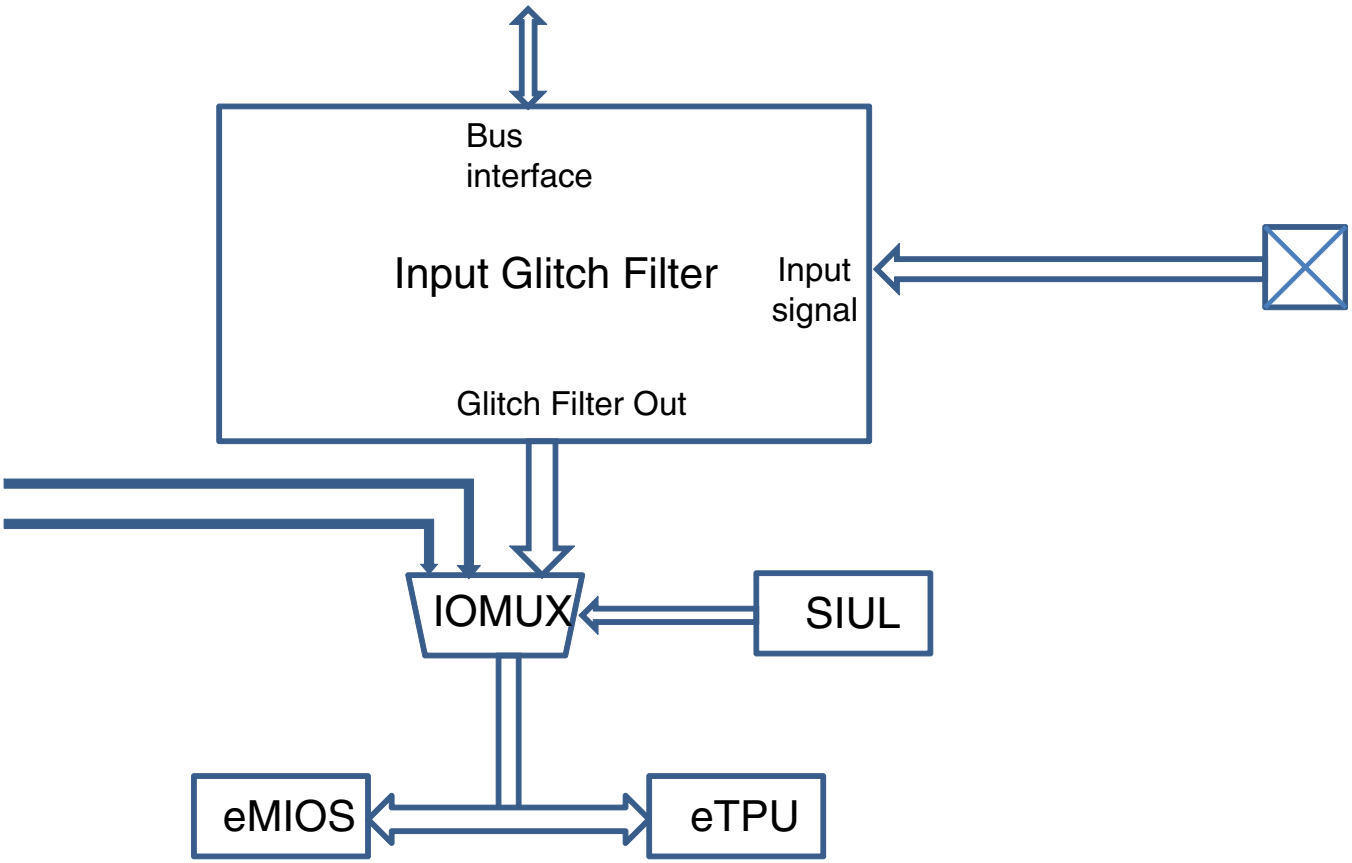


Figure 45-1. Input Glitch Filter SoC Interaction

45.2 Introduction

The Input Glitch Filter provides programmable glitch rejection filtering of digital input signals.

45.3 Filter Cluster

The MPC5746R has 32 Input Glitch Filters organized in to one filter cluster. The first filter in the cluster (Filter 0) has an internal prescaler. All other filters do not have an internal prescaler. There is no external prescaler for the filter cluster.

Each filter in the filter cluster uses a fixed range of addresses from 0x0000 to 0x003F. All filter registers (MCR, MSR, PRESR, RTHR, and FTHR) are placed in this range. The unused addresses in this range are reserved, thus if there is an access to these reserved addresses a transfer error occurs.

For Filters 1 to 31, which do not have an internal prescaler, the addresses for the PRESR register are reserved. For these filters, selection of the internal prescaler by setting MCR[PSSEL] = 0 selects the filter counter to the system clock rate as if the internal prescaler is programmed with zero.

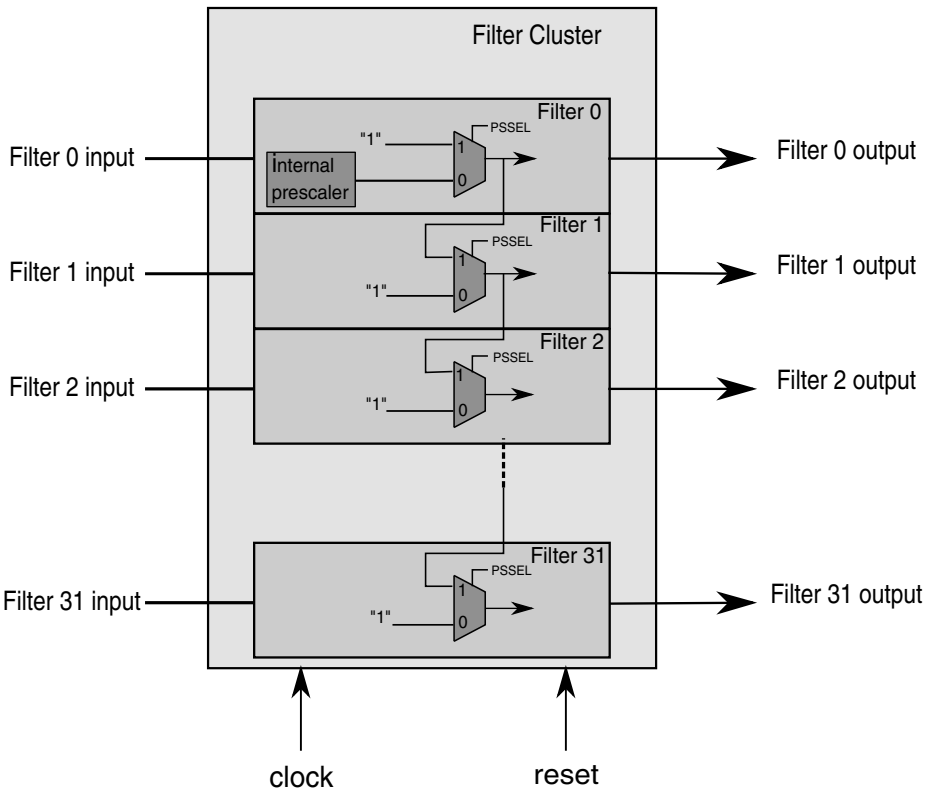


Figure 45-2. MPC5746R Input Glitch Filter Configuration

45.4 Block Diagram

The figure below is the block diagram for the Input Glitch Filter module.

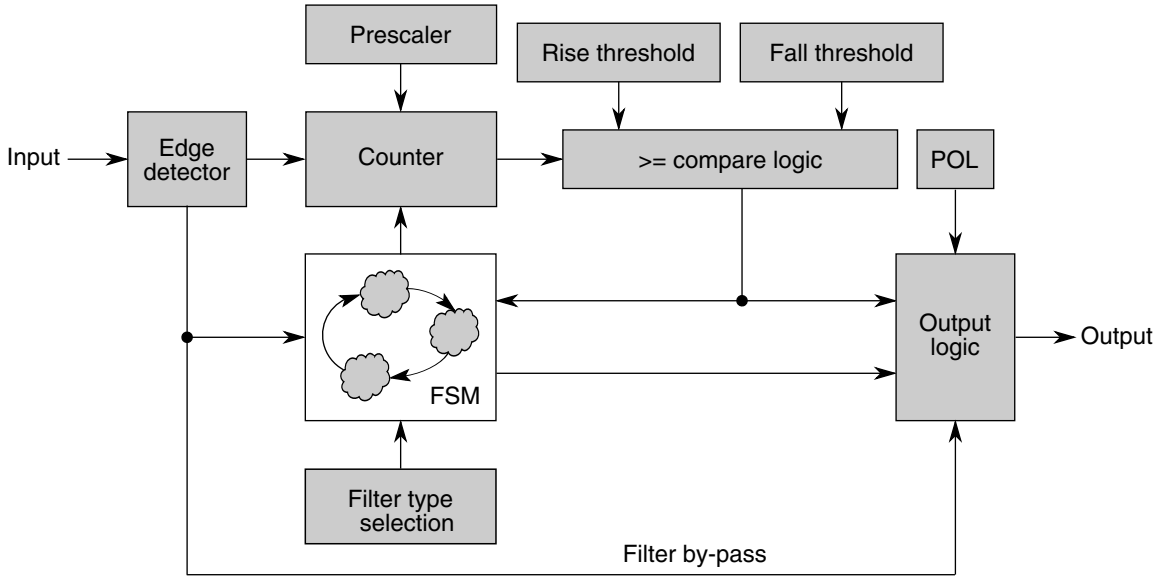


Figure 45-3. Input Glitch Filter block diagram

45.5 Input Glitch Filter overview

The Input Glitch Filter receives an input signal usually from a device PAD. Signals generated externally to the device are affected by environmental noise or glitches. The signal level has to be evaluated with some criteria in order to remove the noise which could affect the measurements done over this signal, such as transition time, period and level.

In a typical application, the Input Glitch Filter interfaces with one or more timer peripherals.

The strategy to remove glitches from an input signal is to monitor the level of the signal during a wide time window thus the glitches can be evaluated and removed.

The input signal should be stable for a certain amount of time before it can be used as a trigger or by a timer input channel. This strategy implies a delay on detecting a transition in the signal.

There is another strategy which is based on removing the glitches by introducing a blanking period after an edge is detected. In this strategy there is no delay from the edge detection in the input signal to the filtered signal, in the filter output. The strategy also allows input signals with different filter periods to be measured at the same time.

45.5.1 Interaction with other peripherals

The figure below shows an example of how the Input Glitch Filter interacts with on-chip peripherals.

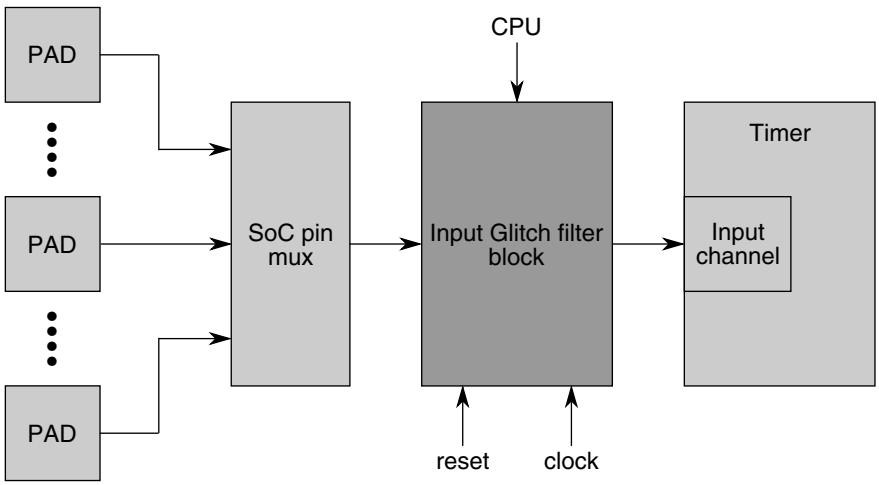


Figure 45-4. Input Glitch Filter application example

45.5.2 Mode of operation

The Input Glitch Filter has the following modes of operation:

- Configuration mode
- Normal mode
- Debug mode
- Low power mode

45.5.2.1 Configuration mode

The configuration mode is the initial mode of the Input Glitch filter after system reset is applied. The configuration mode is selected when FGEN = ‘0’ in the MCR register. The filter parameters such as rising and falling thresholds can be programmed when the filter is in configuration mode without disturbing the internal device logic since the filter output is kept at logic '0' state. Once FGEN is programmed with logic ‘1’ the filter enters the normal operating mode and is able to initiate the filtering process.

NOTE

The MDIS bit in the MCR register, if set to logic '1', makes the filter transition to low power mode independent of FGEN bit settings. Thus, in order to enter the configuration mode MDIS bit needs to be at logic '0'.

45.5.2.2 Normal mode

The normal mode of operation is entered when MCR register FGEN = ‘1’. In this mode the filter implements one of the filter types:

- Bypass filter
- Windowing filter
- Integrating filter
- Integrating-hold filter

These filter types are described in the following sections. Before entering the normal mode, the filter parameters, such as rise and fall thresholds, need to be programmed in the appropriate registers, as well as the type needs to be defined in the corresponding register.

NOTE

In the normal mode it is still possible to change the filter parameters, such as rise and fall thresholds. See section On-the-fly filter reconfiguration for more details.

45.5.2.3 Debug mode

The debug mode is entered when the MCR FRZ bit is set and a system level debug request occurs. Once the debug mode is entered, the internal filter counter is halted and the debug functions controlled by the FBP, FOH, and FOL bits are enabled. The filter output is frozen and remains at the state it was immediately before the debug mode was entered. The FOH and FOL MCR bits can be used to force the filter output to logic '1' or '0', respectively. These bits are self-negating, thus they always read as zero.

By setting the MCR FBP bit, the filter input state is copied to the filter output, thus allowing to control the filter output based on the filter input state. The output is forced during one clock cycle. After that the filter output remains at the new forced state until it is changed again by using a debug control bit or if the filter returns to the normal mode of operation.

NOTE

The system level debug request occurs when a specific event, or a series of events, occurs in one of the device sub-modules and is interpreted by the device debug logic as a debug request. The Input Glitch Filter is not capable of generating an event that causes the assertion of the system level debug request.

45.5.2.4 Low power mode

The low power mode is controlled by the MDIS bit in the MCR register or by the STOP signal which is the system level low power control signal. When entering low power mode the filter output remains unchanged at the state it was just before the low power mode was entered. Since the Input Glitch Filter clocks are stopped in this mode, the internal counter remains inactive. The description of this mode is covered in the Low Power Mode section of this document.

45.6 Memory map

The Input Glitch Filter registers are implemented in 32-bit. 8-bit write operations are allowed. The coherency in case of 8-bit access must be provided by the application software since there is no coherence mechanism built into the hardware.

NOTE

RESERVED bit registers, when available for write operations, should be written always with logic ‘0’. Read access does not return meaningful data from RESERVED bit registers.

IGF memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (IGF_MCR0)	32	R/W	0000_0000h	45.6.1/1761
4	Module Status Register (IGF_MSR0)	32	R/W	0000_0000h	45.6.2/1764
10	Prescaler Register (IGF_PRESR0)	32	R/W	0000_0000h	45.6.3/1765
1C	Rising edge Threshold Register (IGF_RTHR0)	32	R/W	0000_0000h	45.6.4/1766
20	Falling edge Threshold Register (IGF_FTHR0)	32	R/W	0000_0000h	45.6.5/1766
40	Module Configuration Register (IGF_MCR1)	32	R/W	0000_0000h	45.6.1/1761
44	Module Status Register (IGF_MSR1)	32	R/W	0000_0000h	45.6.2/1764
5C	Rising edge Threshold Register (IGF_RTHR1)	32	R/W	0000_0000h	45.6.4/1766
60	Falling edge Threshold Register (IGF_FTHR1)	32	R/W	0000_0000h	45.6.5/1766
80	Module Configuration Register (IGF_MCR2)	32	R/W	0000_0000h	45.6.1/1761
84	Module Status Register (IGF_MSR2)	32	R/W	0000_0000h	45.6.2/1764
9C	Rising edge Threshold Register (IGF_RTHR2)	32	R/W	0000_0000h	45.6.4/1766
A0	Falling edge Threshold Register (IGF_FTHR2)	32	R/W	0000_0000h	45.6.5/1766
C0	Module Configuration Register (IGF_MCR3)	32	R/W	0000_0000h	45.6.1/1761
C4	Module Status Register (IGF_MSR3)	32	R/W	0000_0000h	45.6.2/1764

Table continues on the next page...

IGF memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
DC	Rising edge Threshold Register (IGF_RTHR3)	32	R/W	0000_0000h	45.6.4/1766
E0	Falling edge Threshold Register (IGF_FTHR3)	32	R/W	0000_0000h	45.6.5/1766
100	Module Configuration Register (IGF_MCR4)	32	R/W	0000_0000h	45.6.1/1761
104	Module Status Register (IGF_MSR4)	32	R/W	0000_0000h	45.6.2/1764
11C	Rising edge Threshold Register (IGF_RTHR4)	32	R/W	0000_0000h	45.6.4/1766
120	Falling edge Threshold Register (IGF_FTHR4)	32	R/W	0000_0000h	45.6.5/1766
140	Module Configuration Register (IGF_MCR5)	32	R/W	0000_0000h	45.6.1/1761
144	Module Status Register (IGF_MSR5)	32	R/W	0000_0000h	45.6.2/1764
15C	Rising edge Threshold Register (IGF_RTHR5)	32	R/W	0000_0000h	45.6.4/1766
160	Falling edge Threshold Register (IGF_FTHR5)	32	R/W	0000_0000h	45.6.5/1766
180	Module Configuration Register (IGF_MCR6)	32	R/W	0000_0000h	45.6.1/1761
184	Module Status Register (IGF_MSR6)	32	R/W	0000_0000h	45.6.2/1764
19C	Rising edge Threshold Register (IGF_RTHR6)	32	R/W	0000_0000h	45.6.4/1766
1A0	Falling edge Threshold Register (IGF_FTHR6)	32	R/W	0000_0000h	45.6.5/1766
1C0	Module Configuration Register (IGF_MCR7)	32	R/W	0000_0000h	45.6.1/1761
1C4	Module Status Register (IGF_MSR7)	32	R/W	0000_0000h	45.6.2/1764
1DC	Rising edge Threshold Register (IGF_RTHR7)	32	R/W	0000_0000h	45.6.4/1766
1E0	Falling edge Threshold Register (IGF_FTHR7)	32	R/W	0000_0000h	45.6.5/1766
200	Module Configuration Register (IGF_MCR8)	32	R/W	0000_0000h	45.6.1/1761
204	Module Status Register (IGF_MSR8)	32	R/W	0000_0000h	45.6.2/1764
21C	Rising edge Threshold Register (IGF_RTHR8)	32	R/W	0000_0000h	45.6.4/1766
220	Falling edge Threshold Register (IGF_FTHR8)	32	R/W	0000_0000h	45.6.5/1766
240	Module Configuration Register (IGF_MCR9)	32	R/W	0000_0000h	45.6.1/1761
244	Module Status Register (IGF_MSR9)	32	R/W	0000_0000h	45.6.2/1764
25C	Rising edge Threshold Register (IGF_RTHR9)	32	R/W	0000_0000h	45.6.4/1766
260	Falling edge Threshold Register (IGF_FTHR9)	32	R/W	0000_0000h	45.6.5/1766
280	Module Configuration Register (IGF_MCR10)	32	R/W	0000_0000h	45.6.1/1761
284	Module Status Register (IGF_MSR10)	32	R/W	0000_0000h	45.6.2/1764
29C	Rising edge Threshold Register (IGF_RTHR10)	32	R/W	0000_0000h	45.6.4/1766
2A0	Falling edge Threshold Register (IGF_FTHR10)	32	R/W	0000_0000h	45.6.5/1766
2C0	Module Configuration Register (IGF_MCR11)	32	R/W	0000_0000h	45.6.1/1761
2C4	Module Status Register (IGF_MSR11)	32	R/W	0000_0000h	45.6.2/1764
2DC	Rising edge Threshold Register (IGF_RTHR11)	32	R/W	0000_0000h	45.6.4/1766
2E0	Falling edge Threshold Register (IGF_FTHR11)	32	R/W	0000_0000h	45.6.5/1766
300	Module Configuration Register (IGF_MCR12)	32	R/W	0000_0000h	45.6.1/1761
304	Module Status Register (IGF_MSR12)	32	R/W	0000_0000h	45.6.2/1764
31C	Rising edge Threshold Register (IGF_RTHR12)	32	R/W	0000_0000h	45.6.4/1766
320	Falling edge Threshold Register (IGF_FTHR12)	32	R/W	0000_0000h	45.6.5/1766
340	Module Configuration Register (IGF_MCR13)	32	R/W	0000_0000h	45.6.1/1761

Table continues on the next page...

IGF memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
344	Module Status Register (IGF_MSR13)	32	R/W	0000_0000h	45.6.2/1764
35C	Rising edge Threshold Register (IGF_RTHR13)	32	R/W	0000_0000h	45.6.4/1766
360	Falling edge Threshold Register (IGF_FTHR13)	32	R/W	0000_0000h	45.6.5/1766
380	Module Configuration Register (IGF_MCR14)	32	R/W	0000_0000h	45.6.1/1761
384	Module Status Register (IGF_MSR14)	32	R/W	0000_0000h	45.6.2/1764
39C	Rising edge Threshold Register (IGF_RTHR14)	32	R/W	0000_0000h	45.6.4/1766
3A0	Falling edge Threshold Register (IGF_FTHR14)	32	R/W	0000_0000h	45.6.5/1766
3C0	Module Configuration Register (IGF_MCR15)	32	R/W	0000_0000h	45.6.1/1761
3C4	Module Status Register (IGF_MSR15)	32	R/W	0000_0000h	45.6.2/1764
3DC	Rising edge Threshold Register (IGF_RTHR15)	32	R/W	0000_0000h	45.6.4/1766
3E0	Falling edge Threshold Register (IGF_FTHR15)	32	R/W	0000_0000h	45.6.5/1766
400	Module Configuration Register (IGF_MCR16)	32	R/W	0000_0000h	45.6.1/1761
404	Module Status Register (IGF_MSR16)	32	R/W	0000_0000h	45.6.2/1764
41C	Rising edge Threshold Register (IGF_RTHR16)	32	R/W	0000_0000h	45.6.4/1766
420	Falling edge Threshold Register (IGF_FTHR16)	32	R/W	0000_0000h	45.6.5/1766
440	Module Configuration Register (IGF_MCR17)	32	R/W	0000_0000h	45.6.1/1761
444	Module Status Register (IGF_MSR17)	32	R/W	0000_0000h	45.6.2/1764
45C	Rising edge Threshold Register (IGF_RTHR17)	32	R/W	0000_0000h	45.6.4/1766
460	Falling edge Threshold Register (IGF_FTHR17)	32	R/W	0000_0000h	45.6.5/1766
480	Module Configuration Register (IGF_MCR18)	32	R/W	0000_0000h	45.6.1/1761
484	Module Status Register (IGF_MSR18)	32	R/W	0000_0000h	45.6.2/1764
49C	Rising edge Threshold Register (IGF_RTHR18)	32	R/W	0000_0000h	45.6.4/1766
4A0	Falling edge Threshold Register (IGF_FTHR18)	32	R/W	0000_0000h	45.6.5/1766
4C0	Module Configuration Register (IGF_MCR19)	32	R/W	0000_0000h	45.6.1/1761
4C4	Module Status Register (IGF_MSR19)	32	R/W	0000_0000h	45.6.2/1764
4DC	Rising edge Threshold Register (IGF_RTHR19)	32	R/W	0000_0000h	45.6.4/1766
4E0	Falling edge Threshold Register (IGF_FTHR19)	32	R/W	0000_0000h	45.6.5/1766
500	Module Configuration Register (IGF_MCR20)	32	R/W	0000_0000h	45.6.1/1761
504	Module Status Register (IGF_MSR20)	32	R/W	0000_0000h	45.6.2/1764
51C	Rising edge Threshold Register (IGF_RTHR20)	32	R/W	0000_0000h	45.6.4/1766
520	Falling edge Threshold Register (IGF_FTHR20)	32	R/W	0000_0000h	45.6.5/1766
540	Module Configuration Register (IGF_MCR21)	32	R/W	0000_0000h	45.6.1/1761
544	Module Status Register (IGF_MSR21)	32	R/W	0000_0000h	45.6.2/1764
55C	Rising edge Threshold Register (IGF_RTHR21)	32	R/W	0000_0000h	45.6.4/1766
560	Falling edge Threshold Register (IGF_FTHR21)	32	R/W	0000_0000h	45.6.5/1766
580	Module Configuration Register (IGF_MCR22)	32	R/W	0000_0000h	45.6.1/1761
584	Module Status Register (IGF_MSR22)	32	R/W	0000_0000h	45.6.2/1764
59C	Rising edge Threshold Register (IGF_RTHR22)	32	R/W	0000_0000h	45.6.4/1766
5A0	Falling edge Threshold Register (IGF_FTHR22)	32	R/W	0000_0000h	45.6.5/1766

Table continues on the next page...

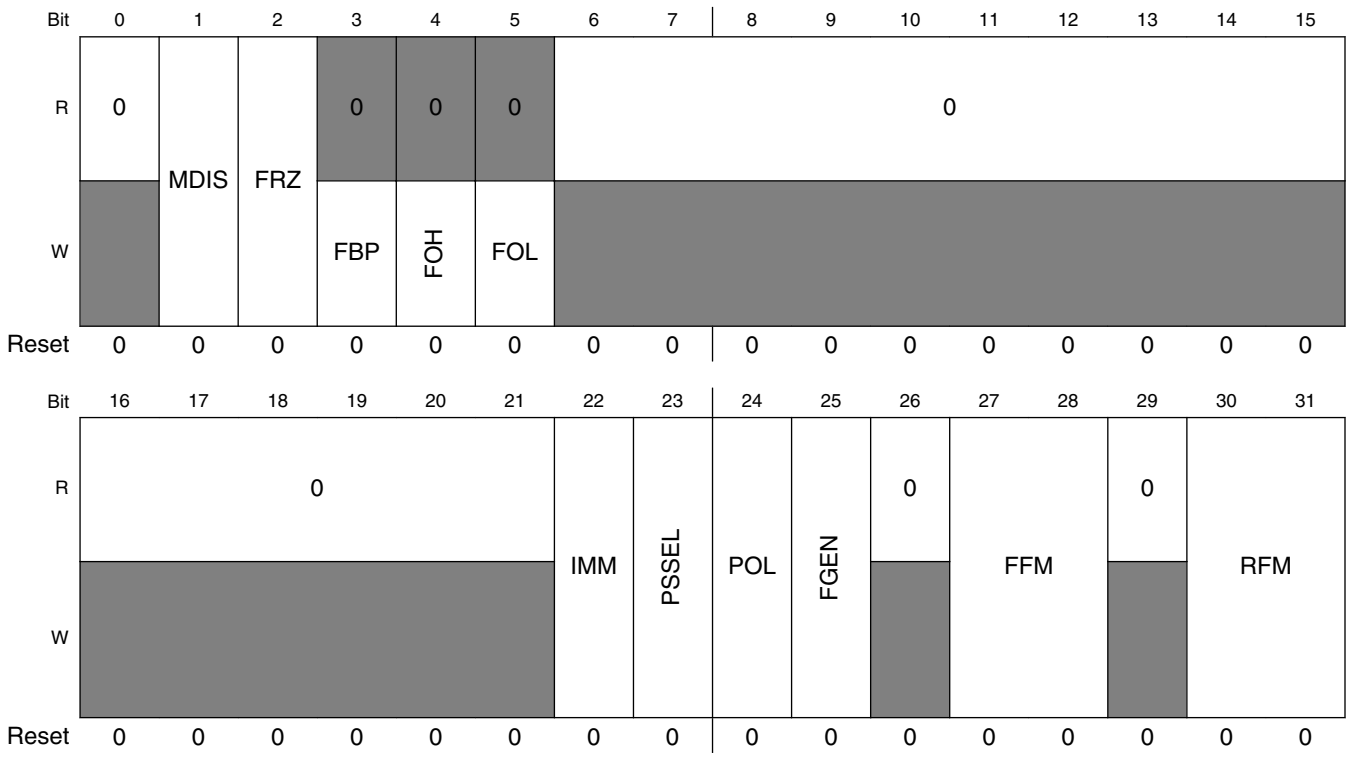
IGF memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5C0	Module Configuration Register (IGF_MCR23)	32	R/W	0000_0000h	45.6.1/1761
5C4	Module Status Register (IGF_MSR23)	32	R/W	0000_0000h	45.6.2/1764
5DC	Rising edge Threshold Register (IGF_RTHR23)	32	R/W	0000_0000h	45.6.4/1766
5E0	Falling edge Threshold Register (IGF_FTHR23)	32	R/W	0000_0000h	45.6.5/1766
600	Module Configuration Register (IGF_MCR24)	32	R/W	0000_0000h	45.6.1/1761
604	Module Status Register (IGF_MSR24)	32	R/W	0000_0000h	45.6.2/1764
61C	Rising edge Threshold Register (IGF_RTHR24)	32	R/W	0000_0000h	45.6.4/1766
620	Falling edge Threshold Register (IGF_FTHR24)	32	R/W	0000_0000h	45.6.5/1766
640	Module Configuration Register (IGF_MCR25)	32	R/W	0000_0000h	45.6.1/1761
644	Module Status Register (IGF_MSR25)	32	R/W	0000_0000h	45.6.2/1764
65C	Rising edge Threshold Register (IGF_RTHR25)	32	R/W	0000_0000h	45.6.4/1766
660	Falling edge Threshold Register (IGF_FTHR25)	32	R/W	0000_0000h	45.6.5/1766
680	Module Configuration Register (IGF_MCR26)	32	R/W	0000_0000h	45.6.1/1761
684	Module Status Register (IGF_MSR26)	32	R/W	0000_0000h	45.6.2/1764
69C	Rising edge Threshold Register (IGF_RTHR26)	32	R/W	0000_0000h	45.6.4/1766
6A0	Falling edge Threshold Register (IGF_FTHR26)	32	R/W	0000_0000h	45.6.5/1766
6C0	Module Configuration Register (IGF_MCR27)	32	R/W	0000_0000h	45.6.1/1761
6C4	Module Status Register (IGF_MSR27)	32	R/W	0000_0000h	45.6.2/1764
6DC	Rising edge Threshold Register (IGF_RTHR27)	32	R/W	0000_0000h	45.6.4/1766
6E0	Falling edge Threshold Register (IGF_FTHR27)	32	R/W	0000_0000h	45.6.5/1766
700	Module Configuration Register (IGF_MCR28)	32	R/W	0000_0000h	45.6.1/1761
704	Module Status Register (IGF_MSR28)	32	R/W	0000_0000h	45.6.2/1764
71C	Rising edge Threshold Register (IGF_RTHR28)	32	R/W	0000_0000h	45.6.4/1766
720	Falling edge Threshold Register (IGF_FTHR28)	32	R/W	0000_0000h	45.6.5/1766
740	Module Configuration Register (IGF_MCR29)	32	R/W	0000_0000h	45.6.1/1761
744	Module Status Register (IGF_MSR29)	32	R/W	0000_0000h	45.6.2/1764
75C	Rising edge Threshold Register (IGF_RTHR29)	32	R/W	0000_0000h	45.6.4/1766
760	Falling edge Threshold Register (IGF_FTHR29)	32	R/W	0000_0000h	45.6.5/1766
780	Module Configuration Register (IGF_MCR30)	32	R/W	0000_0000h	45.6.1/1761
784	Module Status Register (IGF_MSR30)	32	R/W	0000_0000h	45.6.2/1764
79C	Rising edge Threshold Register (IGF_RTHR30)	32	R/W	0000_0000h	45.6.4/1766
7A0	Falling edge Threshold Register (IGF_FTHR30)	32	R/W	0000_0000h	45.6.5/1766
7C0	Module Configuration Register (IGF_MCR31)	32	R/W	0000_0000h	45.6.1/1761
7C4	Module Status Register (IGF_MSR31)	32	R/W	0000_0000h	45.6.2/1764
7DC	Rising edge Threshold Register (IGF_RTHR31)	32	R/W	0000_0000h	45.6.4/1766
7E0	Falling edge Threshold Register (IGF_FTHR31)	32	R/W	0000_0000h	45.6.5/1766

45.6.1 Module Configuration Register (IGF_MCR*n*)

The Module Configuration register (MCR) provides access to low power, debug, and filter control bits.

Address: 0h base + 0h offset + (64d × i), where i=0d to 31d



IGF_MCR*n* field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 MDIS	Module Disable The MDIS bit forces the filter to enter a low power state. After this bit is asserted the clock for the module is removed thus filter operation is not possible. The access to filter registers is allowed. 0 Input Glitch Filter is not in low power mode due to MDIS bit 1 Input Glitch Filter is in low power mode
2 FRZ	Freeze bit for debug operation The FRZ bit enables the Input Glitch Filter to enter freeze mode (debug mode) when a system level signal request the module enters freeze (debug) mode. Once in freeze mode the filter operation is disabled meaning that transitions on the input signal will not propagate through the filter output. The bypass FBP bit can be used by the software to force the filter output to be the same as the filter input. In this mode FOH and FOL also can be used to define the filter output state.

Table continues on the next page...

IGF_MCR*n* field descriptions (continued)

Field	Description
	0 Freeze mode disabled 1 Freeze mode enabled
3 FBP	Force Bypass This bit is self-negated thus is only read as zero. Writing 1 to this bit while in freeze mode forces the filter to bypass the value in the input to the filter output, thus allowing a software control on the filter. This bit is functional only in freeze mode. 0 no filter bypass 1 transfer input signal value to the filter output
4 FOH	Force filter output high If set, this bit forces the Input Glitch Filter output to '1'. This bit is functional only in debug mode. It is self-negated thus always read as zero. 0 no action 1 forces the filter output to '1'
5 FOL	Force filter output low If set this bit forces the Input Glitch Filter output to '0'. This bit is functional only in debug mode. It is self-negated thus always read as zero. 0 no action 1 forces the filter output to '0'
6–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 IMM	Immediate edge propagation control bit The IMM bit controls the propagation of an edge through the filter. If asserted, this bit defines that an edge at the filter input propagates through the filter output independent of the prescaler settings. In bypass mode the propagation delay is three system clock cycles. If negated, the signal propagation depends upon the prescaler settings. The filter hardware will only allow chages to this bit if the filter is disabled, FGEN = 0 0 edge propagation depends on prescaler 1 edge propagation within three system clock cycles
23 PSSEL	Prescaler selection bit The PSSEL selects if the prescaler used by the Input Glitch Filter is from internal source (internal prescaler counter) or from external source. The filter hardware will only allow changes to this bit if the filter is disabled, FGEN=0. 0 internal prescaler selected 1 external prescaler selected
24 POL	Output polarity bit This bit sets the polarity of the filter output to inverted or not inverted. The polarity control implemented by the POL bit is applied at the filter output stage thus also changing the reset state of the output, even if the filter is disabled by the FGEN bit. The filter hardware will only allow changes to this bit if the filter is disabled, FGEN=0.

Table continues on the next page...

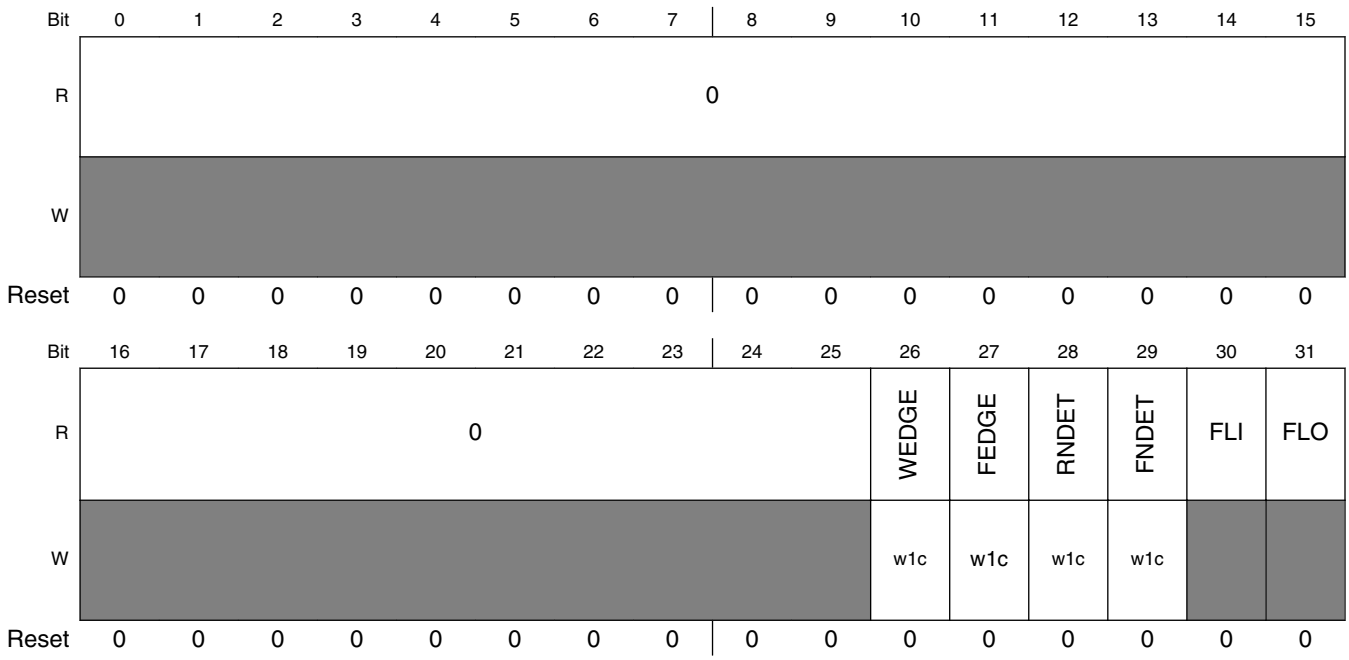
IGF_MCR*n* field descriptions (continued)

Field	Description
	0 not inverted output 1 inverted output
25 FGEN	Filter global enable This control bit enables the filter operation, meaning that if the filter is enabled the filtering selected types are applied to the rising and falling edges of the input signal. If the filter is not enabled, then the filter output remains unchanged independent of the filter input signal. Note that the filter output state is affected by the POL bit when the filter is disabled. 0 filter is disabled: filter output keeps current state. 1 filter is enabled: filtering is applied to selected edges.
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–28 FFM	Falling edge filter type selection This bit field selects the filter type for the filter input signal falling edge. The filter hardware will only allow changes to this bit if the filter is disabled, FGEN=0. 00 bypass: edge is propagated to filter output without any filtering in three system clock cycles 01 windowing: windowing filter is selected for the falling edge 10 integrating: integrating filter is selected for the falling edge 11 integrating-hold: integrating-hold filter is selected for the falling edge
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 RFM	Rising edge filter type selection This bit field selects the filter type for the input signal rising edge. The filter hardware will only allow changes to this bit if the filter is disabled, FGEN=0. 00 bypass: edge is propagated to filter output without any filtering in three system clock cycles 01 windowing: windowing filter is selected for the rising edge 10 integrating: integrating filter is selected for the rising edge 11 integrating-hold: integrating-hold filter is selected for the rising edge

45.6.2 Module Status Register (IGF_MSR*n*)

The Module status register (MSR) provides access to filter status bits and corresponding clear bits as well as access to the state of filter input and output signals.

Address: 0h base + 4h offset + (64d × i), where i=0d to 31d



IGF_MSR*n* field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 WEDGE	Filter is active waiting for an edge This bit is set when the filter is enabled but the internal counter is not active. It indicates that the filter is waiting for an edge to occur at the filter input. It is set by the Input Glitch Filter hardware and cleared by the software by writing 1 to the bit. Since the intention of this bit is to capture the dynamic of the filter hardware, it should be cleared after the read thus being able to capture a new event. 0 Input Glitch Filter is disabled or the internal counter is active 1 Input Glitch Filter is waiting for an edge
27 FEDGE	Filter is active processing an edge This bit is set during the filter active state thus when the filter internal counter is counting, processing an edge in a filtering mode. It is cleared by the software by writing 1 to the bit. Since the intention of this bit is to capture the dynamic of the filter hardware, it should be cleared after the read thus being able to capture a new event. 0 Input Glitch Filter internal counter is not active 1 Input Glitch Filter internal counter is active

Table continues on the next page...

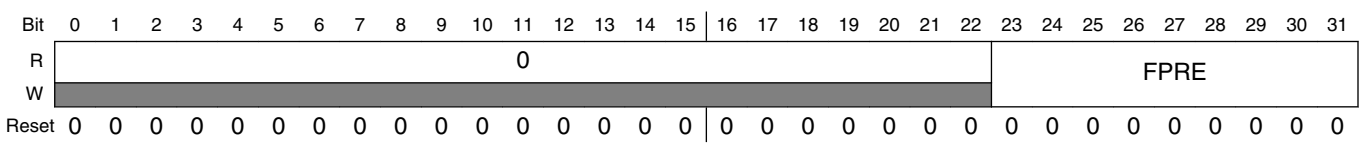
IGF_MSR*n* field descriptions (continued)

Field	Description
28 RNDET	Rise noise detected bit This bit indicates that the filter filtered out noise in the input signal after a rising edge was detected. The criteria to set this bit is when a rising edge is detected thus starting the filter counter and before this edge pass through the filter (if filter is in integrating filter types) or during a window (if filter is in windowing filter types) another edge occurs in the input signal. This bit is set by a noise detection and cleared by the software by writing 1 to this bit. 0 noise was not detected 1 noise was detected after a rising edge
29 FNDET	Fall noise detected bit This bit indicates that the filter filtered out noise in the input signal after a falling edge was detected. The criteria to set this bit is when a falling edge is detected, thus starting the filter counter, and, if the filter is in integrating types, before this edge passes through the filter, or, in windowing types, when another edge occurs in the input signal during a window. This bit is set by a noise detection and cleared by the software by writing 1 to this bit. 0 noise was not detected 1 noise was detected after a falling edge
30 FLI	Filter input The FLI bit follows the state of the filter input value as captured by the filter input logic, which is affected by the IMM bit and prescaler settings.
31 FLO	Filter output The FLO bit follows the state of the filter output value considering the POL bit state.

45.6.3 Prescaler Register (IGF_PRESR*n*)

The prescaler register (PRESR) provides access to the prescaler bits.

Address: 0h base + 10h offset + (64d × i), where i=0d to 0d



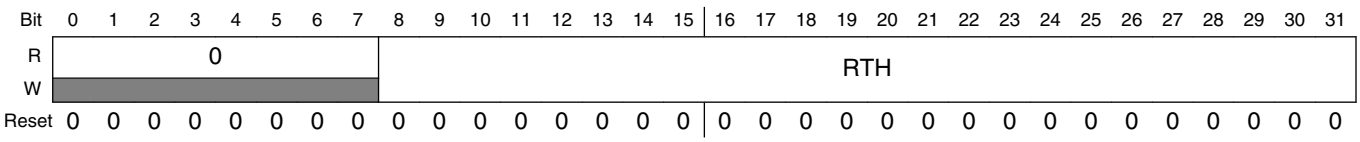
IGF_PRESR*n* field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 FPRE	Filter prescaler The prescaler defines the rate of the filter counter FCOUNT. The filter counter clock rate is defined by the formula: Filter counter clock = module clock / (FPRE + 1).

45.6.4 Rising edge Threshold Register (IGF_RTHR*n*)

The Rising edge Threshold register (RTHR) provides access to the rising edge threshold bits.

Address: 0h base + 1Ch offset + (64d × i), where i=0d to 31d



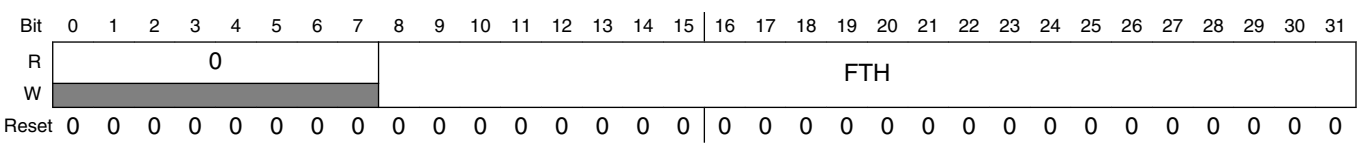
IGF_RTHR*n* field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 RTH	Rising edge threshold This bit field defines the filter counter threshold when a rising edge occurs is being filtered.

45.6.5 Falling edge Threshold Register (IGF_FTHR*n*)

The Falling edge Threshold register (FTHR) provides access to the falling edge threshold bits.

Address: 0h base + 20h offset + (64d × i), where i=0d to 31d



IGF_FTHR*n* field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 FTH	Falling edge threshold This bit field defines the filter counter threshold when a falling edge is being filtered.

45.7 Functional description

The following sections describe the Input Glitch Filter module functionality.

45.7.1 Input double sampling

The filter input is double sampled for clock domain translation thus avoiding asynchronous signals to propagate through the filter internal logic. This stage is composed by two flip-flops connected back to back. The delay caused by this logic is therefore two clock cycles.

Note that due to the synchronous nature of the Input Glitch Filter design, if a glitch smaller than one clock cycle occurs at the filter input it will not propagate to the filter internal logic and thus will not propagate to the filter output. Also, if clock is not applied to the filter (i.e. in low power mode) any transition at the filter input will not propagate to the filter output.

45.7.2 Filter types

The Input Glitch Filter implements three main filter types:

- Windowing
- Integrating
- Integrating hold

These filter types can be selected for the rising and falling edges independently, thus allowing different filter types to be selected for different edges (i.e. rising edge filtered with windowing filter and falling edge filtered with integrating).

45.7.2.1 Windowing filter

This filter type detects an edge (positive or negative edge) at the filter input and then mask out detection of subsequent edges over a period of time, which is defined as the blanking period, thus effectively behaving as a high pass filter. After an edge is detected the level at the filter input is propagated to the filter output within three clock cycles. The delay is caused by the synchronization stage at the input and by the filter output register stage. During the blanking period any transition at the filter input is disregarded. After the blanking period has elapsed, the filter compares the input level state with the filter output state. In case they are different a transition is inferred and re-evaluate according to the filter type assigned to that transition (positive or negative edges). Since, in this example, the windowing filter type is programed for both edges, the inferred negative edge is propagated to the filter output and another blanking period is initiated. The figure below describes the windowing filter type behavior.

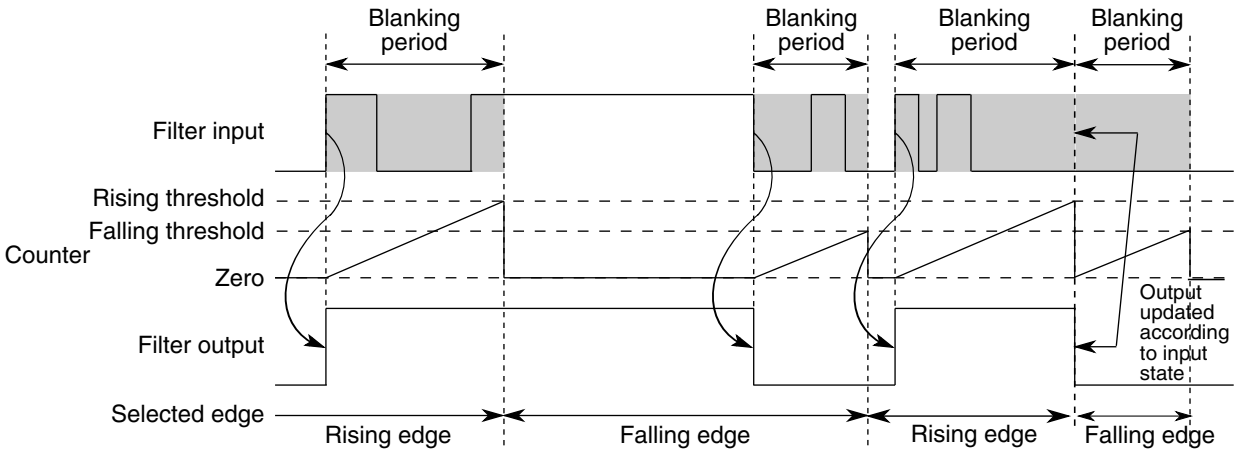


Figure 45-139. Windowing filter type

45.7.2.2 Integrating filter

The integrating filter behaves like a low-pass filter, thus removing high frequency bursts from the input signal. The input and output signals are compared and if they are at different states an internal filter counter is enabled.

The direction of the counter (up or down counting) is defined by the states of the input and output filter signals as follows:

- Filter output is at the same state as the filter input: down-counting
- Filter output is at different state as the filter input: up-counting

The counter is compared with the rising or falling threshold register according to:

- Rising threshold register is selected if the rising edge is being filtered.
- Falling threshold register is selected if the falling edge is being filtered.

The filter output changes state to the same state as the filter input:

- If filter counter reaches the selected threshold for the edge being filtered.

The internal counter is disabled and cleared:

- If filter counter reaches the selected threshold for the edge being filtered.

The figure below describes an example of the Integrating filter type operation.

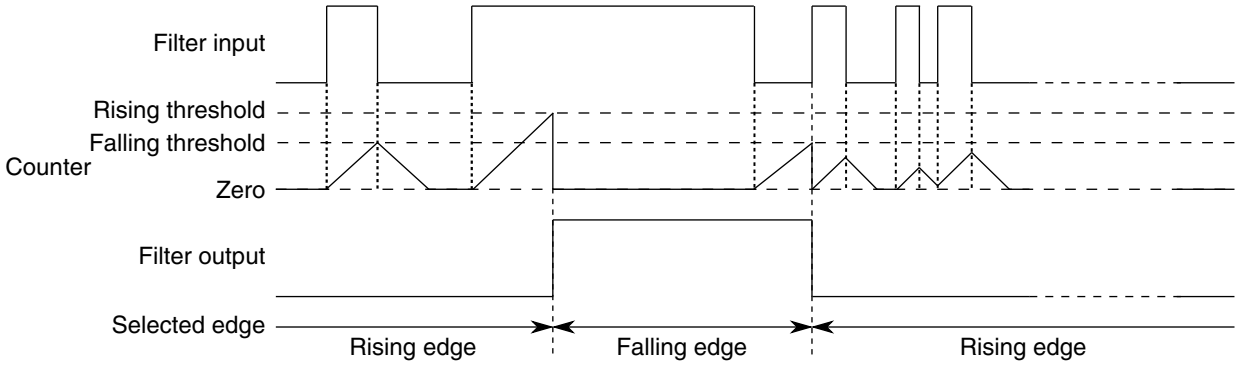


Figure 45-140. Integrating filter type

45.7.2.3 Integrating-hold filter

The integrating-hold filter type is similar to the integrating filter. The difference is the counter, which only counts in up direction instead of up-down in the integrating mode. In this filter configuration the input and output signals are compared and if they are at different states the filter counter is enabled. This counter is controlled by the state of the filter input and filter output signals as follows:

- Filter output is at the same state as the filter input: counter is in hold state (no-counting).
- Filter output is at different state as the filter input: counter is counting up.

The counter is compared with the rising or falling threshold register according to:

- Rising threshold register is selected if the rising edge is being filtered.
- Falling threshold register is selected if the falling edge is being filtered.

The filter output changes state to the same state as the filter input:

- If the filter counter reaches the selected threshold for the edge being filtered.

The internal counter is disabled and cleared:

- If the filter counter reaches the selected threshold for the edge being filtered.

The figure below shows an example of the Integrating-hold filter type operation.

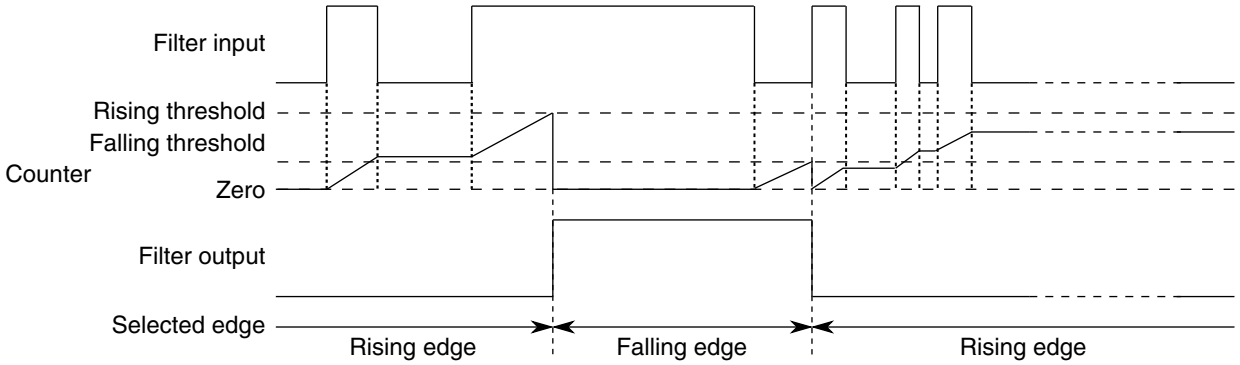


Figure 45-141. Integrating-hold filter type

45.7.2.4 Bypass filter

In the bypass filter type, the Input Glitch Filter immediately propagates any transition on the filter input to the filter output. This filter type does not provide any protection against glitches in the input signal. It is intended to be used in situations where the input signal is already filtered, or it is a glitch free signal, or the filter output is connected to a module that already has filtering capabilities. The bypass filter type is independently selected for rising or falling edges. Thus rising edges may be not filtered by the bypass filter type while the falling edges are filtered by any of the available filter types.

The bypass filter presents a delay in the edge propagation through the filter that is caused by the double sampling at the filter input and by the output filter register. Therefore, the total delay is three clock cycles.

45.7.2.5 Combined filter

The Input Glitch Filter allows the combination of filter types for different edges, positive and negative. As an example, the rising edges may be filtered using the windowing filter while the falling edges are filtered with an integrating filter. This configuration of the Input Glitch Filter is provided for situations where the noise present in the rising edges is different from the noise in the falling edges. Another use case is when the delay requirements of the input signal is different for rising and falling edges.

The figure below shows the combined filter operation. The windowing filter type is applied to the rising edges. The integrating filter is used for the falling edges.

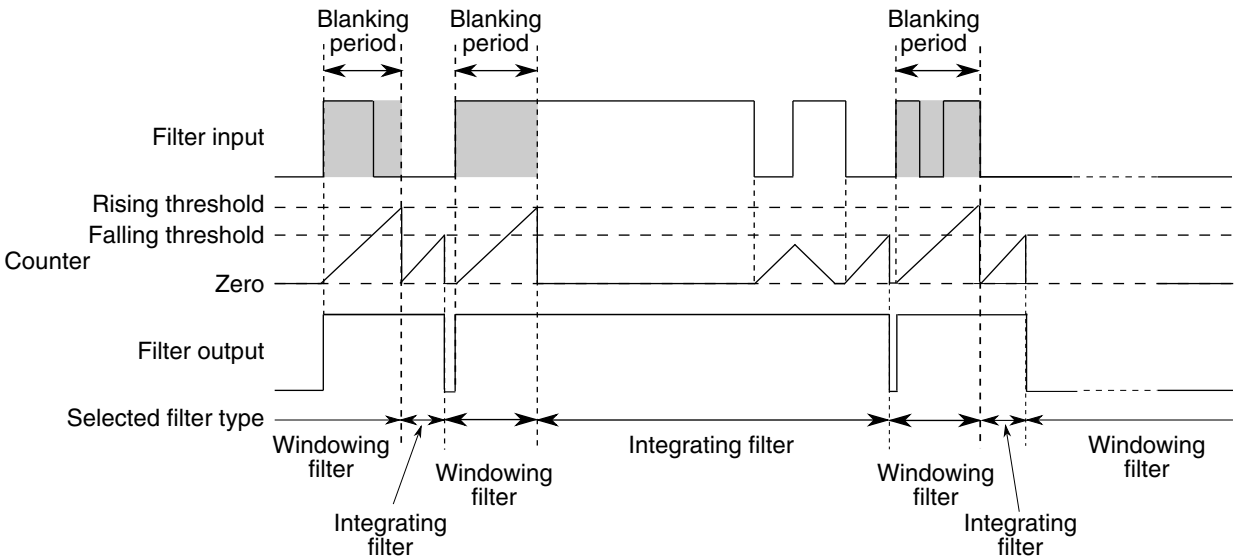


Figure 45-142. Combined filter type (windowing + integrating)

45.7.3 Application

This section describes several applications for the Input Glitch filter module.

45.7.3.1 Application use case 1: integrating filter

The integrating filter type is particularly useful in applications where glitches occur spuriously while the signal is expected to be in a steady state (either 1 or 0). Choosing correct up and down filter clock rates can ensure that the effects of spurious glitches can be completely eliminated, whilst minimizing propagation delays.

The application must be able to tolerate delays between the actual edge occurrence and the time that the signal is propagated through the filter logic. Because the delays are constants defined by the detection thresholds, application software may compensate for them, provided little or no noise occurs on the detected edge.

45.7.3.2 Application use case 2: windowing filter

The windowing filter type is particularly useful for removing noise bounce produced by a switch or sensor when its signal changes state. In this case, there is no propagation delay (except for the filter input synchronization logic and output register) between the filter input and filter output signals. This type of filter would not necessarily provide much immunity to spurious noise that occurred during the steady state portion of the signal. In

order to optimize the characteristics of this filter, the blanking period should be of the same magnitude as the expected input signal rate. Signals that change frequency can be accommodated by software by dynamically updating the blanking period.

45.7.3.3 Application use case 3: windowing filter

Safety applications may need to compare the inputs obtained from different input pins, but derived from a single external source. This form of redundancy may detect non-common mode faults that may occur in the portions of the SoC that route and process the incoming signal. Having two different instances of windowing filter used on each input pin will ensure that the signal edge timing for both input paths is equal. Applying different filter characteristics for each input path can enhance the ability to detect SoC fault conditions.

45.7.3.4 Application use case 4: combined filter

Some applications may have rising edge signals that exhibit different characteristics to falling edge signals which might require different filtering to optimize the filtering effect. For example the “first” rising edge may represent a true real time event with a fast slew rate that must be captured without delay, while the falling edge signal may have a slow slew rate with many noisy transitions. The Input Glitch Filter can apply, for example, a windowing filter on the rising edge to ensure no loss of real time information and an integrating filter on the falling edge to eliminate noise from being detected as a new rising transition.

45.7.3.5 Application use case 5: prescaler settings in windowing filter

The prescaler settings influence the timing for an edge to propagate through the filter. In windowing filter the IMM bit in the MCR register controls the way an edge is propagated through the filter. The figure below describes the windowing filter operation with regard to the IMM bit settings.

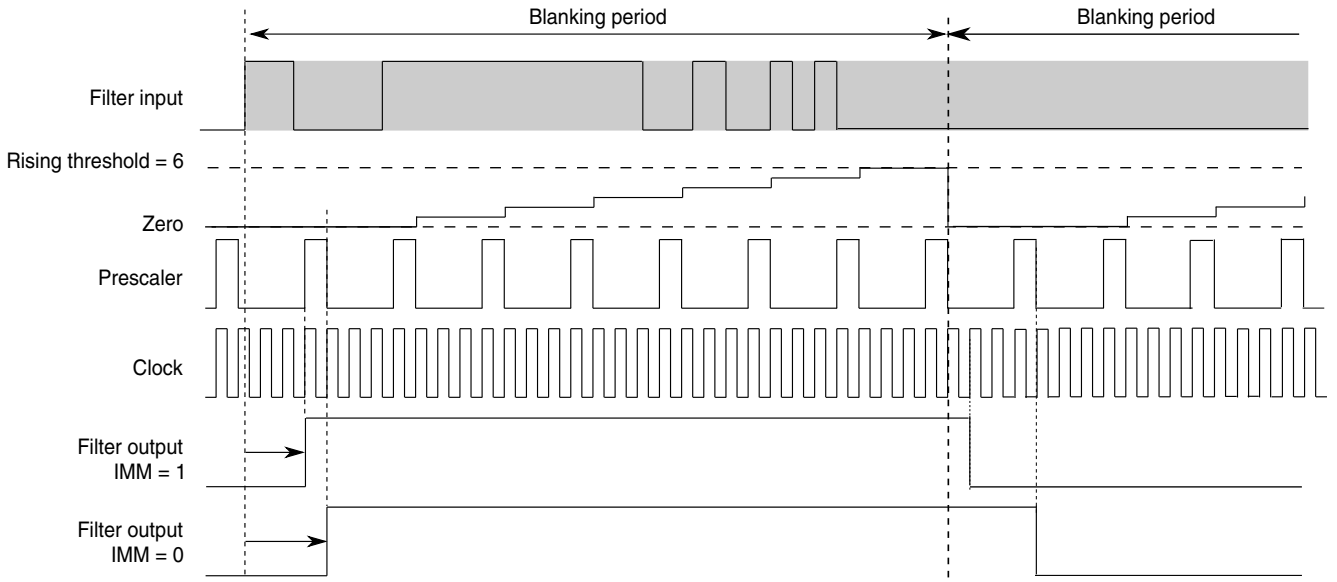


Figure 45-143. Input signal propagation through the filter in windowing filter type

If IMM = 0 then the input signal edges are propagated according to the prescaler ticks. If IMM = 1 then the input edges are propagated immediately (within 3 clock cycles) to the filter output.

45.7.3.6 Application use case 6: prescaler settings in bypass

The prescaler settings influence the timing for an input signal edge to propagate through the filter. In bypass, the IMM bit in the MCR register controls the way an edge is propagated through the filter. If IMM= 0, the prescaled clock is used to sample the input filter signal. The resolution the input signal is sampled is therefore related to the prescaler frequency. Glitches narrower than the prescaler period that are not aligned with the prescaler clock are not captured by the filter. If IMM = 1, the system clock is used to sample the filter input signal instead of the prescaled clock. In this case, the resolution of the sampled input signal is the system clock.

The figure below describes the filter input signal sampling behavior for IMM = 0 and IMM = 1.

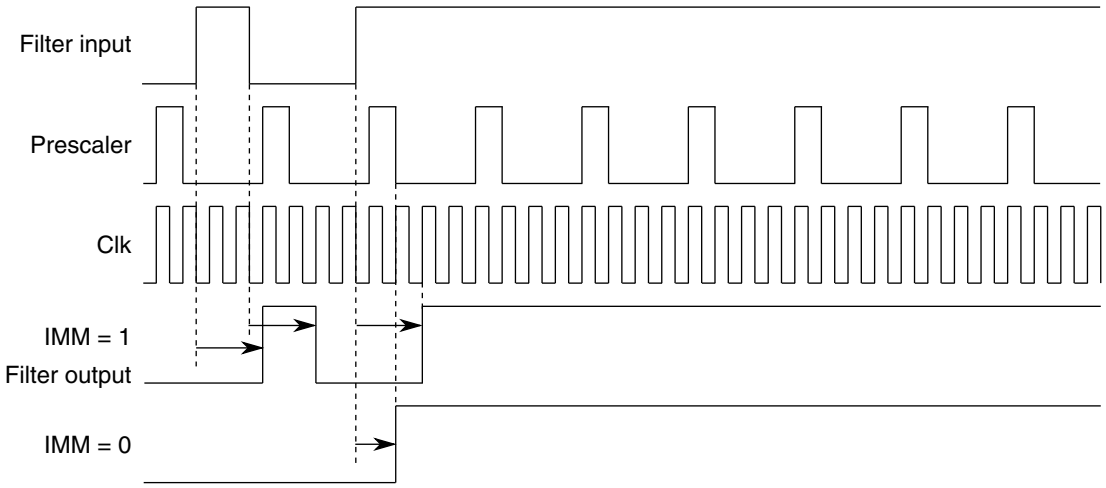


Figure 45-144. Input signal propagation through the filter in bypass

45.7.4 Programming guidelines

The following generalized procedure may be used to configure the Input Glitch Filter Block:

1. Program the SoC pin mux logic to route an external signal to the input of the selected glitch filter.
2. Enable the destination module to receive the Input Glitch filter output.
3. Enable the clock source at the SoC level.
4. Program the Glitch Filter counter prescaler(s) to the desired rate.
5. Select the filter type for each edge using MCR[FFM] and MCR[RFM].
6. Clear any status bits if necessary.
7. Enable the filter using MCR[FGEN]

45.7.5 On-the-fly filter reconfiguration

The architecture of the Input Glitch Filter allows for on the fly reconfiguration of filter parameters such as rise and fall threshold values.

If a parameter is changed it influences the filter operation immediately after the new parameter value was written. The main problem when changing parameters, such as a threshold value, when the filter is enabled is that the filter counter could lose a match due to the parameter change. A great or equal comparator is used in the threshold match logic which allows correct filter operation when the threshold parameters are changed. The figure below shows an example of changing the rising edge parameter value. Note that in this case the match occurs immediately after the parameter is changed since the new parameter value is below the current counter value.

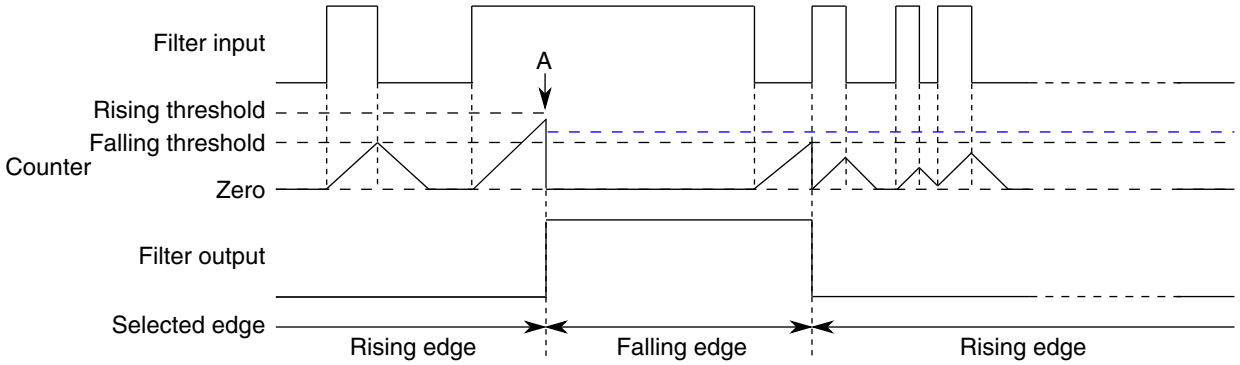


Figure 45-145. On-the-fly rising threshold parameter change

In the case shown in the figure above the match occurs immediately after the new rising threshold is programmed at ‘A’.

NOTE

Note that when changing the threshold values the match may not occur when the counter is equal to the threshold value (the old or the new programmed value). Instead, the match may occur in between the old and new programmed thresholds when the threshold re-programming instant A is done when the counter is greater than the new value. This is a transitory effect which may occur when the threshold value is changed and should be considered by the application.

45.8 Power saving mode

The Input Glitch Filter implements power saving mode thus allowing to reduce power consumption when device inputs connected to the filter are not used or the filter is not necessary to be functioning in a specific SoC mode of operation.

45.8.1 STOP mode

There are two ways for the Input Glitch filter to enter low power mode:

- MDIS bit
- STOP mode system request

The MDIS control bit in the MCR register, when set, disables the filter logic by stopping the module clock signals and thus halting the internal filter counter which reduces the dynamic power consumption of the module. The MDIS bit provides a selective control of the low power operation of each one of the Input Glitch Filter instances in the device.

The STOP mode system request is similar to MDIS operation with exception that it applies to all Input Glitch Filter instances in the device. It does not need to be enabled by any control bit in the Input Glitch filter module. The low power mode in this case is controlled by the assertion of the STOP signal which is a system wide low power control signal.

Chapter 46
Body Cross-triggering Unit (BCTU)

46.1 Body Cross-triggering Unit (BCTU) chip-specific module information

46.1.1 Overview

The BCTU interfaces between the eTPU, eMIOS, PIT, Reaction Module, and the SAR ADCs on the device. This BCTU is a collection of Event Configuration Registers that convert the events generated by the timer sources and pin events and trigger ADC conversion requests. The main enhancement is the increased number of trigger sources and the inclusion of a list to allow multiple ADC conversions from a single trigger.

The BCTU connects to the injection trigger inputs of the SAR ADCs. The input trigger sources to the BCTU are from the ETPU, EMIOS, PIT, and external pins and are defined in the table below. The trigger sources are connected directly to the BCTU, and there is no second level of multiplexing.

All SAR ADCs on the device connect to the BCTU, and the functionality for each SAR is orthogonal.

Table 46-1. BCTU Input Trigger Mapping

BCTU Input Trigger	Trigger Source
0	ETPU15_A
1	ETPU14_A
2	ETPU13_A
3	ETPU12_A
4	ETPU11_A
5	ETPU10_A
6	ETPU9_A
7	ETPU8_A

Table continues on the next page...

Table 46-1. BCTU Input Trigger Mapping (continued)

BCTU Input Trigger	Trigger Source
8	ETPU15_B
9	ETPU14_B
10	ETPU13_B
11	ETPU12_B
12	ETPU11_B
13	ETPU10_B
14	ETPU9_B
15	ETPU8_B
16	EMIOS0_15
17	EMIOS0_14
18	EMIOS0_13
19	EMIOS0_12
20	EMIOS1_15
21	EMIOS1_14
22	EMIOS1_13
23	EMIOS1_12
24	PIT0_0
25	PIT0_1
26	PIT0_2
27	PIT0_3
28	PD[1]
29	PD[10][1]
30	PJ[9][2]
31	PJ[12]

1. Package pins mentioned in above table have a safe state 1. So triggers get latched as soon ipg_clk starts coming to BCTU. These triggers will need to be cleared and pads will have to driven zero before configuring BCTU."
2. These pads have a high state 1. So triggers get latched as soon ipg_clk starts coming to BCTU. These triggers will need to be cleared and pads will have to driven zero before configuring BCTU.

The following figure shows how the BCTU is connected to other modules.

Figure 46-1. BCTU block diagram

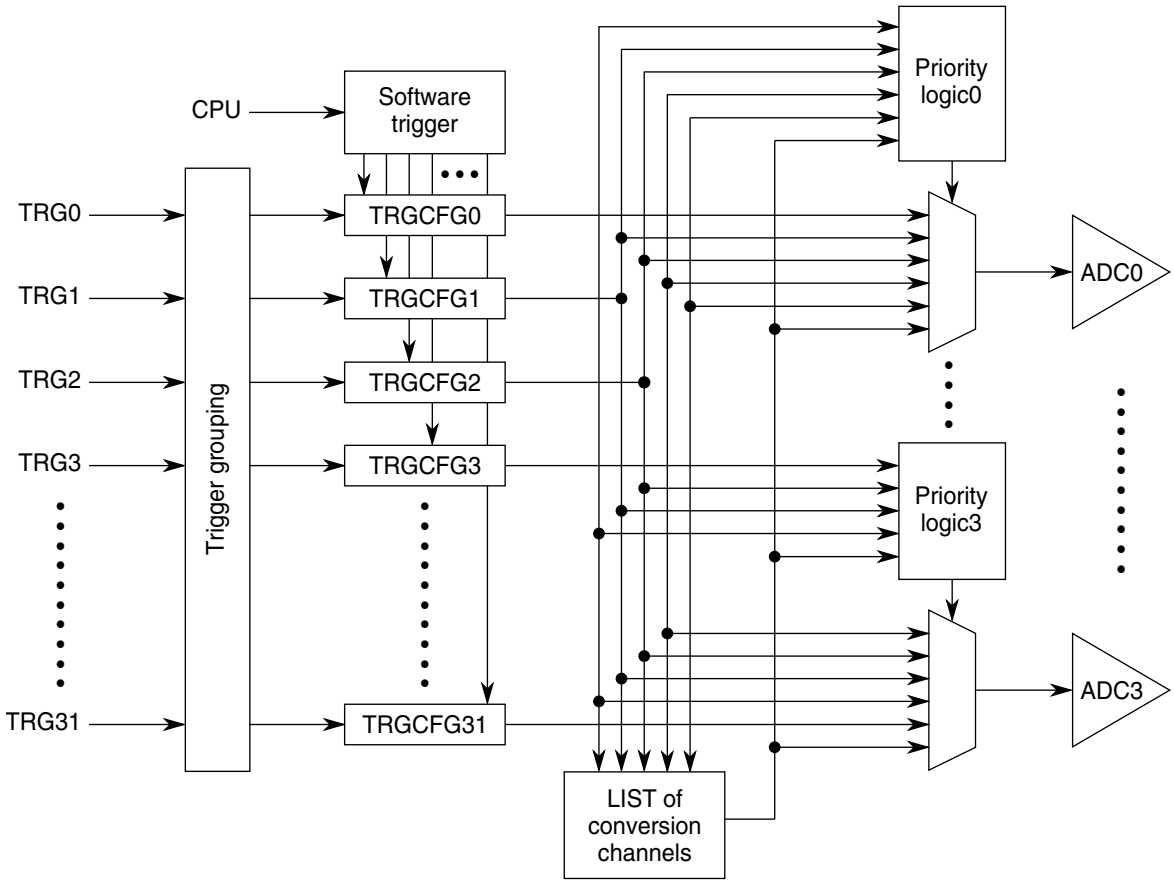
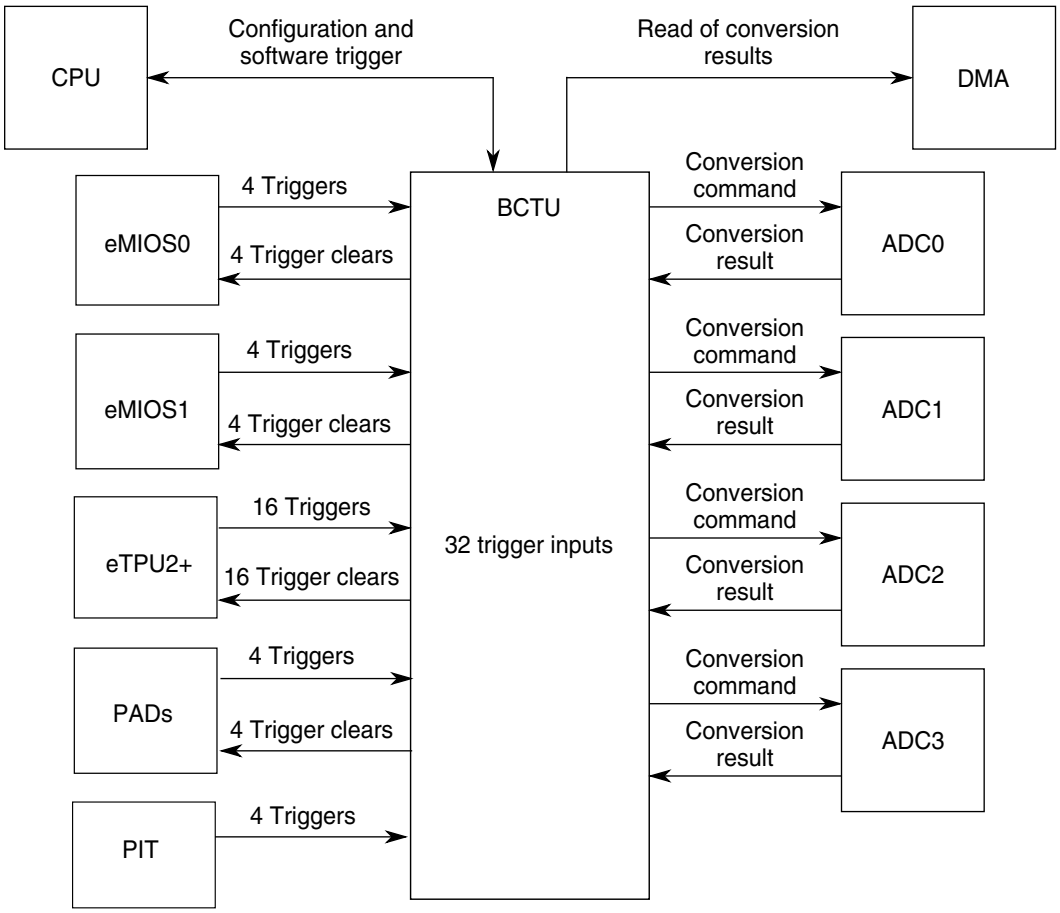


Figure 46-2. BCTU connections to other on chip modules



The following figures show the priority stages architecture using 32 trigger inputs.

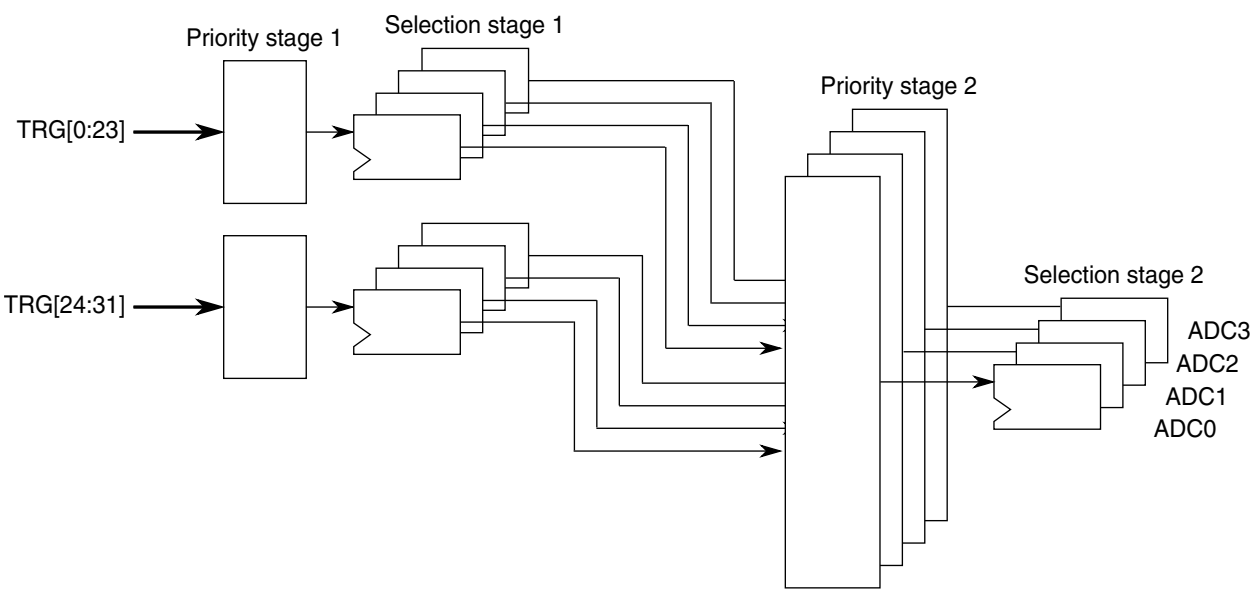


Figure 46-3. Priority stages architecture

46.1.2 Functional Description Summary

The enhanced BCTU allows synchronization between an ADC conversion and a timer event. Channels from the eMIOS modules, eTPU and PIT channels and package pins can be used to trigger an ADC conversion.

While the eTPU is triggered by channel output, BCTU is triggered by the eMIOS flag (eMIOS_Sn[FLAG]) rather than the channel output. The eMIOS flag can be cleared by a BCTU clear signal only if eMIOS_Cn[FEN] is not set.

46.2 Introduction

The Body Cross-triggering Unit (BCTU) is a module that is used to trigger the on chip ADCs. The trigger sources are generated by on chip timer modules. Usually in Body domain of applications the timers used are the enhanced Modular IO Sub-system (eMIOS), enhanced Time Processing Unit (ETPU), and the Programmable Interrupt Timer (PIT). In addition to triggering ADC single conversions, the BCTU also implements a LIST—a set of registers programmed by software that stores ADC channel numbers—that triggers several conversions based on a single trigger occurrence. The BCTU has 32 trigger inputs. The CPU can also be a source of conversion triggers, thus implementing a software trigger control. Up to 4 ADC modules are supported. The conversion results are stored in a register which is available for CPU read access. Figure below shows the BCTU top level diagram.

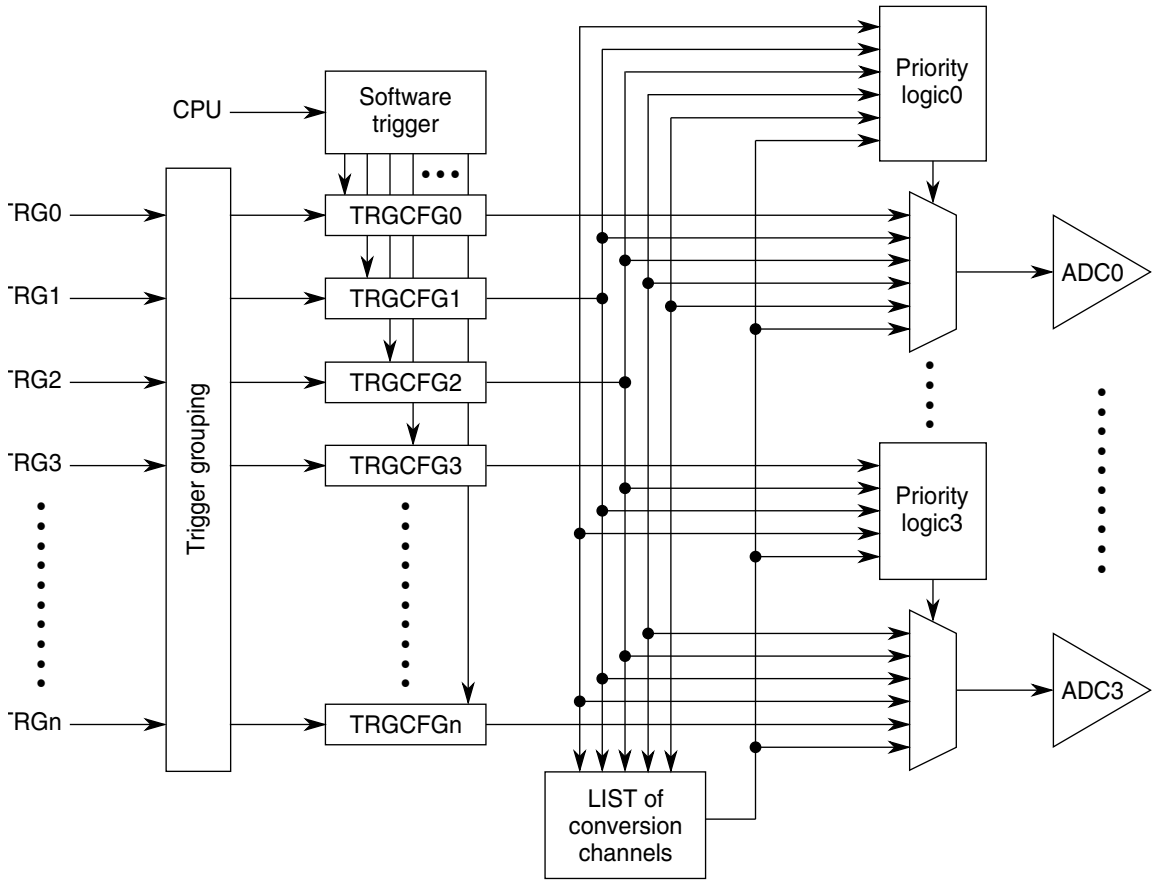


Figure 46-4. BCTU block diagram

46.3 BCTU overview

The BCTU receives several triggers from on chip timers and a software trigger controlled by the host CPU. These signals are used in order to generate the ADC trigger signals for up to 4 ADC modules. One trigger input selects one trigger configuration register. The trigger configuration register has the information about the target ADC and channel for the conversion. Alternatively a list of ADC channels can be associated with a trigger configuration register. In this case the target ADC converts a sequence of channels which are stored in the LIST.

The BCTU inputs are up to 32 trigger signals, which are level-sensitive; thus, if a logical level 1 is detected, an event is captured by the BCTU.

The figure above shows a high level BCTU block diagram. The BCTU hardware architecture consists of:

- Up to 32 conversion configuration registers TRGCFG
- One priority selection logic per ADC

- Up to 4 selection multiplexers to select one trigger at a time for each ADC
- A LIST of ADC channels

The priority logic is responsible for selecting among all active trigger inputs the one which is used to trigger an ADC conversion. The basic operation of the priority logic is to select the smallest trigger number as the higher priority. Thus if TRG[0] and TRG[1] are active TRG[0] is selected first and then TRG[1]. This priority scheme is used only if TRG[0] and TRG[1] target the same ADC. If a different ADC is selected by each trigger then there is no conflict and both triggers are selected at the same time. In addition to the trigger priority scheme the BCTU provides a LIST of channels that are used in a sequence of conversions. The LIST is a set of registers programmed by the software that stores ADC Channel numbers. The LIST also stores information of the last conversion to be executed in a sequence of conversions. A regular single conversion or a LIST of conversions are configured in the Trigger Configuration register. Since the ADC is selected by the ADC_SEL bit in the TRGCFG register for one single or a list of conversions, the LIST does not store the target ADC selection, thus all channels in the LIST refers to the ADCs specified in the TRGCFG register.

NOTE

Differently from the single command, the LIST can execute conversions on more than one ADC. Up to four ADCs can be addressed in a LIST.

46.4 BCTU Interacting with other peripherals

Figure below shows an example of how the BCTU interacts with on chip peripherals.

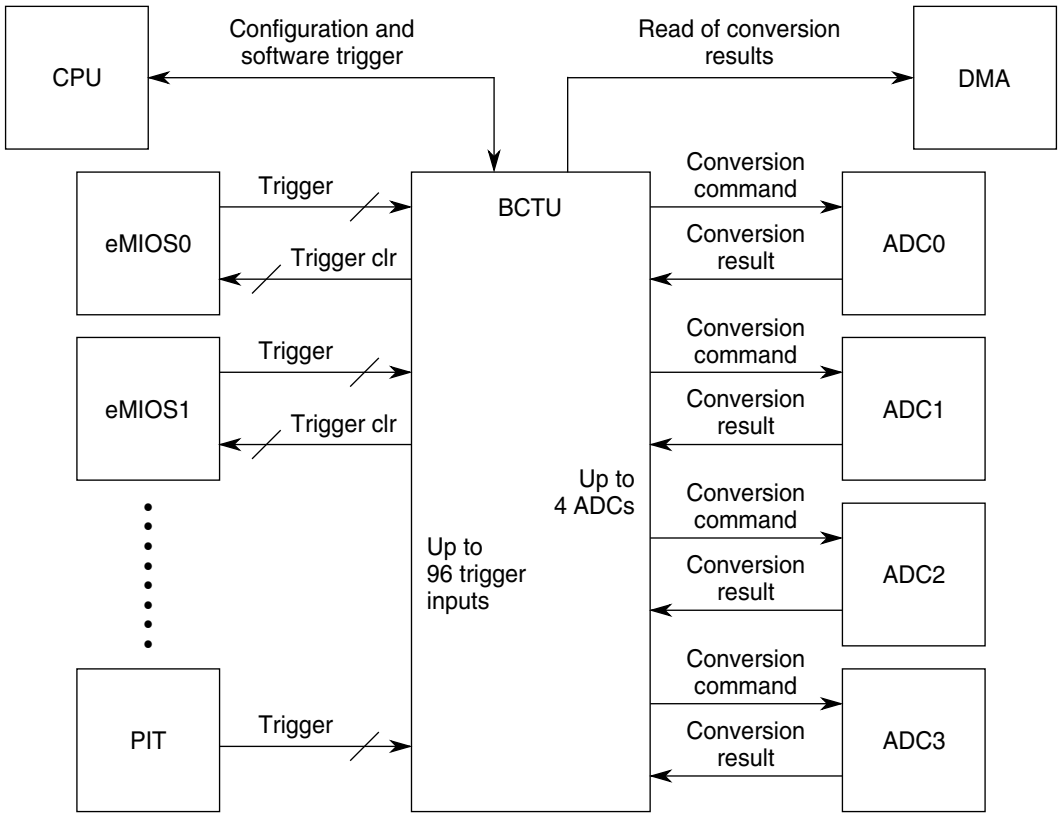


Figure 46-5. BCTU connections to other on chip modules

The timers (i.e. eMIOS, eTPU and PIT) generate triggers which are asserted at a specific time when the ADC conversion should be done. The transition, from zero to one, of these signals is captured by the BCTU logic and stored until the corresponding ADC conversion starts. At this time the BCTU asserts a trigger clear signal which clears both this internal captured trigger signal and the external trigger source if they are still asserted.

NOTE

Since the timer trigger may remain active until the corresponding conversion has initiated, there could be a case where another match occurs in the timer (i.e. eMIOS or PIT). The detection of that condition is not done by the BCTU logic. It is up to the application software to evaluate such a trigger overrun.

46.5 BCTU Modes of operation

The BCTU has following modes of operation, which are described below.

- Configuration mode

- Normal mode
- Low power mode

46.5.1 Configuration Mode

In the configuration mode the BCTU does not issue triggers to the ADCs connected to it. This mode allows for the software to program the TRGCFG registers, freely, without generating any spurious trigger to the ADCs. All trigger inputs are disabled. The exit from the configuration mode is made by setting the global enable bit GTRGEN in the MCR register. This procedure allows to enable all trigger inputs, which are already individually enabled by the TRIGEN bit in the TRGCFG register, with a single software access.

NOTE

In order to reinitialize the trigger input interface registers, the application software should clear all triggers by writing 1 to the TRG_FLAG bit in all TRGCFG registers before asserting the GTRGEN bit . This procedure clears the BCTU trigger input registers as well as the trigger sources that are connected to BCTU trigger clear signals (i.e. eMIOS).

46.5.2 Normal Mode

In the normal mode of operation the BCTU generates triggers for the ADCs based on trigger inputs from the on-chip timer modules. Due to backward compatibility reasons this is the mode selected coming out of reset.

46.5.3 Low Power Mode

The low power mode is entered when the MDIS bit in the MCR register is set. In this mode the BCTU is disabled, meaning that all trigger inputs are disregarded.

No triggers are issued to the ADCs connected to it. If MDIS is set any on going conversion is finished before the low power mode is entered. The TRG and NDATAx flags are not set when the module enters in low power mode.

If there are pending active triggers, they remain active (no trigger clear signal is issued). In this case it is up for the application software to clear the pending triggers, if required.

In the low power mode, the clocks for the BCTU module may be stopped, thus allowing to reduce the dynamic power consumption of the module. The exit from low power mode is made by clearing the MDIS bit in the MCR register. The module becomes immediately active after the MDIS bit is cleared, meaning that if the GTRGEN bit is set any active trigger input is processed and the corresponding ADC is triggered in order to execute a new conversion. In low power mode the ADCDRx registers are available for software read operation.

46.6 Memory Map

BCTU registers are implemented in 32-bit. 8-bit write operations are allowed. The coherency in this case must be provided by the application software since there is no coherence mechanism built into the BCTU hardware.

NOTE

RESERVED bit registers, when available for write operations, should be written always with logic ‘0’. Read access does not return meaningful data from RESERVED bit registers.

NOTE

Registers with write protection, if protection mechanism is active and an attempt for writing to those registers is done, will not cause any change to the register content and a transfer error will be issued.

BCTU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (BCTU_MCR)	32	R/W	0000_0000h	46.6.1/1789
8	Module Status register (BCTU_MSR)	32	R/W	0000_0000h	46.6.2/1792
18	Trigger Configuration Register (BCTU_TRGCFG_0)	32	R/W	0000_0000h	46.6.3/1796
1C	Trigger Configuration Register (BCTU_TRGCFG_1)	32	R/W	0000_0000h	46.6.3/1796
20	Trigger Configuration Register (BCTU_TRGCFG_2)	32	R/W	0000_0000h	46.6.3/1796
24	Trigger Configuration Register (BCTU_TRGCFG_3)	32	R/W	0000_0000h	46.6.3/1796
28	Trigger Configuration Register (BCTU_TRGCFG_4)	32	R/W	0000_0000h	46.6.3/1796
2C	Trigger Configuration Register (BCTU_TRGCFG_5)	32	R/W	0000_0000h	46.6.3/1796
30	Trigger Configuration Register (BCTU_TRGCFG_6)	32	R/W	0000_0000h	46.6.3/1796
34	Trigger Configuration Register (BCTU_TRGCFG_7)	32	R/W	0000_0000h	46.6.3/1796
38	Trigger Configuration Register (BCTU_TRGCFG_8)	32	R/W	0000_0000h	46.6.3/1796
3C	Trigger Configuration Register (BCTU_TRGCFG_9)	32	R/W	0000_0000h	46.6.3/1796
40	Trigger Configuration Register (BCTU_TRGCFG_10)	32	R/W	0000_0000h	46.6.3/1796

Table continues on the next page...

BCTU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
44	Trigger Configuration Register (BCTU_TRGCFG_11)	32	R/W	0000_0000h	46.6.3/1796
48	Trigger Configuration Register (BCTU_TRGCFG_12)	32	R/W	0000_0000h	46.6.3/1796
4C	Trigger Configuration Register (BCTU_TRGCFG_13)	32	R/W	0000_0000h	46.6.3/1796
50	Trigger Configuration Register (BCTU_TRGCFG_14)	32	R/W	0000_0000h	46.6.3/1796
54	Trigger Configuration Register (BCTU_TRGCFG_15)	32	R/W	0000_0000h	46.6.3/1796
58	Trigger Configuration Register (BCTU_TRGCFG_16)	32	R/W	0000_0000h	46.6.3/1796
5C	Trigger Configuration Register (BCTU_TRGCFG_17)	32	R/W	0000_0000h	46.6.3/1796
60	Trigger Configuration Register (BCTU_TRGCFG_18)	32	R/W	0000_0000h	46.6.3/1796
64	Trigger Configuration Register (BCTU_TRGCFG_19)	32	R/W	0000_0000h	46.6.3/1796
68	Trigger Configuration Register (BCTU_TRGCFG_20)	32	R/W	0000_0000h	46.6.3/1796
6C	Trigger Configuration Register (BCTU_TRGCFG_21)	32	R/W	0000_0000h	46.6.3/1796
70	Trigger Configuration Register (BCTU_TRGCFG_22)	32	R/W	0000_0000h	46.6.3/1796
74	Trigger Configuration Register (BCTU_TRGCFG_23)	32	R/W	0000_0000h	46.6.3/1796
78	Trigger Configuration Register (BCTU_TRGCFG_24)	32	R/W	0000_0000h	46.6.3/1796
7C	Trigger Configuration Register (BCTU_TRGCFG_25)	32	R/W	0000_0000h	46.6.3/1796
80	Trigger Configuration Register (BCTU_TRGCFG_26)	32	R/W	0000_0000h	46.6.3/1796
84	Trigger Configuration Register (BCTU_TRGCFG_27)	32	R/W	0000_0000h	46.6.3/1796
88	Trigger Configuration Register (BCTU_TRGCFG_28)	32	R/W	0000_0000h	46.6.3/1796
8C	Trigger Configuration Register (BCTU_TRGCFG_29)	32	R/W	0000_0000h	46.6.3/1796
90	Trigger Configuration Register (BCTU_TRGCFG_30)	32	R/W	0000_0000h	46.6.3/1796
94	Trigger Configuration Register (BCTU_TRGCFG_31)	32	R/W	0000_0000h	46.6.3/1796
228	Write Protection register (BCTU_WRPROT)	32	R/W	0000_0000h	46.6.4/1798
22C	Software Trigger register 1 (BCTU_SFTRGR1)	32	R/W	0000_0000h	46.6.5/1799
234	Software Trigger register 1 (BCTU_SFTRGR3)	32	R/W	0000_0000h	46.6.6/1802
23C	ADC0 Result Data register (BCTU_ADC0DR)	32	R	0000_0000h	46.6.7/1806
240	ADC1 result data register (BCTU_ADC1DR)	32	R	0000_0000h	46.6.8/1807
244	ADC2 result data register (BCTU_ADC2DR)	32	R	0000_0000h	46.6.9/1808
248	ADC3 result data register (BCTU_ADC3DR)	32	R	0000_0000h	46.6.10/ 1809
24C	LIST Size Status register (BCTU_LISTSTAR)	32	R/W	0000_1100h	46.6.11/ 1809
250	LIST Channel Address Register (BCTU_LISTCHR_0)	32	R/W	See section	46.6.12/ 1810
254	LIST Channel Address Register (BCTU_LISTCHR_1)	32	R/W	See section	46.6.12/ 1810
258	LIST Channel Address Register (BCTU_LISTCHR_2)	32	R/W	See section	46.6.12/ 1810
25C	LIST Channel Address Register (BCTU_LISTCHR_3)	32	R/W	See section	46.6.12/ 1810
260	LIST Channel Address Register (BCTU_LISTCHR_4)	32	R/W	See section	46.6.12/ 1810

Table continues on the next page...

BCTU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
264	LIST Channel Address Register (BCTU_LISTCHR_5)	32	R/W	See section	46.6.12/ 1810
268	LIST Channel Address Register (BCTU_LISTCHR_6)	32	R/W	See section	46.6.12/ 1810
26C	LIST Channel Address Register (BCTU_LISTCHR_7)	32	R/W	See section	46.6.12/ 1810
270	LIST Channel Address Register (BCTU_LISTCHR_8)	32	R/W	See section	46.6.12/ 1810
274	LIST Channel Address Register (BCTU_LISTCHR_9)	32	R/W	See section	46.6.12/ 1810
278	LIST Channel Address Register (BCTU_LISTCHR_10)	32	R/W	See section	46.6.12/ 1810
27C	LIST Channel Address Register (BCTU_LISTCHR_11)	32	R/W	See section	46.6.12/ 1810
280	LIST Channel Address Register (BCTU_LISTCHR_12)	32	R/W	See section	46.6.12/ 1810
284	LIST Channel Address Register (BCTU_LISTCHR_13)	32	R/W	See section	46.6.12/ 1810
288	LIST Channel Address Register (BCTU_LISTCHR_14)	32	R/W	See section	46.6.12/ 1810
28C	LIST Channel Address Register (BCTU_LISTCHR_15)	32	R/W	See section	46.6.12/ 1810
290	LIST Channel Address Register (BCTU_LISTCHR_16)	32	R/W	See section	46.6.12/ 1810
294	LIST Channel Address Register (BCTU_LISTCHR_17)	32	R/W	See section	46.6.12/ 1810
298	LIST Channel Address Register (BCTU_LISTCHR_18)	32	R/W	See section	46.6.12/ 1810
29C	LIST Channel Address Register (BCTU_LISTCHR_19)	32	R/W	See section	46.6.12/ 1810
2A0	LIST Channel Address Register (BCTU_LISTCHR_20)	32	R/W	See section	46.6.12/ 1810
2A4	LIST Channel Address Register (BCTU_LISTCHR_21)	32	R/W	See section	46.6.12/ 1810
2A8	LIST Channel Address Register (BCTU_LISTCHR_22)	32	R/W	See section	46.6.12/ 1810
2AC	LIST Channel Address Register (BCTU_LISTCHR_23)	32	R/W	See section	46.6.12/ 1810
2B0	LIST Channel Address Register (BCTU_LISTCHR_24)	32	R/W	See section	46.6.12/ 1810
2B4	LIST Channel Address Register (BCTU_LISTCHR_25)	32	R/W	See section	46.6.12/ 1810
2B8	LIST Channel Address Register (BCTU_LISTCHR_26)	32	R/W	See section	46.6.12/ 1810

Table continues on the next page...

BCTU memory map (continued)

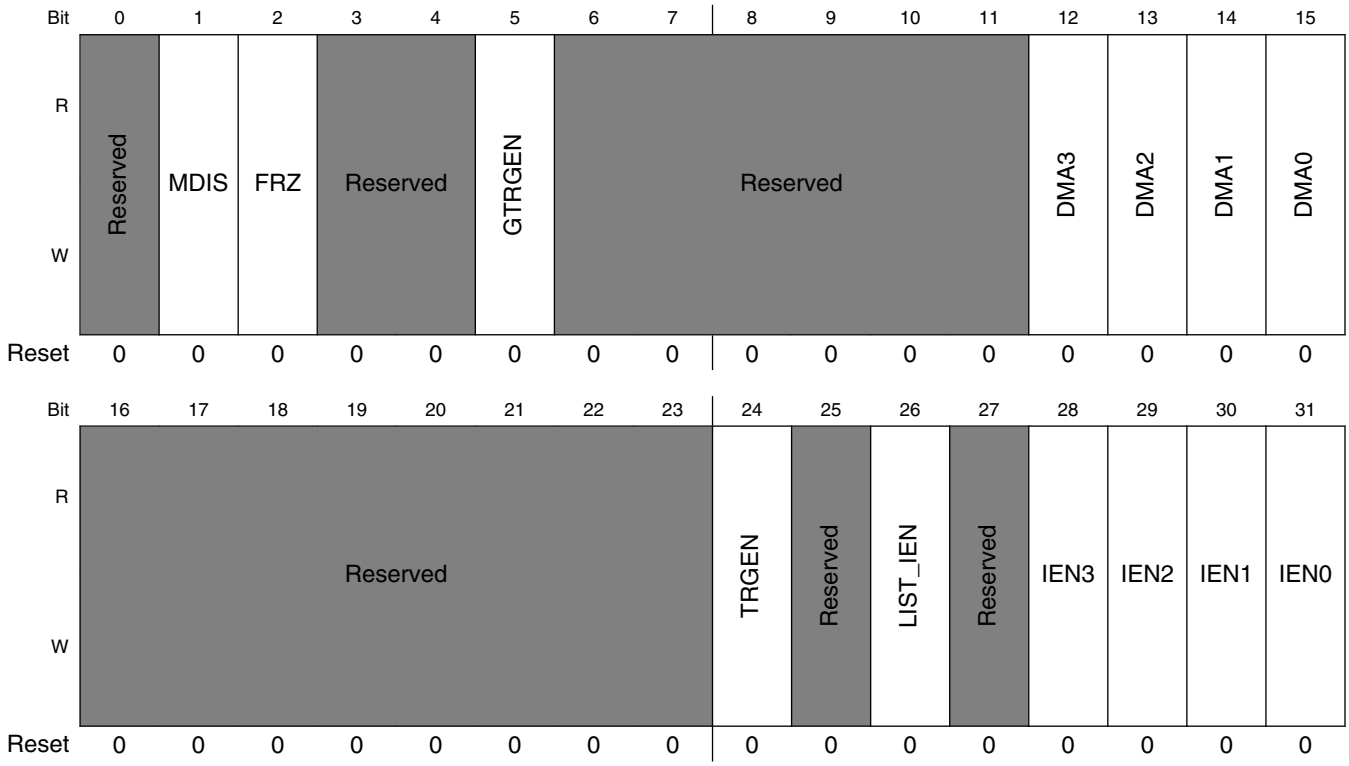
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2BC	LIST Channel Address Register (BCTU_LISTCHR_27)	32	R/W	See section	46.6.12/ 1810
2C0	LIST Channel Address Register (BCTU_LISTCHR_28)	32	R/W	See section	46.6.12/ 1810
2C4	LIST Channel Address Register (BCTU_LISTCHR_29)	32	R/W	See section	46.6.12/ 1810
2C8	LIST Channel Address Register (BCTU_LISTCHR_30)	32	R/W	See section	46.6.12/ 1810
2CC	LIST Channel Address Register (BCTU_LISTCHR_31)	32	R/W	See section	46.6.12/ 1810
2D0	Conversion configuration register (BCTU_CCR_0)	32	R/W	See section	46.6.13/ 1811
2D4	Conversion configuration register (BCTU_CCR_1)	32	R/W	See section	46.6.13/ 1811
2D8	Conversion configuration register (BCTU_CCR_2)	32	R/W	See section	46.6.13/ 1811
2DC	Conversion configuration register (BCTU_CCR_3)	32	R/W	See section	46.6.13/ 1811
2E0	Conversion configuration register (BCTU_CCR_4)	32	R/W	See section	46.6.13/ 1811
2E4	Conversion configuration register (BCTU_CCR_5)	32	R/W	See section	46.6.13/ 1811
2E8	Conversion configuration register (BCTU_CCR_6)	32	R/W	See section	46.6.13/ 1811
2EC	Conversion configuration register (BCTU_CCR_7)	32	R/W	See section	46.6.13/ 1811
2F0	Conversion configuration register (BCTU_CCR_8)	32	R/W	See section	46.6.13/ 1811
2F4	Conversion configuration register (BCTU_CCR_9)	32	R/W	See section	46.6.13/ 1811
2F8	Conversion configuration register (BCTU_CCR_10)	32	R/W	See section	46.6.13/ 1811

46.6.1 Module Configuration Register (BCTU_MCR)

The Module Configuration register (MCR) provides access to low power, debug, trigger control, DMA control and Interrupt control bits.

Access: User read/write

Address: 0h base + 0h offset = 0h



BCTU_MCR field descriptions

Field	Description
0 Reserved	This field is reserved.
1 MDIS	Module Disable 0 Module is in normal mode. 1 Module is in low power mode.
2 FRZ	Freeze mode - for debug operation. The freeze mode disables all trigger inputs from timers but leave the software trigger enabled, thus allowing the software to trigger a conversion and read the conversion result.
3–4 Reserved	This field is reserved.
5 GTRGEN	Global Trigger Enable. Enables all trigger inputs at the same time. The individual trigger enable must be set in order for that trigger to become active.
6–11 Reserved	This field is reserved.
12 DMA3	ADC3DR data transferring by DMA. Enables DMA operation on transferring ADC3DR data when there is new data available in this register. 0 Disable DMA operation. 1 Enable DMA operation.

Table continues on the next page...

BCTU_MCR field descriptions (continued)

Field	Description
13 DMA2	ADC2DR data transferring by DMA. Enables DMA operation on transferring ADC2DR data when there is new data available in this register. 0 Disable DMA operation. 1 Enable DMA operation.
14 DMA1	ADC1DR data transferring by DMA. Enables DMA operation on transferring ADC1DR data when there is new data available in this register. 0 Disable DMA operation. 1 Enable DMA operation.
15 DMA0	ADC0DR data transferring by DMA. Enables DMA operation on transferring ADC0DR data when there is new data available in this register. 0 Disable DMA operation. 1 Enable DMA operation.
16–23 Reserved	This field is reserved.
24 TRGEN	Trigger interrupt request enable. Enables TRGF flag set in MSR register to generate interrupt request 0 Trigger interrupt request disabled. 1 Trigger interrupt request enabled. A request is generated if the TRGF flag bit in MSR register is set.
25 Reserved	This field is reserved.
26 LIST_IEN	LIST interrupt enable. If a LIST last conversion is executed an interrupt request is issued to the CPU if LIST_IEN is set. For the interrupt to occur the LIST and LAST flags must be set. 0 No interrupt request. 1 Interrupt request.
27 Reserved	This field is reserved.
28 IEN3	Interrupt enable bit for ADC3DR new data. This bit enables an interrupt request if the ADC3DR register is written by the hardware, meaning that there is new conversion data available. It also enables an interrupt request to be issued if there is an overrun in the ADC3DR register, meaning that the previous data in this register was overwritten by a new data. See also NDATA3 and DATAOVR3 in the MSR register. If DMA3 bit is set the interrupt for ADC3DR new data is disabled but the interrupt for ADC3DR data overrun remains enabled. In this case if an overrun condition occurs an interrupt request is generated. 0 Disable interrupt request. 1 Enable interrupt request.
29 IEN2	Interrupt enable bit for ADC2DR new data. This bit enables an interrupt request if the ADC2DR register is written by the hardware, meaning that there is new conversion data available. It also enables an interrupt request to be issued if there is an overrun in

Table continues on the next page...

BCTU_MCR field descriptions (continued)

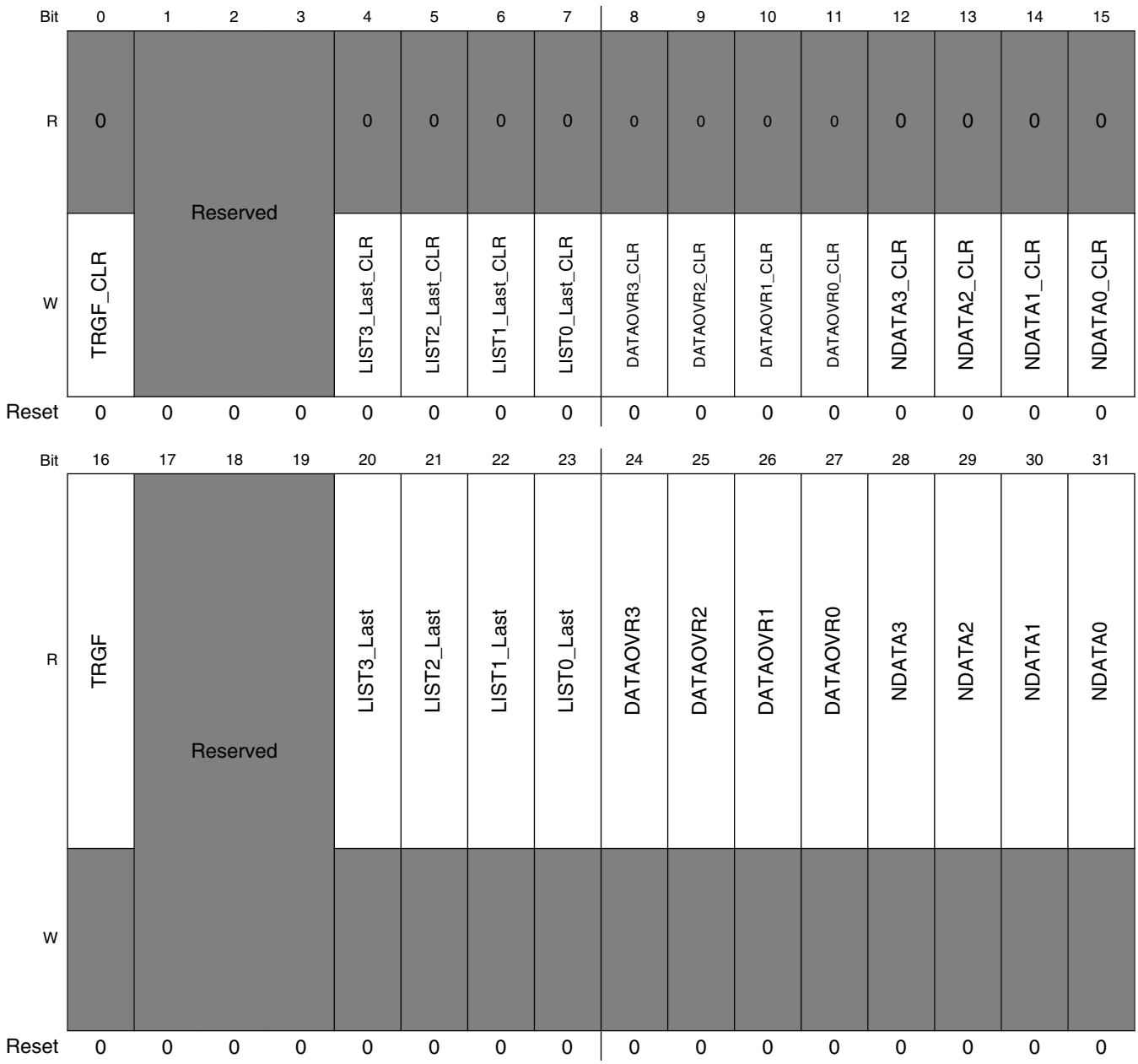
Field	Description
	the ADC2DR register, meaning that the previous data in this register was overwritten by a new data. See also NDATA2 and DATAOVR2 in the MSR register. If DMA2 bit is set the interrupt for ADC2DR new data is disabled but the interrupt for ADC2DR data overrun remains enabled. In this case if an overrun condition occurs an interrupt request is generated. 0 Disable interrupt request. 1 Enable interrupt request.
30 IEN1	Interrupt enable bit for ADC1DR new data. This bit enables an interrupt request if the ADC1DR register is written by the hardware, meaning that there is new conversion data available. It also enables an interrupt request to be issued if there is an overrun in the ADC1DR register, meaning that the previous data in this register was overwritten by a new data. See also NDATA1 and DATAOVR1 in the MSR register. If DMA1 bit is set the interrupt for ADC1DR new data is disabled but the interrupt for ADC1DR data overrun remains enabled. In this case if an overrun condition occurs an interrupt request is generated. 0 Disable interrupt request. 1 Enable interrupt request.
31 IEN0	Interrupt enable bit for ADC0DR new data. This bit enables an interrupt request if the ADC0DR register is written by the hardware, meaning that there is new conversion data available. It also enables an interrupt request to be issued if there is an overrun in the ADC0DR register, meaning that the previous data in this register was overwritten by a new data. See also NDATA0 and DATAOVR0 in the MSR register. If DMA0 bit is set the interrupt for ADC0DR new data is disabled but the interrupt for ADC0DR data overrun remains enabled. In this case if an overrun condition occurs an interrupt request is generated. 0 Disable interrupt request. 1 Enable interrupt request.

46.6.2 Module Status register (BCTU_MSR)

The Status register provides software access to flags that indicate if conversion data is available for all ADCs. There are also indication of LIST status bits, such as last conversion command executed in a LIST.

Access: User read/write

Address: 0h base + 8h offset = 8h



BCTU_MSR field descriptions

Field	Description
0 TRGF_CLR	TRGF clear This bit is the clear for the TRGF flag bit 0 No action 1 Clears TRGF flag
1–3 Reserved	This field is reserved.

Table continues on the next page...

BCTU_MSR field descriptions (continued)

Field	Description
4 LIST3_Last_CLR	Clears LIST_Last flag for ADC3 0 No action 1 Clears LIST3_Last bit
5 LIST2_Last_CLR	Clears LIST_Last flag for ADC2 0 No action 1 Clears LIST2_Last bit
6 LIST1_Last_CLR	Clears LIST_Last flag for ADC1 0 no action 1 clears LIST1_Last flag
7 LIST0_Last_CLR	Clears LIST_Last flag for ADC0 0 no action 1 Clears LIST0_Last flag
8 DATAOVR3_ CLR	Clears DATAOVR3 flag 0 no action 1 Clears NDATA3 flag
9 DATAOVR2_ CLR	Clears DATAOVR2 flag 0 no action 1 Clears DATAOVR2 flag
10 DATAOVR1_ CLR	Clears DATAOVR1 flag 0 no action 1 Clears DATAOVR1 flag
11 DATAOVR0_ CLR	Clears DATAOVR0 flag 0 no action 1 Clears DATAOVR0 flag
12 NDATA3_CLR	Clears NDATA3 flag 0 No action 1 Clears NDATA3 flag
13 NDATA2_CLR	Clears NDATA2 flag 0 no action 1 Clears NDATA2 flag
14 NDATA1_CLR	Clears NDATA1 flag 0 no action 1 Clears NDATA1 flag
15 NDATA0_CLR	Clears NDATA0 flag 0 no action 1 Clears NDATA0 flag

Table continues on the next page...

BCTU_MSR field descriptions (continued)

Field	Description
16 TRGF	Trigger Flag Indicates that at least one ADC was triggered. 0 ADC was not triggered 1 An ADC was triggered
17–19 Reserved	This field is reserved.
20 LIST3_Last	Execution of the last conversion in a LIST of conversions for ADC3 0 not the last conversion 1 last conversion in a LIST
21 LIST2_Last	Execution of the last conversion in a LIST of conversions for ADC2 0 not the last conversion 1 last conversion in a LIST
22 LIST1_Last	Execution of the last conversion in a LIST of conversions for ADC1 0 not the last conversion 1 last conversion in a LIST
23 LIST0_Last	Execution of the last conversion in a LIST of conversions for ADC0 0 not the last conversion 1 last conversion in a LIST
24 DATAOVR3	Data overrun for ADC3. Old data in the ADCDR3 register was overwritten 0 data not overwritten 1 data overwritten
25 DATAOVR2	Data overrun for ADC2. Old data in the ADCDR2 register was overwritten 0 data not overwritten 1 data overwritten
26 DATAOVR1	Data overrun for ADC1. Old data in the ADCDR1 register was overwritten 0 data not overwritten 1 data overwritten
27 DATAOVR0	Data overrun for ADC0. Old data in the ADCDR0 register was overwritten 0 data not overwritten 1 data overwritten
28 NDATA3	New conversion data is available for ADC3 0 no new data available 1 new data available
29 NDATA2	New conversion data is available for ADC2 0 no new data available 1 new data available

Table continues on the next page...

BCTU_MSR field descriptions (continued)

Field	Description
30 NDATA1	New conversion data is available for ADC1 0 no new data available 1 new data available
31 NDATA0	New conversion data is available for ADC0 0 no new data available 1 new data available

46.6.3 Trigger Configuration Register (BCTU_TRGCFG_*n*)

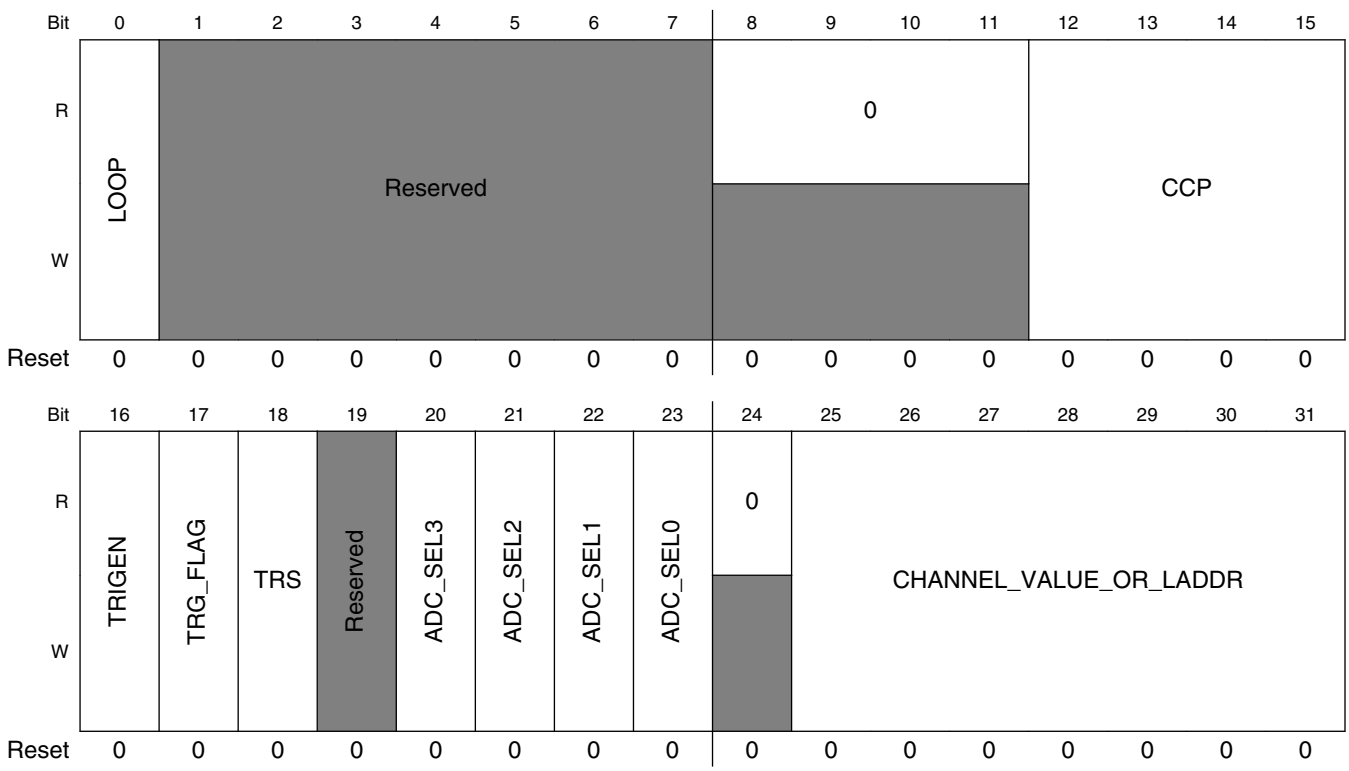
The Trigger Configuration register is a set of registers, one register per input trigger. This register controls the target ADC and ADC channel to be used in a conversion. It also controls if a LIST is used instead of a single ADC conversion command.

NOTE

x varies from 0 up to the number of input triggers.

Access: User read/write

Address: 0h base + 18h offset + (4d × i), where i=0d to 31d



BCTU_TRGCFG_*n* field descriptions

Field	Description
0 LOOP	LOOP bit The LOOP bit , defined if the current trigger should execute a loop, meaning that the conversion triggered by the input trigger is executed again until the LOOP bit is cleared. Each time the new conversion is triggered it should win the priority evaluation among all pending triggers. It is only executed if it is the highest priority triggered conversion at the moment the conversion starts.
1–7 Reserved	This field is reserved.
8–11 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
12–15 CCP	Conversion Configuration Pointer (CCP) The CCP field, if different from zero, points to a Conversion configuration register in the Conversion Configuration register bank.
16 TRIGEN	Trigger enable Enable/disable triggers from the input trigger (hardware) to initiate an ADC conversion. This bit does not disable software triggers. 0 disable hardware triggers 1 enable hardware triggers
17 TRG_FLAG	Clear timer flag through software 0 When written, indicates no action. When read, indicates that there is no ongoing ADC conversion initiated by this trigger. 1 When written, clears timer flag through software. Which means that any previous input trigger event is cleared, preventing a new ADC conversion to be initiated by this trigger source. When read, indicates that there is an ongoing ADC conversion initiated by this trigger.
18 TRS	Trigger resolution Selects single conversion or LIST of conversions 0 single conversion 1 List of conversions
19 Reserved	This field is reserved.
20 ADC_SEL3	Selects ADC 3 for conversion 0 does not selected ADC3 1 selects ADC3
21 ADC_SEL2	Selects ADC 2 for conversion 0 does not selected ADC2 1 selects ADC2
22 ADC_SEL1	Selects ADC 0 for conversion 0 does not selected ADC1 1 selects ADC1

Table continues on the next page...

BCTU_TRGCFG_*n* field descriptions (continued)

Field	Description
23 ADC_SEL0	Selects ADC 0 for conversion 0 does not selected ADC0 1 selects ADC0
24 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
25–31 CHANNEL_ VALUE_OR_ LADDR	Channel Value or List Address Value. If TRS = 0. this field is the ADC Channel selected for the conversion. If TRS = 1, this field represents the address of the first LIST position to be executed.

46.6.4 Write Protection register (BCTU_WRPROT)

Coming out from reset state the protection is enabled. In order to allow writes to protected registers it is necessary to write a special code in a 4-bit field. There are two possible codes that allow writes thus removing the protection. One 4-bit code that permanently disables the protection mechanism and another code that disables the protection during one write cycle to a protected register. After that the state of the protection code returns to its initial (protected) state.

Access: User read/write

Address: 0h base + 228h offset = 228h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved																												PROTEC_ CODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BCTU_WRPROT field descriptions

Field	Description
0–27 Reserved	This field is reserved.
28–31 PROTEC_CODE	The Protection code sets protection for the write protected registers There are two codes which disables the protection for the write protected registers. One code disables the write protection permanently and another code disables the protection for one write access only. **NOTE:** This field is writable only if the MCR[MDIS] bit is 0. 1010 This code defines that the protected registers are not in the protected mode, thus writes are enabled.

Table continues on the next page...

BCTU_WRPROT field descriptions (continued)

Field	Description
	1001 Protection temporarily deactivated. This code disables the protection for writes on the protected registers for one write cycle operation. After that the protection code is reset to the initial state "0000" thus restoring the protection for the protected registers. Protection is active for other values

46.6.5 Software Trigger register 1 (BCTU_SFTRGR1)

The Software Trigger register allows the software to trigger any of the conversions associated with the Trigger Configuration registers. In order to avoid ambiguity in the conversion results regarding the source of trigger (if from hardware or from the software) the trigger enable (TRIGEN) bit in the Trigger Configuration register should be cleared when the software trigger is used thus the trigger for that configuration register is only issued by the SFTRGR register. This register is write protected by the WRPROT register. If protection is enabled writes to this register do not modify the register content and generates a transfer error.

All bits in this register are set by the software and cleared by the hardware.

Access: User read/write; this register is write protected by the BCTU_WRPROT register.

Address: 0h base + 22Ch offset = 22Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SFTRG31	SFTRG30	SFTRG29	SFTRG28	SFTRG27	SFTRG26	SFTRG25	SFTRG24	SFTRG23	SFTRG22	SFTRG21	SFTRG20	SFTRG19	SFTRG18	SFTRG17	SFTRG16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SFTRG15	SFTRG14	SFTRG13	SFTRG12	SFTRG11	SFTRG10	SFTRG9	SFTRG8	SFTRG7	SFTRG6	SFTRG5	SFTRG4	SFTRG3	SFTRG2	SFTRG1	SFTRG0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BCTU_SFTRGR1 field descriptions

Field	Description
0 SFTRG31	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger

Table continues on the next page...

BCTU_SFTRGR1 field descriptions (continued)

Field	Description
1 SFTRG30	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
2 SFTRG29	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
3 SFTRG28	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
4 SFTRG27	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
5 SFTRG26	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
6 SFTRG25	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
7 SFTRG24	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
8 SFTRG23	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
9 SFTRG22	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
10 SFTRG21	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
11 SFTRG20	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
12 SFTRG19	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger

Table continues on the next page...

BCTU_SFTRGR1 field descriptions (continued)

Field	Description
13 SFTRG18	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
14 SFTRG17	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
15 SFTRG16	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
16 SFTRG15	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
17 SFTRG14	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
18 SFTRG13	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
19 SFTRG12	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
20 SFTRG11	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
21 SFTRG10	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
22 SFTRG9	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
23 SFTRG8	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
24 SFTRG7	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger

Table continues on the next page...

BCTU_SFTRGR1 field descriptions (continued)

Field	Description
25 SFTRG6	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
26 SFTRG5	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
27 SFTRG4	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
28 SFTRG3	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
29 SFTRG2	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
30 SFTRG1	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
31 SFTRG0	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger

46.6.6 Software Trigger register 1 (BCTU_SFTRGR3)

The Software Trigger register allows the software to trigger any of the conversions associated with the Trigger Configuration registers. In order to avoid ambiguity in the conversion results regarding the source of trigger (if from hardware or from the software) the trigger enable (TRIGEN) bit in the Trigger Configuration register should be cleared when the software trigger is used thus the trigger for that configuration register is only issued by the SFTRGR register. This register is write protected by the WRPROT register. If protection is enabled writes to this register do not modify the register content and generates a transfer error.

All bits in this register are set by the software and cleared by the hardware.

Access: User read/write; this register is write protected by the BCTU_WRPROT register.

Address: 0h base + 234h offset = 234h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SFTRG95	SFTRG94	SFTRG93	SFTRG92	SFTRG91	SFTRG90	SFTRG89	SFTRG88	SFTRG87	SFTRG86	SFTRG85	SFTRG84	SFTRG83	SFTRG82	SFTRG81	SFTRG80
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SFTRG79	SFTRG78	SFTRG77	SFTRG76	SFTRG75	SFTRG74	SFTRG73	SFTRG72	SFTRG71	SFTRG70	SFTRG69	SFTRG68	SFTRG67	SFTRG66	SFTRG65	SFTRG64
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BCTU_SFTRGR3 field descriptions

Field	Description
0 SFTRG95	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
1 SFTRG94	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
2 SFTRG93	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
3 SFTRG92	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
4 SFTRG91	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
5 SFTRG90	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
6 SFTRG89	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
7 SFTRG88	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger

Table continues on the next page...

BCTU_SFTRGR3 field descriptions (continued)

Field	Description
8 SFTRG87	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
9 SFTRG86	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
10 SFTRG85	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
11 SFTRG84	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
12 SFTRG83	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
13 SFTRG82	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
14 SFTRG81	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
15 SFTRG80	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
16 SFTRG79	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
17 SFTRG78	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
18 SFTRG77	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
19 SFTRG76	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger

Table continues on the next page...

BCTU_SFTRGR3 field descriptions (continued)

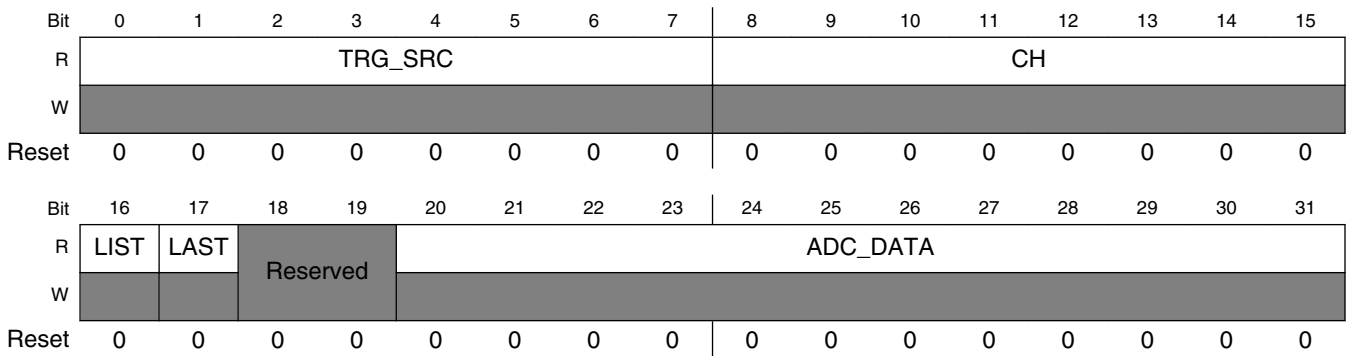
Field	Description
20 SFTRG75	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
21 SFTRG74	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
22 SFTRG73	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
23 SFTRG72	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
24 SFTRG71	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
25 SFTRG70	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
26 SFTRG69	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
27 SFTRG68	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
28 SFTRG67	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
29 SFTRG66	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
30 SFTRG65	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
31 SFTRG64	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger

46.6.7 ADC0 Result Data register (BCTU_ADC0DR)

The ADC Result Data register allows software access to ADC conversion result data. It also indicates the ADC channel used in that conversion, the trigger source that initiated that conversion and LIST status indication in case the conversion was initiated from a LIST.

Access: User read

Address: 0h base + 23Ch offset = 23Ch



BCTU_ADC0DR field descriptions

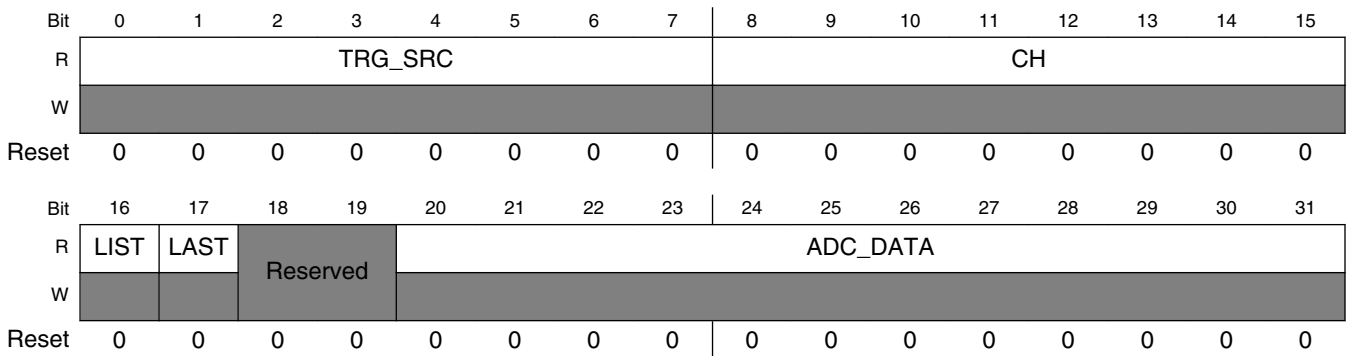
Field	Description
0–7 TRG_SRC	Trigger number used to triggered the conversion
8–15 CH	ADC Channel used for the conversion
16 LIST	Indicates if the conversion was initiated by a LIST element or single conversion trigger 0: single conversion 0 single conversion 1 LIST
17 LAST	In case the conversion was done through a LIST, if set to 1, this bit indicates that the conversion result is the last conversion in the LIST. If the conversion was not controlled by a LIST this bit is always zero. 0 The conversion result is not the last coversion in a LIST 1 The conversion result is the last conversion in a LIST
18–19 Reserved	This field is reserved.
20–31 ADC_DATA	DATA from ADC conversion

46.6.8 ADC1 result data register (BCTU_ADC1DR)

The ADC Result Data register allows software access to ADC conversion result data. It also indicates the ADC channel used in that conversion, the trigger source that initiated that conversion and LIST status indication in case the conversion was initiated from a LIST.

Access: User read

Address: 0h base + 240h offset = 240h



BCTU_ADC1DR field descriptions

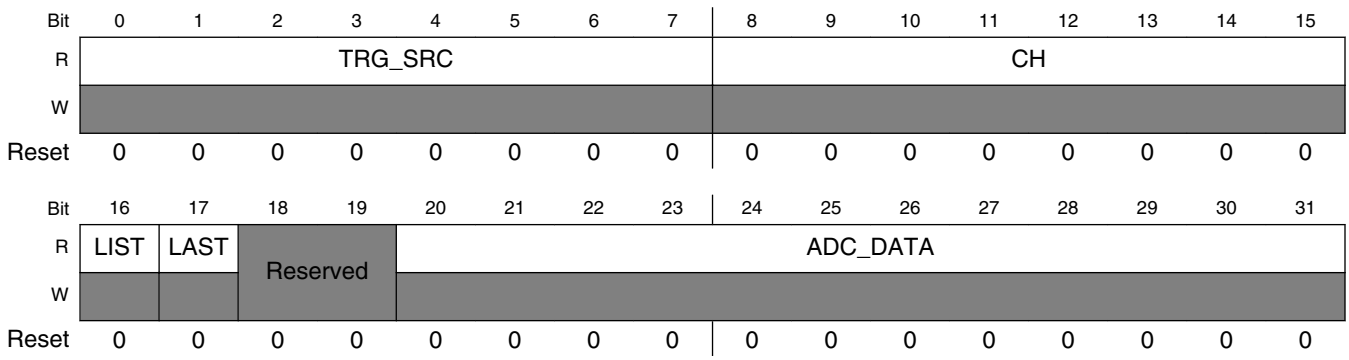
Field	Description
0–7 TRG_SRC	Trigger number used to triggered the conversion
8–15 CH	ADC Channel used for the conversion
16 LIST	Indicates if the conversion was initiated by a LIST element or single conversion trigger 0 single conversion 1 LIST
17 LAST	In case the conversion was done through a LIST, if set to 1, this bit indicates that the conversion result is the last conversion in the LIST. If the conversion was not controlled by a LIST this bit is always zero. 0 The conversion result is not the last coversion in a LIST 1 The conversion result is the last conversion in a LIST
18–19 Reserved	This field is reserved.
20–31 ADC_DATA	DATA from ADC conversion

46.6.9 ADC2 result data register (BCTU_ADC2DR)

The ADC Result Data register allows software access to ADC conversion result data. It also indicates the ADC channel used in that conversion, the trigger source that initiated that conversion and LIST status indication in case the conversion was initiated from a LIST.

Access: User read

Address: 0h base + 244h offset = 244h



BCTU_ADC2DR field descriptions

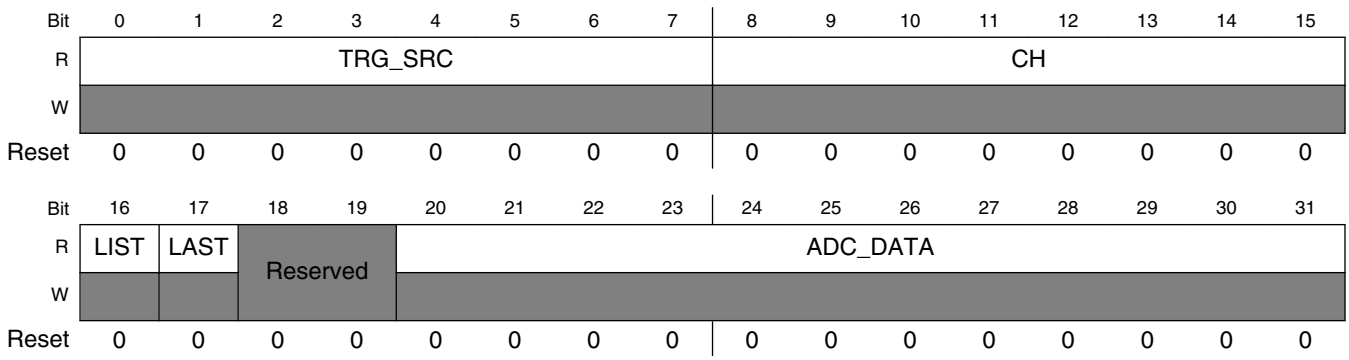
Field	Description
0–7 TRG_SRC	Trigger number used to triggered the conversion
8–15 CH	ADC Channel used for the conversion
16 LIST	Indicates if the conversion was initiated by a LIST element or single conversion trigger 0 single conversion 1 LIST
17 LAST	In case the conversion was done through a LIST, if set to 1, this bit indicates that the conversion result is the last conversion in the LIST. If the conversion was not controlled by a LIST this bit is always zero. 0 The conversion result is not the last coversion in a LIST 1 The conversion result is the last conversion in a LIST
18–19 Reserved	This field is reserved.
20–31 ADC_DATA	DATA from ADC conversion

46.6.10 ADC3 result data register (BCTU_ADC3DR)

The ADC Result Data register allows software access to ADC conversion result data. It also indicates the ADC channel used in that conversion, the trigger source that initiated that conversion and LIST status indication in case the conversion was initiated from a LIST.

Access: User read

Address: 0h base + 248h offset = 248h



BCTU_ADC3DR field descriptions

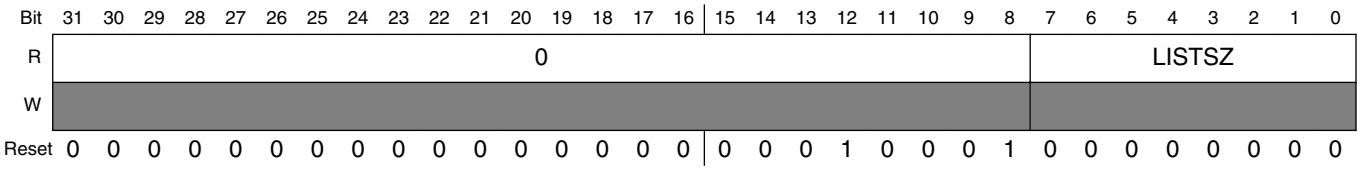
Field	Description
0–7 TRG_SRC	Trigger number used to triggered the conversion
8–15 CH	ADC Channel used for the conversion
16 LIST	Indicates if the conversion was initiated by a LIST element or single conversion trigger 0 single conversion 1 LIST
17 LAST	In case the conversion was done through a LIST, if set to 1, this bit indicates that the conversion result is the last conversion in the LIST. If the conversion was not controlled by a LIST this bit is always zero. 0 The conversion result is not the last coversion in a LIST 1 The conversion result is the last conversion in a LIST
18–19 Reserved	This field is reserved.
20–31 ADC_DATA	DATA from ADC conversion

46.6.11 LIST Size Status register (BCTU_LISTSTAR)

Each element has the channel number used in one conversion.

Access: User read

Address: 0h base + 24Ch offset = 24Ch



BCTU_LISTSTAR field descriptions

Field	Description
31–8 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
7–0 LISTSZ	Size of the LIST in number of elements. This field reflects the value of a hardware parameter.

46.6.12 LIST Channel Address Register (BCTU_LISTCHR_*n*)

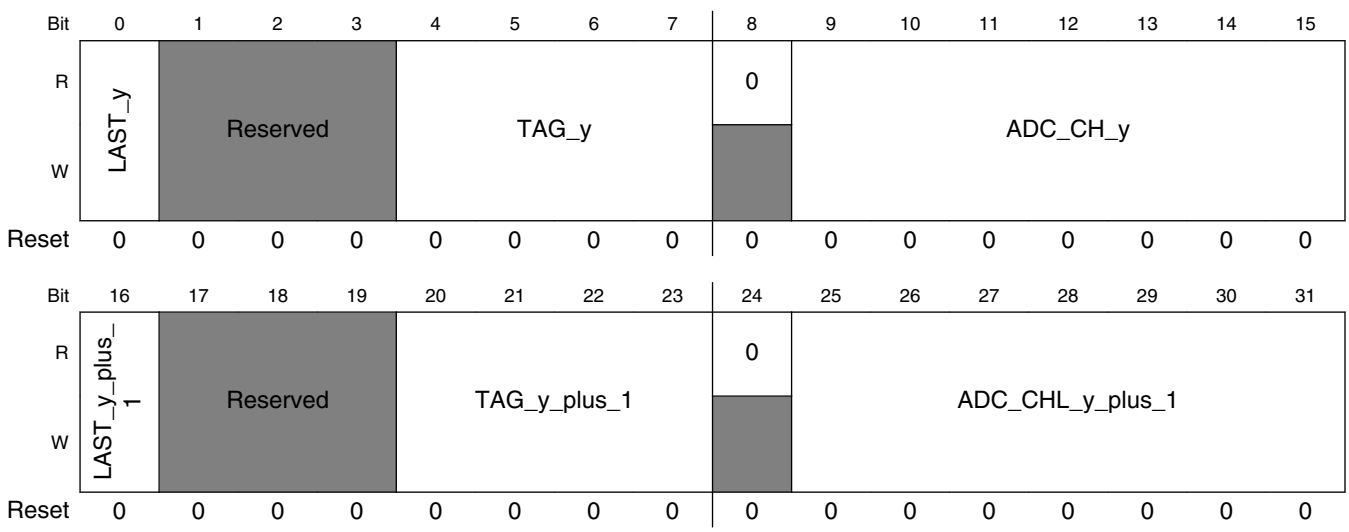
The LIST channel address register corresponds to two LIST elements which stores the ADC channel number to be used in a conversion as well as the indication if that element corresponds to the last element of a LIST of conversion commands.

NOTE

y varies from 0 to 63.

Access: User read/write

Address: 0h base + 250h offset + (4d × i), where i=0d to 31d



BCTU_LISTCHR_*n* field descriptions

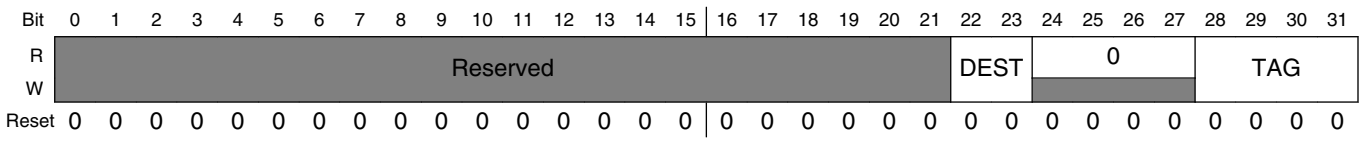
Field	Description
0 LAST_y	Last channel in the list of channels. 0 It is not the last channel in the list 1 This is the last channel in the List
1–3 Reserved	This field is reserved.
4–7 TAG_y	TAG for the n LIST element The TAG field is used in the Parallel Side Interface as an attribute to the WDATA bus. It is used by the external module connected to the PSI in order to correctly use the conversion result. TAG[3:0] used by the PSI to be transmitted along with the ADC conversion result.
8 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
9–15 ADC_CH_y	ADC channel selection. y indicates the position (address) of the ADC channel in the LIST of channels.
16 LAST_y_plus_1	Second to last channel in the list of channels.
17–19 Reserved	This field is reserved.
20–23 TAG_y_plus_1	TAG for the [y+1] LIST element. The TAG field is used in the Parallel Side Interface as an attribute to the WDATA bus. It is used by the external module connected to the PSI in order to correctly use the conversion result. TAG[3:0] used by the PSI to be transmitted along with the ADC conversion result.
24 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
25–31 ADC_CHL_y_ plus_1	ADC channel selection. y indicates the position (address) of the ADC channel in the LIST of channels.

46.6.13 Conversion configuration register (BCTU_CCR_*n*)

The CCR, conversion configuration register controls the destination of the conversion result. Possible choices are the result register and parallel side port, PSI interface to communicate with external SoC modules. These registers are organized in a Bank of registers. Each element in the bank is pointed by the CCP (conversion configuration pointer) field in the TRGCFG register.

Access: User read/write

Address: 0h base + 2D0h offset + (4d × i), where i=0d to 10d



BCTU_CCR_*n* field descriptions

Field	Description
0–21 Reserved	This field is reserved.
22–23 DEST	Destination selection The DEST field selects the destination of the conversion result. It points to the PSI as an alternate destination for the conversion results besides the result register. 00 conversion results are sent to result register 01 Reserved 10 conversion results are sent to the PSI and to result register 11 Reserved
24–27 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
28–31 TAG	The TAG field is used in the Parallel Side Port as the TAG bits. The TAG filed has only meaning if the parallel side port is selected as the destination of the conversion result. It is copy as is to the TAG bits in the PSI. In this case this bit field is used for further addressing in the modules connected to the side port. The TAG filed is used in the PSI (Parallel Side Port) in order to allow further addressing capabilities for the modules connected to this interface.

46.7 Functional description

The following sections describes the BCTU functionality.

46.7.1 Trigger Configuration

The triggers from on-chip timers and software are configured in the TRGCFG (Trigger Configuration Register). Each trigger can be individually enabled by the TRIGEN bit in the TRGCFG_n registers. The target ADC and channel are also defined in this register. The TRS (Trigger Resolution bit) sets if either a single conversion is executed for the corresponding trigger or if the LIST of conversions should be executed. In the case of a LIST the channel/LADDR field is used as a pointer for the LIST element. If the LIST is used, multiple conversions are executed due to the corresponding trigger assertion.

A protocol is implemented such that when an input trigger is selected to issue an ADC conversion, the BCTU clears the corresponding trigger source when the ADC conversion is initiated.

NOTE

If a LIST is triggered the corresponding trigger will be cleared when the last conversion in the LIST is started.

The software trigger is controlled by the SFTRGR (Software Trigger Register). By writing to a bit in that register, an ADC conversion is triggered similar to a conversion initiated by an input trigger. A software triggered conversion does not require to be enabled by the TRIGEN bit in the TRGCFG register. Thus if any bit in the SFTRGR register is set, the corresponding conversion is triggered independent of the TRIGEN bit state. This functionality is useful such that it avoids conflict between software and hardware triggers.

NOTE

The software triggered conversion is enabled by the GTRGEN bit in the MCR register meaning that this bit needs to be set in order to allow a software conversion. This enable bit does not prevent the SFTRGR bits to be set, it just disables the conversion start. As soon as the GTRGEN bit is set the software conversions already triggered are initiated following the predefined priority scheme, i.e. from the lowest to the highest SFTRG bit number.

46.7.2 Trigger priority and multiplexing

The ADC triggers are prioritized based on the trigger number. The lower the trigger input number the higher is the priority associated with it. Thus TRG[0] is the highest priority trigger and TRG[95], in case 96 triggers are used, is the lowest priority. The trigger input priority scheme applies when the triggers occur at the same time or when the ADC is busy in a conversion. Thus there are two possible scenarios for the trigger priority selection: when there are no on-going conversions and when the ADC is executing a conversion.

The figure below describes the BCTU input triggers related to the ADC trigger signal generation. There is four clock cycles from the input trigger to the ADC trigger. This figure also shows the CLR[n] (clear) signal generated by the BCTU which clears the input trigger signal.

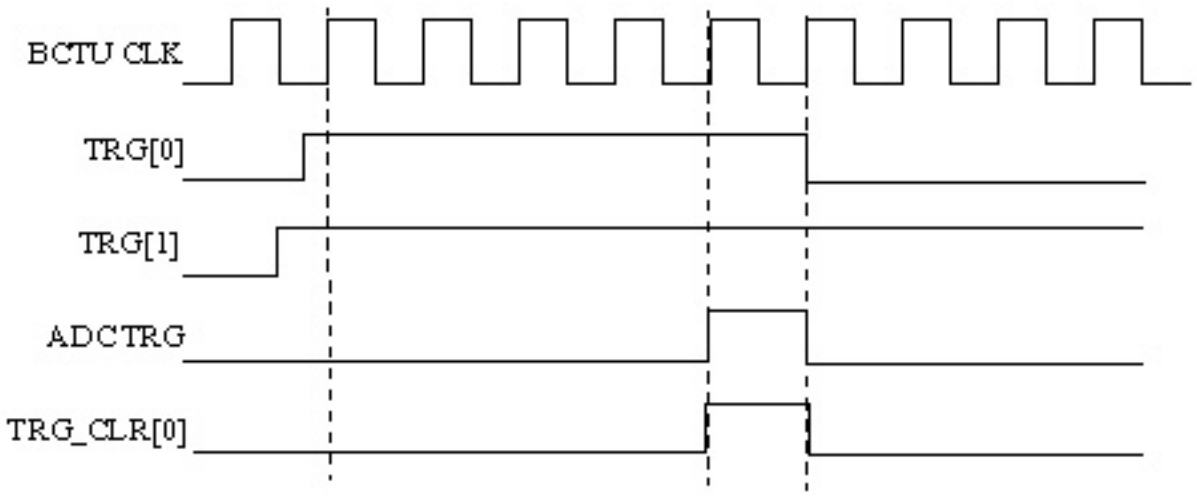


Figure 46-94. Input trigger to ADC trigger timing

The figure below describes the priority logic behavior in case TRG[0] and TRG[1] occur at the same time. Since TRG[0] has higher priority then it is selected for the ADC trigger generation. TRG[1] remains active (pending) until the conversion related to TRG[0] is executed. Note that if TRG[0] signal is asserted again, before the next trigger selection opportunity at the end of the current conversion, then it is selected one more time and TRG[1] remains asserted (pending) for more one ADC conversion time frame.

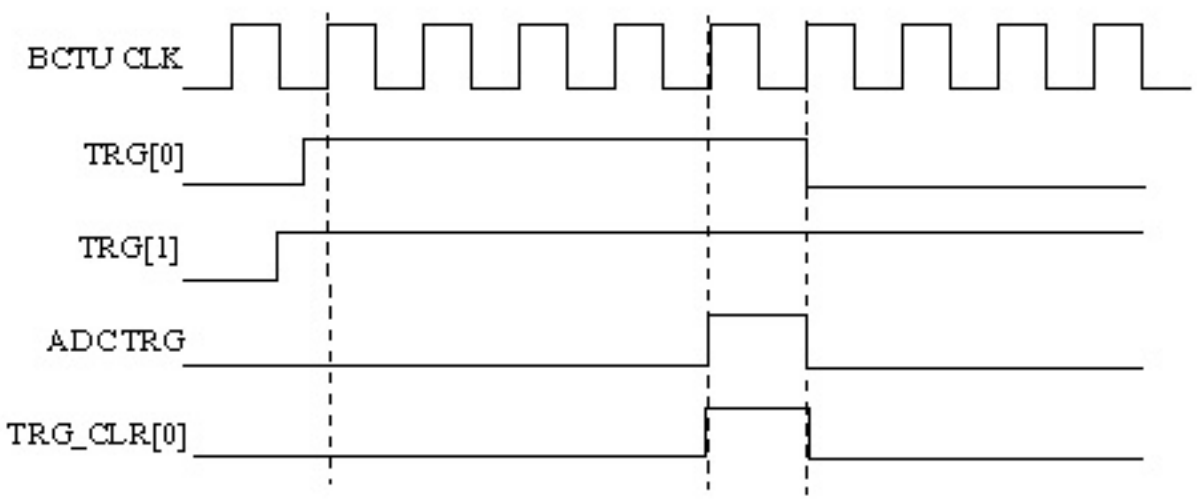


Figure 46-95. TRG[0] and TRG[1] priority example

The figure below describes the pipeline and logic (combinational logic) stages architecture, which resolves the priority among trigger inputs into two pipeline stages. The sel logic 1 and 2,combinational logic, are responsible for selecting the priority among all 96 inputs. The pipeline stage 3 stores this result and connects to a pulse generator stage which generates the trigger for the ADCs. Another stage isolates any

combinational logic in the BCTU to the external modules, thus allowing a clean clock domain crossing to connected to the ADC clock domain. In this architecture the timing from input trigger to the ADC trigger is 4 BCTU clock cycles.

NOTE

Back to back conversions are executed with 3 BCTU clock delay since the trigger selection is done during the current conversion thus the next trigger is ready when the conversion ends. The 4 clock delays occurs only when there are no conversions being executed by the ADC and new trigger occurs at the BCTU input.

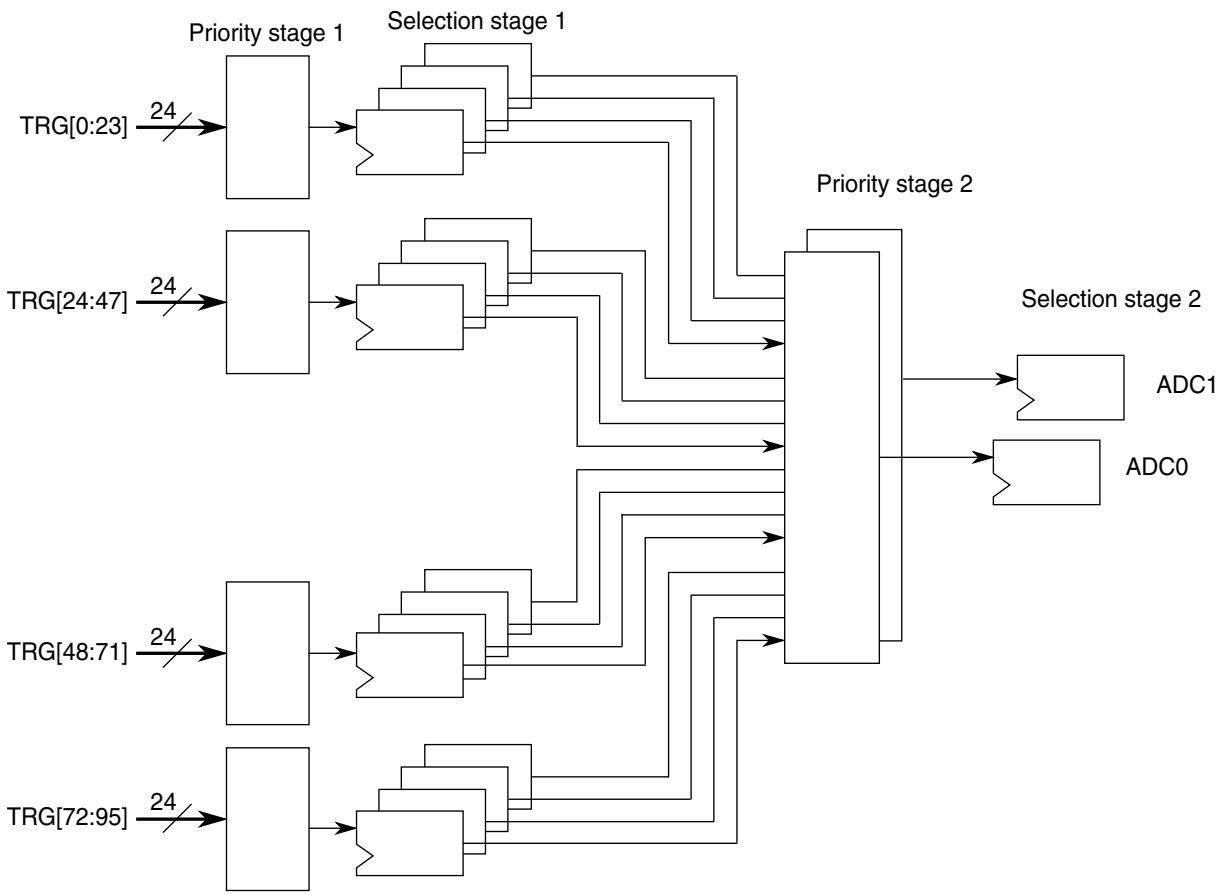
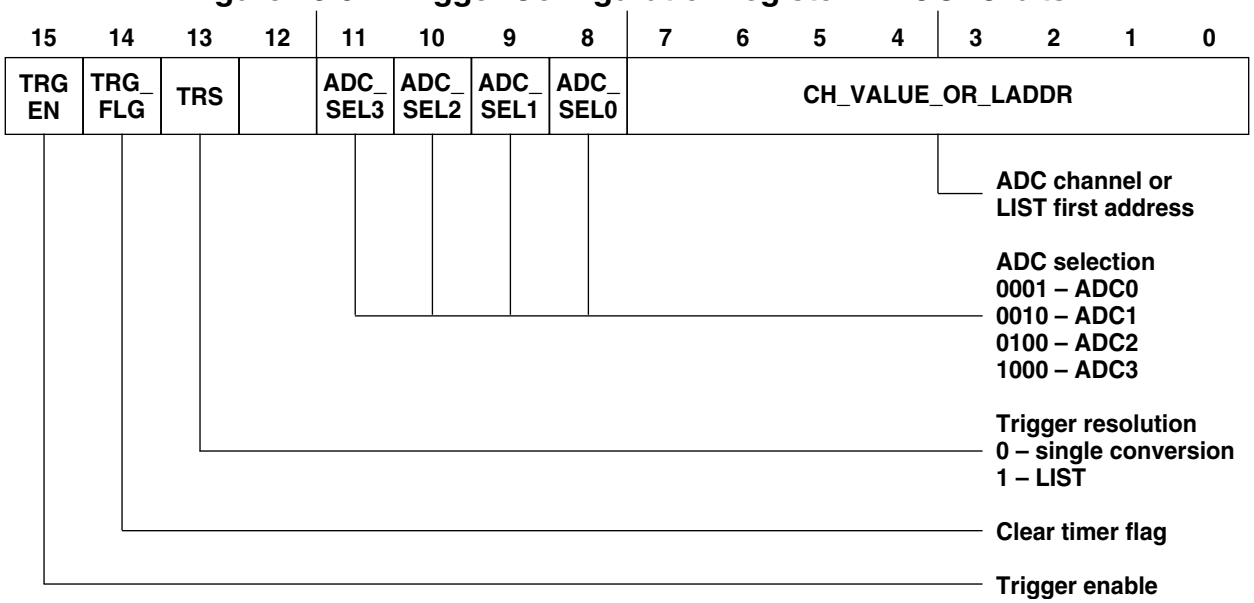


Figure 46-96. Priority stages architecture

46.7.3 Conversion LIST

The figure below describes the bits of the Trigger Configuration register (TRGCFG). The TRS bit selects between a single conversion and a list of conversions. In case a list of conversions is selected the CH_VALUE/LADDR field is used as the list pointer. The TRG_EN and CLR_FLAG bits have similar functionality regardless of single or list of conversion.

Figure 46-97. Trigger Configuration register -TRGCFG bits

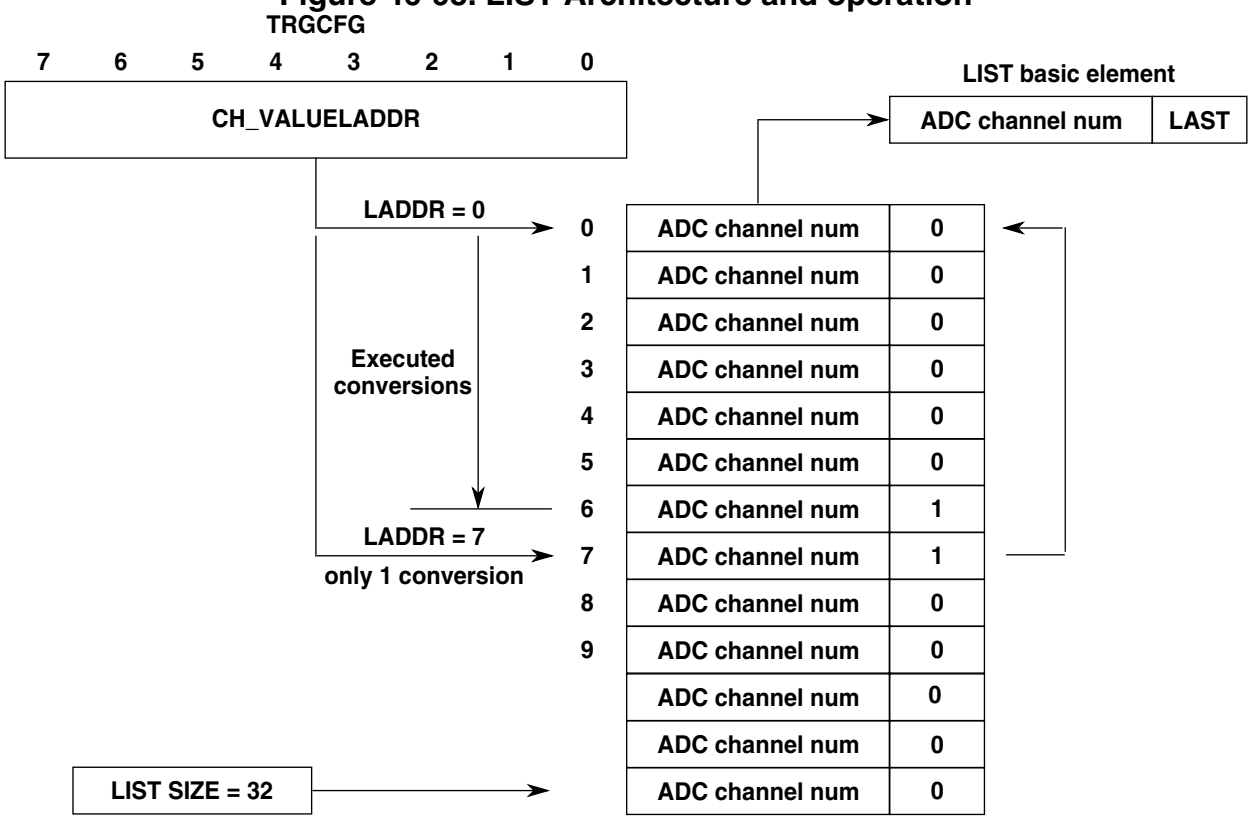


NOTE

ADC_SEL bits are one hot encoded, meaning that more than one ADC can be selected for a conversion. The multiple ADC selection is only valid for multiple parallel conversions LIST functionality. If more than one ADC is selected for a single conversion, instead of for a LIST, no conversion will be triggered by this register.

The figure below describes the LIST of channels architecture and operation. If the LADDR field in the TRGCFG register points to a LIST element then a sequence of conversions is initiated. The sequence of conversions are executed until the LAST field is set. The channel with the LAST field set is the last channel to be converted in a sequence of conversions.

Figure 46-98. LIST Architecture and operation



If LADDR points to a LIST element with the LAST field set, only one conversion is executed. In this case the LIST behaves such as a single conversion.

The LIST of channels has the following rules:

- A sequence of conversions is executed until the LAST field is evaluated as 1
- Conversions are executed in a sequence, thus other conversions for the same ADC are not allowed in parallel (the LIST is atomic)
- Two or more triggers for different ADCs may point to the LIST. In this case both LISTs are executed in parallel for the selected ADCs
- LADDR should point to a valid list address < LIST SIZE, otherwise address 0 is used
- After LIST SIZE -1 address is executed the list pointer wraps to address 0

46.7.4 ADC conversion results access

The access to ADC results can be done by the software using the flags and interrupt request or by the DMA interface. The interrupt request is based on the Status Register (MSR) flags. The results are available in the ADCxDR registers, one for each ADC. This register also indicates the channel used in the conversion and the trigger input that initiated that conversion. In order to give support for conversions from a LIST, two bits are available in the ADCxDR register, the LIST and LAST bits. The LIST bit indicates that the conversion was generated by a LIST element. The LAST bit indicates that the data is the last conversion of a LIST.

The software access to converted data can be based on interrupt request. An interrupt request is generated if the IENx bit in the MCR register is set and the corresponding DMAx bit is not set. In this case on each available conversion data in the ADCxDR register an interrupt request is asserted. The software can then check which is the source of the interrupt by reading the MSR register. Note that more then one conversion data may be available at the same time, since the conversions on different ADCs are executed in parallel. The NDATAx bit in the MSR register indicates which ADCxDR has a new conversion data available. The NDATAx flag can be cleared by using the NDATAx_CLR bit in the MSR register.

If DMA operation is selected by the DMA bit in the MCR register, a new data in the ADCxDR registers generates a DMA request. The NDATAx flag is also asserted in this case. A DMA DONE signal from the DMA controller de-asserts the DMA request and clears the NDATAx bit.

In order to allow DMA operation for all ADCxDR registers it is required one DMA channel per ADC, thus four DMA channels if four ADCs are used.

In case a new ADC conversion data is available before the old data had been read from the ADCxDR register then the DATAOVRx flag, new data overrun flag, is set indicating that the previous data in the ADCxDR register was overwritten by a new conversion result. The DATAOVRx flag also sets an interrupt request which is also enabled by the IENx bit. This interrupt request is asserted independently of the DMA bit settings. The DATAOVRx flag can only be cleared by the software writing 1 to the DATAOVRx_CLR bit.

46.7.5 Trigger clearing mechanism

There are two sources of ADC conversion triggers: hardware trigger, from the trigger inputs and software triggers, from SFTRGR register

A logical level 1 is detected in the input and captured into an input trigger register. This register remains set until the related conversion starts. At this time, the input trigger register is cleared. If another logical level 1 is detected while the trigger register is already set, the new level is not considered by the BCTU hardware. If a new logical level 1 occurs at the same time the trigger register is being cleared, then the register remains set and a new conversion is triggered by this input.

NOTE

The new triggered conversion will participate in the trigger priority selection thus may not trigger the next conversion.

The software trigger behaves similarly to the hardware trigger. The difference is that the trigger register is set by a CPU write operation. The trigger register is cleared by the BCTU hardware when the related conversion starts. If the CPU tries to set the trigger register by writing to the SFTRGR register corresponding bit, while the register is already set, the new attempt to set the register is not considered by the BCTU hardware. If a set occurs at the same time the hardware is attempting to clear the same trigger register, then the set has precedence and the register remains set. A new conversion will be triggered following the trigger priority criteria implemented in the hardware.

In case a LIST of conversions is triggered, instead of a single conversion, the trigger register is cleared at the start of the last conversion in the LIST.

46.7.6 On-the-fly reconfiguration of BCTU registers

The support for on-the-fly reconfiguration of triggers, specially for the Trigger Configuration register is done by the TRG_FLAG. If this bit field is read as 1, then the related trigger had started a conversion which is still ongoing. Therefore it is not safe to change the register bits.

The process of changing the Trigger Configuration register should start by clearing the TRIGEN bit. This will disable any new conversion be started by the corresponding trigger. After that the value of the TRG_FLAG bit should be evaluated. If this bit is set then it indicates that the current ongoing conversion or LIST execution was started by the trigger controlled by this register. The software should then read the TRGCFG register until the TRG_FLAG bit is cleared, which indicates that the on-going conversion had finished. After that the register can be modified safely. No conversions occur while the TRIGEN bit is cleared. In order to make sure that any trigger that occurred before the TRGCFG reconfiguration starts a new conversion, the application software should write 1 to the TRG_FLAG bit, thus clearing the trigger sources. Only triggers that occurred after this write operation was done are able to start a new conversion.

46.7.7 Multiple Parallel Conversions

The Multiple Parallel Conversions functionality is implemented using the LIST of channels. In order to execute this function more than one ADC should be selected in the TRGCFG register and the TRS bit must be set. In this case the TRGCFG CH_VALUE/ LADDR field points to an element in the LIST. The channels in the LIST are interpreted as referring to the ADCs selected in the TRGCFG register. The figure below describes the parallel conversion execution. Note that the channel numbers in the LIST are interpreted in pairs since two ADCs are selected, ADC0 and ADC1. If three ADC are selected, the LIST is interpreted in groups of three elements, being the order from ADC0 through ADC2.

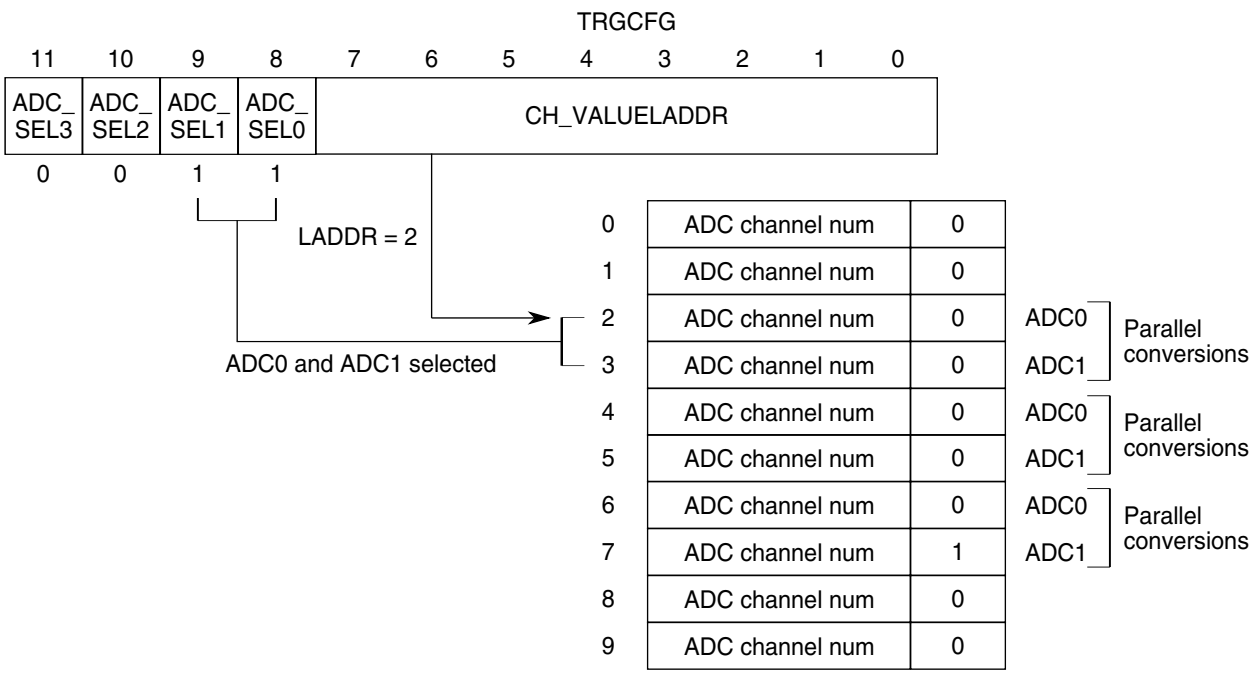


Figure 46-99. Multiple Parallel Conversions LIST operation

The Multiple Parallel Conversion (MPC) follows the same priority rules of a single conversion which is based on the trigger input number: the lower the trigger the higher the priority is. The difference is that a MPC cannot start until all the ADCs defined in the LIST are free. Thus if a MPC is triggered for ADC0 and ADC1, this LIST of conversions does not initiate until ADC0 and ADC1 are free. Note that if ADC0 is free but there is a higher priority trigger for ADC1 then the MPC does not start. If two MPC that have at least one ADC in common are triggered, the lower trigger number has higher priority. Note that the conflict in the priority only occurs if the triggered conversions are trying to use at least one ADC in common. If the MPCs are using different ADCs than there is no conflict and the MPCs are executed at the same time.

46.8 Interrupts and DMA requests

The BCTU module has several interrupts to signal input triggers and conversion results events. These interrupts can also be used as DMA requests.

Trigger interrupt : signals an ADC trigger event occurred, thus a conversion have started. This interrupt line is an or of all ADC triggers (up to four).

ADC data ready: these interrupts (up to four) signals when an ADC result is available. The software should read the BCTU_ADCx_DR register in order to access the ADC conversion result.

LIST interrupt: the LIST interrupt line signals when the LIST executed the last conversion. If the conversion from a LIST was transferred to the system memory then this interrupt signals to the software that all results, from a LIST, are ready to be processed. Note that this interrupt always occur along with the ADC data ready interrupt.

Chapter 47
CAN (FlexCAN)

47.1 CAN chip-specific module information

This section describes the implementation of this module in the device.

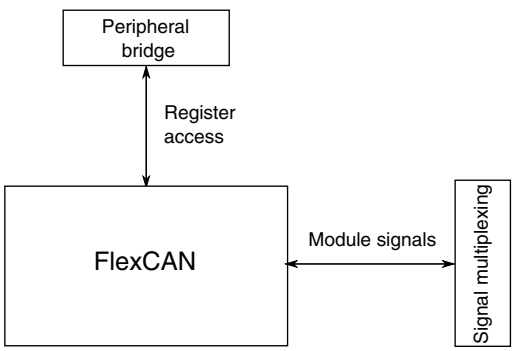


Figure 47-1. CAN chip-specific module information

47.1.1 FlexCAN modules on this chip

This device contains four FlexCAN module instances. Their differences are as follows.

Table 47-1. FlexCAN module instances

Instance name	Number of message buffers	CAN_CTRL2[TASD] reset value
CAN0	96	0b01100
CAN1	96	0b01100
CAN2	64	0b10000
CAN3	64	0b10000

47.1.2 Message buffer size

Each message buffer is 16 bytes.

47.1.3 Message buffer address offsets

The following table lists the address offsets of the message buffers for the CAN0 and CAN1.

For the CAN2 and CAN3, the message buffers go up to only MB63.

Table 47-2. Message buffer address offsets for CAN0 and CAN1

Offset	Message buffer
Base + 0x0080	MB0
Base + 0x0090	MB1
Base + 0x00A0	MB2
Base + 0x00B0	MB3
Base + 0x00C0	MB4
Base + 0x00D0	MB5
Base + 0x00E0	MB6
Base + 0x00F0	MB7
Base + 0x0100	MB8
Base + 0x0110	MB9
Base + 0x0120	MB10
Base + 0x0130	MB11
Base + 0x0140	MB12
Base + 0x0150	MB13
Base + 0x0160	MB14
Base + 0x0170	MB15
Base + 0x0180	MB16
Base + 0x0190	MB17
Base + 0x01A0	MB18
Base + 0x01B0	MB19
Base + 0x01C0	MB20
Base + 0x01D0	MB21
Base + 0x01E0	MB22
Base + 0x01F0	MB23
Base + 0x0200	MB24
Base + 0x0210	MB25
Base + 0x0220	MB26
Base + 0x0230	MB27

Table continues on the next page...

Table 47-2. Message buffer address offsets for CAN0 and CAN1 (continued)

Offset	Message buffer
Base + 0x0240	MB28
Base + 0x0250	MB29
Base + 0x0260	MB30
Base + 0x0270	MB31
Base + 0x0280	MB32
Base + 0x0290	MB33
Base + 0x02A0	MB34
Base + 0x02B0	MB35
Base + 0x02C0	MB36
Base + 0x02D0	MB37
Base + 0x02E0	MB38
Base + 0x02F0	MB39
Base + 0x0300	MB40
Base + 0x0310	MB41
Base + 0x0320	MB42
Base + 0x0330	MB43
Base + 0x0340	MB44
Base + 0x0350	MB45
Base + 0x0360	MB46
Base + 0x0370	MB47
Base + 0x0380	MB48
Base + 0x0390	MB49
Base + 0x03A0	MB50
Base + 0x03B0	MB51
Base + 0x03C0	MB52
Base + 0x03D0	MB53
Base + 0x03E0	MB54
Base + 0x03F0	MB55
Base + 0x0400	MB56
Base + 0x0410	MB57
Base + 0x0420	MB58
Base + 0x0430	MB59
Base + 0x0440	MB60
Base + 0x0450	MB61
Base + 0x0460	MB62
Base + 0x0470	MB63
Base + 0x0480	MB64
Base + 0x0490	MB65
Base + 0x04A0	MB66

Table continues on the next page...

Table 47-2. Message buffer address offsets for CAN0 and CAN1 (continued)

Offset	Message buffer
Base + 0x04B0	MB67
Base + 0x04C0	MB68
Base + 0x04D0	MB69
Base + 0x04E0	MB70
Base + 0x04F0	MB71
Base + 0x0500	MB72
Base + 0x0510	MB73
Base + 0x0520	MB74
Base + 0x0530	MB75
Base + 0x0540	MB76
Base + 0x0550	MB77
Base + 0x0560	MB78
Base + 0x0570	MB79
Base + 0x0580	MB80
Base + 0x0590	MB81
Base + 0x05A0	MB82
Base + 0x05B0	MB83
Base + 0x05C0	MB84
Base + 0x05D0	MB85
Base + 0x05E0	MB86
Base + 0x05F0	MB87
Base + 0x0600	MB88
Base + 0x0610	MB89
Base + 0x0620	MB90
Base + 0x0630	MB91
Base + 0x0640	MB92
Base + 0x0650	MB93
Base + 0x0660	MB94
Base + 0x0670	MB95

47.1.4 Reset value of MDIS bit

The CAN_MCR[MDIS] bit is set after reset. Therefore, FlexCAN module is disabled following a reset.

47.1.5 FlexCAN Clocking

The clock to each CAN module can be gated on and off using the MC_ME[PCTL] bits. These bits are cleared after any reset, which disables the clock to the corresponding module. The appropriate clock enable bit should be set by software at the beginning of the FlexCAN initialization routine to enable the module clock before attempting to initialize any of the FlexCAN registers.

47.1.6 FlexCAN Interrupts

The FlexCAN has multiple sources of interrupt requests. However, some of these sources are OR'd together to generate a single interrupt request. See below for the mapping of the individual interrupt sources to the interrupt request.

Table 47-3. Interrupts for CAN modules

Request	Sources
Message buffer	Message buffers 0-63
Bus off	Bus off
Error	• Bit1 error • Bit0 error • Acknowledge error • Cyclic redundancy check (CRC) error • Form error • Stuffing error • Transmit error warning • Receive error warning

For the mapping of interrupt requests and sources, see section Interrupt sources in the Interrupt Controller chapter.

47.1.7 FlexCAN CLKSRC

This signal is not used on the MPC5746R. The clock source selection is controlled by MC_CGM_AC8_SC in the Clock Generation Module chapter.

47.1.8 FlexCAN RAM to memory map address mapping

Use the following table to convert from the memory map address to the location in the physical FlexCAN RAM where the error is to be injected, defined in CAN_ERRIAR[INJADDR].

Table 47-4. FlexCAN RAM to memory map address mapping

RAM contents	Injection address	Memory map
FlexCAN Registers	Not mapped	-
MBs	0x0000	0x0080
Reserved	-	0x0480
RXIMRs	0x0400	0x0880
Reserved	-	0x0980
RXFIR_0	0x0500	0x0A80
RXFIR_1	0x0504	0x0A84
RXFIR_2	0x0508	0x0A88
RXFIR_3	0x050C	0x0A8C
RXFIR_4	0x0510	0x0A90
RXFIR_5	0x0514	0x0A94
Reserved	-	0x0A98
RXMGMASK	0x0520	0x0AA0
RXFGMASK	0x0524	0x0AA4
RX14MASK	0x0528	0x0AA8
RX15MASK	0x052C	0x0AAC
Tx_SMB	0x0530	0x0AB0
Rx_SMB0	0x0540	0x0AC0
Rx_SMB1	0x0550	0x0AD0
ECC Registers	Not mapped	0x0AE0
Reserved	-	0x0B00
Reserved	-	0x0C00
Reserved	-	0x0C0C

47.2 Introduction

The FlexCAN module is a communication controller implementing the CAN protocol according to the ISO 11898-1 standard and CAN 2.0 B protocol specifications. A general block diagram is shown in the following figure, which describes the main subblocks implemented in the FlexCAN module, including one associated memory for storing message buffers, Receive Global Mask registers, Receive Individual Mask registers, Receive FIFO filters, and Receive FIFO ID filters. The functions of the submodules are described in subsequent sections.

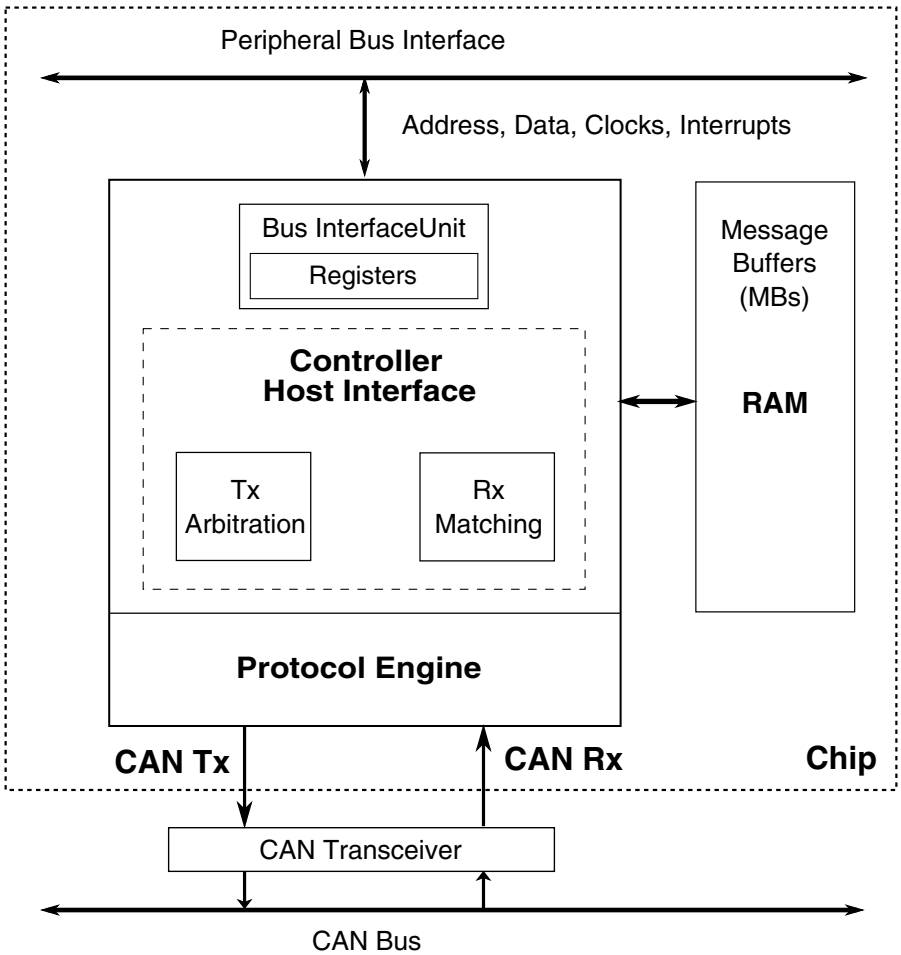


Figure 47-2. FlexCAN block diagram

47.2.1 Overview

The CAN protocol was primarily designed to be used as a vehicle serial data bus, meeting the specific requirements of this field:

- Real-time processing
- Reliable operation in the EMI environment of a vehicle
- Cost-effectiveness
- Required bandwidth

The FlexCAN module is a full implementation of the CAN protocol specification, the CAN 2.0 version B protocol, which supports both standard and extended message frames. The message buffers are stored in an embedded RAM dedicated to the FlexCAN module. See the chip configuration details for the actual number of message buffers configured in the chip.

The Protocol Engine (PE) submodule manages the serial communication on the CAN bus:

- Requesting RAM access for receiving and transmitting message frames
- Validating received messages
- Performing error handling

The Controller Host Interface (CHI) sub-module handles message buffer selection for reception and transmission, taking care of arbitration and ID matching algorithms.

The Bus Interface Unit (BIU) sub-module controls the access to and from the internal interface bus, in order to establish connection to the CPU and to other blocks. Clocks, address and data buses, interrupt outputs, DMA and test signals are accessed through the BIU.

47.2.2 FlexCAN module features

The FlexCAN module includes these distinctive features:

- Full implementation of the CAN protocol specification, Version 2.0 B
    - Standard data frames
    - Extended data frames
    - Zero to eight bytes data length
    - Programmable bit rate up to 1 Mb/sec
    - Content-related addressing
- Compliant with the ISO 11898-1 standard
- Flexible mailboxes of zero to eight bytes data length
- Each mailbox configurable as receive or transmit, all supporting standard and extended messages
- Individual Rx Mask registers per mailbox
- Full-featured Rx FIFO with storage capacity for up to six frames and automatic internal pointer handling with DMA support
- Transmission abort capability
- Flexible message buffers (MBs), totaling 96 message buffers of 8 bytes data length each, configurable as Rx or Tx

- RAM not used by reception or transmission structures can be used as general purpose RAM space
- Listen-Only mode capability
- Programmable Loop-Back mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number, or highest priority
- Time stamp based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Maskable interrupts
- Independence from the transmission medium (an external transceiver is assumed)
- Short latency time due to an arbitration scheme for high-priority messages
- Low power modes
- Remote request frames may be handled automatically or by software
- CAN bit time settings and configuration bits can only be written in Freeze mode
- Tx mailbox status (Lowest priority buffer or empty buffer)
- Identifier Acceptance Filter Hit Indicator (IDHIT) register for received frames
- SYNCH bit available in Error in Status 1 register to inform that the module is synchronous with CAN bus
- CRC status for transmitted message
- Rx FIFO Global Mask register
- Selectable priority between mailboxes and Rx FIFO during matching process
- Powerful Rx FIFO ID filtering, capable of matching incoming IDs against either 128 extended, 256 standard, or 512 partial (8 bit) IDs, with up to 32 individual masking capability
- 100% backward compatibility with previous FlexCAN version
- Supports detection and correction of errors in memory read accesses. Each byte of FlexCAN memory is associated to 5 parity bits and the error correction mechanism ensures that in this 13-bit word, errors in one bit can be corrected (correctable errors) and errors in 2 bits can be detected but not corrected (non-correctable errors)

47.2.3 Modes of operation

The FlexCAN module has these functional modes:

- Normal mode (User or Supervisor):

  In Normal mode, the module operates receiving and/or transmitting message frames, errors are handled normally, and all CAN Protocol functions are enabled. User and Supervisor Modes differ in the access to some restricted control registers.

- Freeze mode:

  Freeze mode is enabled when the FRZ bit in MCR is asserted. If enabled, Freeze mode is entered when MCR[HALT] is set or when Debug mode is requested at chip level and MCR[FRZ_ACK] is asserted by the FlexCAN. In this mode, no transmission or reception of frames is done and synchronicity to the CAN bus is lost. See Freeze mode for more information.

- Loop-Back mode:

  The module enters this mode when the LPB field in the Control 1 Register is asserted. In this mode, FlexCAN performs an internal loop back that can be used for self-test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic '1'). FlexCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.

- Listen-Only mode:

  The module enters this mode when the LOM field in the Control 1 Register is asserted. In this mode, transmission is disabled, all error counters are frozen, and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.

For low-power operation, the FlexCAN module has:

- Module Disable mode:

This low-power mode is entered when the MDIS bit in the MCR Register is asserted by the CPU and the LPM_ACK is asserted by the FlexCAN. When disabled, the module requests to disable the clocks to the CAN Protocol Engine and Controller Host Interface submodules. Exit from this mode is done by negating the MDIS bit in the MCR register. See Module Disable mode for more information.

- Stop mode:

  This low power mode is entered when Stop mode is requested at chip level and the LPM_ACK bit in the MCR Register is asserted by the FlexCAN. When in Stop Mode, the module puts itself in an inactive state and then informs the CPU that the clocks can be shut down globally. Exit from this mode happens when the Stop mode request is removed. See Stop mode for more information.

47.3 FlexCAN signal descriptions

The FlexCAN module has two I/O signals connected to the external chip pins. These signals are summarized in the following table and described in more detail in the next subsections.

Table 47-5. FlexCAN signal descriptions

Signal	Description	I/O
CAN Rx	CAN Receive Pin	Input
CAN Tx	CAN Transmit Pin	Output

47.3.1 CAN Rx

This pin is the receive pin from the CAN bus transceiver. Dominant state is represented by logic level 0. Recessive state is represented by logic level 1.

47.3.2 CAN Tx

This pin is the transmit pin to the CAN bus transceiver. Dominant state is represented by logic level 0. Recessive state is represented by logic level 1.

47.4 Memory map/register definition

This section describes the registers and data structures in the FlexCAN module. The base address of the module depends on the particular memory map of the chip.

47.4.1 FlexCAN memory mapping

The memory map for the FlexCAN module is shown in the following table.

The address space occupied by FlexCAN has 128 bytes for registers starting at the module base address, followed by embedded RAM starting at address offset 0x0080.

Each individual register is identified by its complete name and the corresponding mnemonic. The access type can be Supervisor (S) or Unrestricted (U). Most of the registers can be configured to have either Supervisor or Unrestricted access by programming the SUPV field in the MCR register. These registers are identified as S/U in the Access column of Table 47-6.

Table 47-6. Register access and reset information

Register	Access type	Affected by hard reset	Affected by soft reset
Module Configuration Register (CAN_MCR)	S	Yes	Yes
Control 1 register (CAN_CTRL1)	S/U	Yes	No
Free Running Timer register (CAN_TIMER)	S/U	Yes	Yes
Rx Mailboxes Global Mask register (CAN_RXMGMASK)	S/U	No	No
Rx Buffer 14 Mask register (CAN_RX14MASK)	S/U	No	No
Rx Buffer 15 Mask register (CAN_RX15MASK)	S/U	No	No
Error Counter Register (CAN_ECR)	S/U	Yes	Yes
Error and Status 1 Register (CAN_ESR1)	S/U	Yes	Yes
Interrupt Masks 2 register (CAN_IMASK2)	S/U	Yes	Yes
Interrupt Masks 1 register (CAN_IMASK1)	S/U	Yes	Yes
Interrupt Flags 2 register (CAN_IFLAG2)	S/U	Yes	Yes
Interrupt Flags 1 register (CAN_IFLAG1)	S/U	Yes	Yes
Control 2 Register (CAN_CTRL2)	S/U	Yes	No
Error and Status 2 Register (CAN_ESR2)	S/U	Yes	Yes
CRC Register (CAN_CRCR)	S/U	Yes	Yes
Rx FIFO Global Mask register (CAN_RXFGMASK)	S/U	No	No
Rx FIFO Information Register (CAN_RXFIR)	S/U	No	No
CAN Bit Timing Register (CAN_CBT)	S/U	Yes	No
Interrupt Masks 3 register (CAN_IMASK3)	S/U	Yes	Yes

Table continues on the next page...

Table 47-6. Register access and reset information (continued)

Register	Access type	Affected by hard reset	Affected by soft reset
Interrupt Flags 3 register (CAN_IFLAG3)	S/U	Yes	Yes
Message buffers	S/U	No	No
Rx Individual Mask Registers	S/U	No	No
Memory Error Control register (CAN_MECR)	S/U	Yes	Yes
Error Injection Address register (CAN_ERRIAR)	S/U	Yes	Yes
Error Injection Data Pattern register (CAN_ERRIDPR)	S/U	Yes	Yes
Error Injection Parity Pattern register (CAN_ERRIPPR)	S/U	Yes	Yes
Error Report Address register (CAN_RERRAR)	S/U	Yes	Yes
Error Report Data register (CAN_RERRDR)	S/U	Yes	Yes
Error Report Syndrome register (CAN_RERRSYNR)	S/U	Yes	Yes
Error Status register (CAN_ERRSR)	S/U	Yes	Yes

The FlexCAN module can store CAN messages for transmission and reception using mailboxes and Rx FIFO structures.

The table below shows the FlexCAN memory map.

The address range from offset 0x80 to 0x67F allocates the ninety-six 128-bit Message Buffers (MBs).

CAN memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (CAN_MCR)	32	R/W	See section	47.4.2/1842
4	Control 1 register (CAN_CTRL1)	32	R/W	0000_0000h	47.4.3/1847
8	Free Running Timer (CAN_TIMER)	32	R/W	0000_0000h	47.4.4/1851
10	Rx Mailboxes Global Mask Register (CAN_RXMGMASK)	32	R/W	Undefined	47.4.5/1851
14	Rx 14 Mask register (CAN_RX14MASK)	32	R/W	Undefined	47.4.6/1853
18	Rx 15 Mask register (CAN_RX15MASK)	32	R/W	Undefined	47.4.7/1853
1C	Error Counter (CAN_ECR)	32	R/W	0000_0000h	47.4.8/1854
20	Error and Status 1 register (CAN_ESR1)	32	R/W	See section	47.4.9/1856
24	Interrupt Masks 2 register (CAN_IMASK2)	32	R/W	0000_0000h	47.4.10/ 1862
28	Interrupt Masks 1 register (CAN_IMASK1)	32	R/W	0000_0000h	47.4.11/ 1862
2C	Interrupt Flags 2 register (CAN_IFLAG2)	32	R/W	0000_0000h	47.4.12/ 1863
30	Interrupt Flags 1 register (CAN_IFLAG1)	32	R/W	0000_0000h	47.4.13/ 1864

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
34	Control 2 register (CAN_CTRL2)	32	R/W	See section	47.4.14/ 1867
38	Error and Status 2 register (CAN_ESR2)	32	R/W	0000_0000h	47.4.15/ 1870
44	CRC Register (CAN_CRCR)	32	R	0000_0000h	47.4.16/ 1872
48	Rx FIFO Global Mask register (CAN_RXFGMASK)	32	R/W	Undefined	47.4.17/ 1872
4C	Rx FIFO Information Register (CAN_RXFIR)	32	R	Undefined	47.4.18/ 1873
50	CAN Bit Timing Register (CAN_CBT)	32	R/W	See section	47.4.19/ 1874
6C	Interrupt Masks 3 Register (CAN_IMASK3)	32	R/W	0000_0000h	47.4.20/ 1876
74	Interrupt Flags 3 Register (CAN_IFLAG3)	32	R/W	0000_0000h	47.4.21/ 1877
880	Rx Individual Mask Registers (CAN_RXIMR0)	32	R/W	Undefined	47.4.22/ 1877
884	Rx Individual Mask Registers (CAN_RXIMR1)	32	R/W	Undefined	47.4.22/ 1877
888	Rx Individual Mask Registers (CAN_RXIMR2)	32	R/W	Undefined	47.4.22/ 1877
88C	Rx Individual Mask Registers (CAN_RXIMR3)	32	R/W	Undefined	47.4.22/ 1877
890	Rx Individual Mask Registers (CAN_RXIMR4)	32	R/W	Undefined	47.4.22/ 1877
894	Rx Individual Mask Registers (CAN_RXIMR5)	32	R/W	Undefined	47.4.22/ 1877
898	Rx Individual Mask Registers (CAN_RXIMR6)	32	R/W	Undefined	47.4.22/ 1877
89C	Rx Individual Mask Registers (CAN_RXIMR7)	32	R/W	Undefined	47.4.22/ 1877
8A0	Rx Individual Mask Registers (CAN_RXIMR8)	32	R/W	Undefined	47.4.22/ 1877
8A4	Rx Individual Mask Registers (CAN_RXIMR9)	32	R/W	Undefined	47.4.22/ 1877
8A8	Rx Individual Mask Registers (CAN_RXIMR10)	32	R/W	Undefined	47.4.22/ 1877
8AC	Rx Individual Mask Registers (CAN_RXIMR11)	32	R/W	Undefined	47.4.22/ 1877
8B0	Rx Individual Mask Registers (CAN_RXIMR12)	32	R/W	Undefined	47.4.22/ 1877
8B4	Rx Individual Mask Registers (CAN_RXIMR13)	32	R/W	Undefined	47.4.22/ 1877

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8B8	Rx Individual Mask Registers (CAN_RXIMR14)	32	R/W	Undefined	47.4.22/ 1877
8BC	Rx Individual Mask Registers (CAN_RXIMR15)	32	R/W	Undefined	47.4.22/ 1877
8C0	Rx Individual Mask Registers (CAN_RXIMR16)	32	R/W	Undefined	47.4.22/ 1877
8C4	Rx Individual Mask Registers (CAN_RXIMR17)	32	R/W	Undefined	47.4.22/ 1877
8C8	Rx Individual Mask Registers (CAN_RXIMR18)	32	R/W	Undefined	47.4.22/ 1877
8CC	Rx Individual Mask Registers (CAN_RXIMR19)	32	R/W	Undefined	47.4.22/ 1877
8D0	Rx Individual Mask Registers (CAN_RXIMR20)	32	R/W	Undefined	47.4.22/ 1877
8D4	Rx Individual Mask Registers (CAN_RXIMR21)	32	R/W	Undefined	47.4.22/ 1877
8D8	Rx Individual Mask Registers (CAN_RXIMR22)	32	R/W	Undefined	47.4.22/ 1877
8DC	Rx Individual Mask Registers (CAN_RXIMR23)	32	R/W	Undefined	47.4.22/ 1877
8E0	Rx Individual Mask Registers (CAN_RXIMR24)	32	R/W	Undefined	47.4.22/ 1877
8E4	Rx Individual Mask Registers (CAN_RXIMR25)	32	R/W	Undefined	47.4.22/ 1877
8E8	Rx Individual Mask Registers (CAN_RXIMR26)	32	R/W	Undefined	47.4.22/ 1877
8EC	Rx Individual Mask Registers (CAN_RXIMR27)	32	R/W	Undefined	47.4.22/ 1877
8F0	Rx Individual Mask Registers (CAN_RXIMR28)	32	R/W	Undefined	47.4.22/ 1877
8F4	Rx Individual Mask Registers (CAN_RXIMR29)	32	R/W	Undefined	47.4.22/ 1877
8F8	Rx Individual Mask Registers (CAN_RXIMR30)	32	R/W	Undefined	47.4.22/ 1877
8FC	Rx Individual Mask Registers (CAN_RXIMR31)	32	R/W	Undefined	47.4.22/ 1877
900	Rx Individual Mask Registers (CAN_RXIMR32)	32	R/W	Undefined	47.4.22/ 1877
904	Rx Individual Mask Registers (CAN_RXIMR33)	32	R/W	Undefined	47.4.22/ 1877
908	Rx Individual Mask Registers (CAN_RXIMR34)	32	R/W	Undefined	47.4.22/ 1877
90C	Rx Individual Mask Registers (CAN_RXIMR35)	32	R/W	Undefined	47.4.22/ 1877

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
910	Rx Individual Mask Registers (CAN_RXIMR36)	32	R/W	Undefined	47.4.22/ 1877
914	Rx Individual Mask Registers (CAN_RXIMR37)	32	R/W	Undefined	47.4.22/ 1877
918	Rx Individual Mask Registers (CAN_RXIMR38)	32	R/W	Undefined	47.4.22/ 1877
91C	Rx Individual Mask Registers (CAN_RXIMR39)	32	R/W	Undefined	47.4.22/ 1877
920	Rx Individual Mask Registers (CAN_RXIMR40)	32	R/W	Undefined	47.4.22/ 1877
924	Rx Individual Mask Registers (CAN_RXIMR41)	32	R/W	Undefined	47.4.22/ 1877
928	Rx Individual Mask Registers (CAN_RXIMR42)	32	R/W	Undefined	47.4.22/ 1877
92C	Rx Individual Mask Registers (CAN_RXIMR43)	32	R/W	Undefined	47.4.22/ 1877
930	Rx Individual Mask Registers (CAN_RXIMR44)	32	R/W	Undefined	47.4.22/ 1877
934	Rx Individual Mask Registers (CAN_RXIMR45)	32	R/W	Undefined	47.4.22/ 1877
938	Rx Individual Mask Registers (CAN_RXIMR46)	32	R/W	Undefined	47.4.22/ 1877
93C	Rx Individual Mask Registers (CAN_RXIMR47)	32	R/W	Undefined	47.4.22/ 1877
940	Rx Individual Mask Registers (CAN_RXIMR48)	32	R/W	Undefined	47.4.22/ 1877
944	Rx Individual Mask Registers (CAN_RXIMR49)	32	R/W	Undefined	47.4.22/ 1877
948	Rx Individual Mask Registers (CAN_RXIMR50)	32	R/W	Undefined	47.4.22/ 1877
94C	Rx Individual Mask Registers (CAN_RXIMR51)	32	R/W	Undefined	47.4.22/ 1877
950	Rx Individual Mask Registers (CAN_RXIMR52)	32	R/W	Undefined	47.4.22/ 1877
954	Rx Individual Mask Registers (CAN_RXIMR53)	32	R/W	Undefined	47.4.22/ 1877
958	Rx Individual Mask Registers (CAN_RXIMR54)	32	R/W	Undefined	47.4.22/ 1877
95C	Rx Individual Mask Registers (CAN_RXIMR55)	32	R/W	Undefined	47.4.22/ 1877
960	Rx Individual Mask Registers (CAN_RXIMR56)	32	R/W	Undefined	47.4.22/ 1877
964	Rx Individual Mask Registers (CAN_RXIMR57)	32	R/W	Undefined	47.4.22/ 1877

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
968	Rx Individual Mask Registers (CAN_RXIMR58)	32	R/W	Undefined	47.4.22/ 1877
96C	Rx Individual Mask Registers (CAN_RXIMR59)	32	R/W	Undefined	47.4.22/ 1877
970	Rx Individual Mask Registers (CAN_RXIMR60)	32	R/W	Undefined	47.4.22/ 1877
974	Rx Individual Mask Registers (CAN_RXIMR61)	32	R/W	Undefined	47.4.22/ 1877
978	Rx Individual Mask Registers (CAN_RXIMR62)	32	R/W	Undefined	47.4.22/ 1877
97C	Rx Individual Mask Registers (CAN_RXIMR63)	32	R/W	Undefined	47.4.22/ 1877
980	Rx Individual Mask Registers (CAN_RXIMR64)	32	R/W	Undefined	47.4.22/ 1877
984	Rx Individual Mask Registers (CAN_RXIMR65)	32	R/W	Undefined	47.4.22/ 1877
988	Rx Individual Mask Registers (CAN_RXIMR66)	32	R/W	Undefined	47.4.22/ 1877
98C	Rx Individual Mask Registers (CAN_RXIMR67)	32	R/W	Undefined	47.4.22/ 1877
990	Rx Individual Mask Registers (CAN_RXIMR68)	32	R/W	Undefined	47.4.22/ 1877
994	Rx Individual Mask Registers (CAN_RXIMR69)	32	R/W	Undefined	47.4.22/ 1877
998	Rx Individual Mask Registers (CAN_RXIMR70)	32	R/W	Undefined	47.4.22/ 1877
99C	Rx Individual Mask Registers (CAN_RXIMR71)	32	R/W	Undefined	47.4.22/ 1877
9A0	Rx Individual Mask Registers (CAN_RXIMR72)	32	R/W	Undefined	47.4.22/ 1877
9A4	Rx Individual Mask Registers (CAN_RXIMR73)	32	R/W	Undefined	47.4.22/ 1877
9A8	Rx Individual Mask Registers (CAN_RXIMR74)	32	R/W	Undefined	47.4.22/ 1877
9AC	Rx Individual Mask Registers (CAN_RXIMR75)	32	R/W	Undefined	47.4.22/ 1877
9B0	Rx Individual Mask Registers (CAN_RXIMR76)	32	R/W	Undefined	47.4.22/ 1877
9B4	Rx Individual Mask Registers (CAN_RXIMR77)	32	R/W	Undefined	47.4.22/ 1877
9B8	Rx Individual Mask Registers (CAN_RXIMR78)	32	R/W	Undefined	47.4.22/ 1877
9BC	Rx Individual Mask Registers (CAN_RXIMR79)	32	R/W	Undefined	47.4.22/ 1877

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9C0	Rx Individual Mask Registers (CAN_RXIMR80)	32	R/W	Undefined	47.4.22/ 1877
9C4	Rx Individual Mask Registers (CAN_RXIMR81)	32	R/W	Undefined	47.4.22/ 1877
9C8	Rx Individual Mask Registers (CAN_RXIMR82)	32	R/W	Undefined	47.4.22/ 1877
9CC	Rx Individual Mask Registers (CAN_RXIMR83)	32	R/W	Undefined	47.4.22/ 1877
9D0	Rx Individual Mask Registers (CAN_RXIMR84)	32	R/W	Undefined	47.4.22/ 1877
9D4	Rx Individual Mask Registers (CAN_RXIMR85)	32	R/W	Undefined	47.4.22/ 1877
9D8	Rx Individual Mask Registers (CAN_RXIMR86)	32	R/W	Undefined	47.4.22/ 1877
9DC	Rx Individual Mask Registers (CAN_RXIMR87)	32	R/W	Undefined	47.4.22/ 1877
9E0	Rx Individual Mask Registers (CAN_RXIMR88)	32	R/W	Undefined	47.4.22/ 1877
9E4	Rx Individual Mask Registers (CAN_RXIMR89)	32	R/W	Undefined	47.4.22/ 1877
9E8	Rx Individual Mask Registers (CAN_RXIMR90)	32	R/W	Undefined	47.4.22/ 1877
9EC	Rx Individual Mask Registers (CAN_RXIMR91)	32	R/W	Undefined	47.4.22/ 1877
9F0	Rx Individual Mask Registers (CAN_RXIMR92)	32	R/W	Undefined	47.4.22/ 1877
9F4	Rx Individual Mask Registers (CAN_RXIMR93)	32	R/W	Undefined	47.4.22/ 1877
9F8	Rx Individual Mask Registers (CAN_RXIMR94)	32	R/W	Undefined	47.4.22/ 1877
9FC	Rx Individual Mask Registers (CAN_RXIMR95)	32	R/W	Undefined	47.4.22/ 1877
AE0	Memory Error Control Register (CAN_MECR)	32	R/W	800C_0080h	47.4.23/ 1878
AE4	Error Injection Address Register (CAN_ERRIAR)	32	R/W	0000_0000h	47.4.24/ 1880
AE8	Error Injection Data Pattern Register (CAN_ERRIDPR)	32	R/W	0000_0000h	47.4.25/ 1881
AEC	Error Injection Parity Pattern Register (CAN_ERRIPPR)	32	R/W	0000_0000h	47.4.26/ 1881
AF0	Error Report Address Register (CAN_RERRAR)	32	R	0000_0000h	47.4.27/ 1882
AF4	Error Report Data Register (CAN_RERRDR)	32	R	0000_0000h	47.4.28/ 1884

Table continues on the next page...

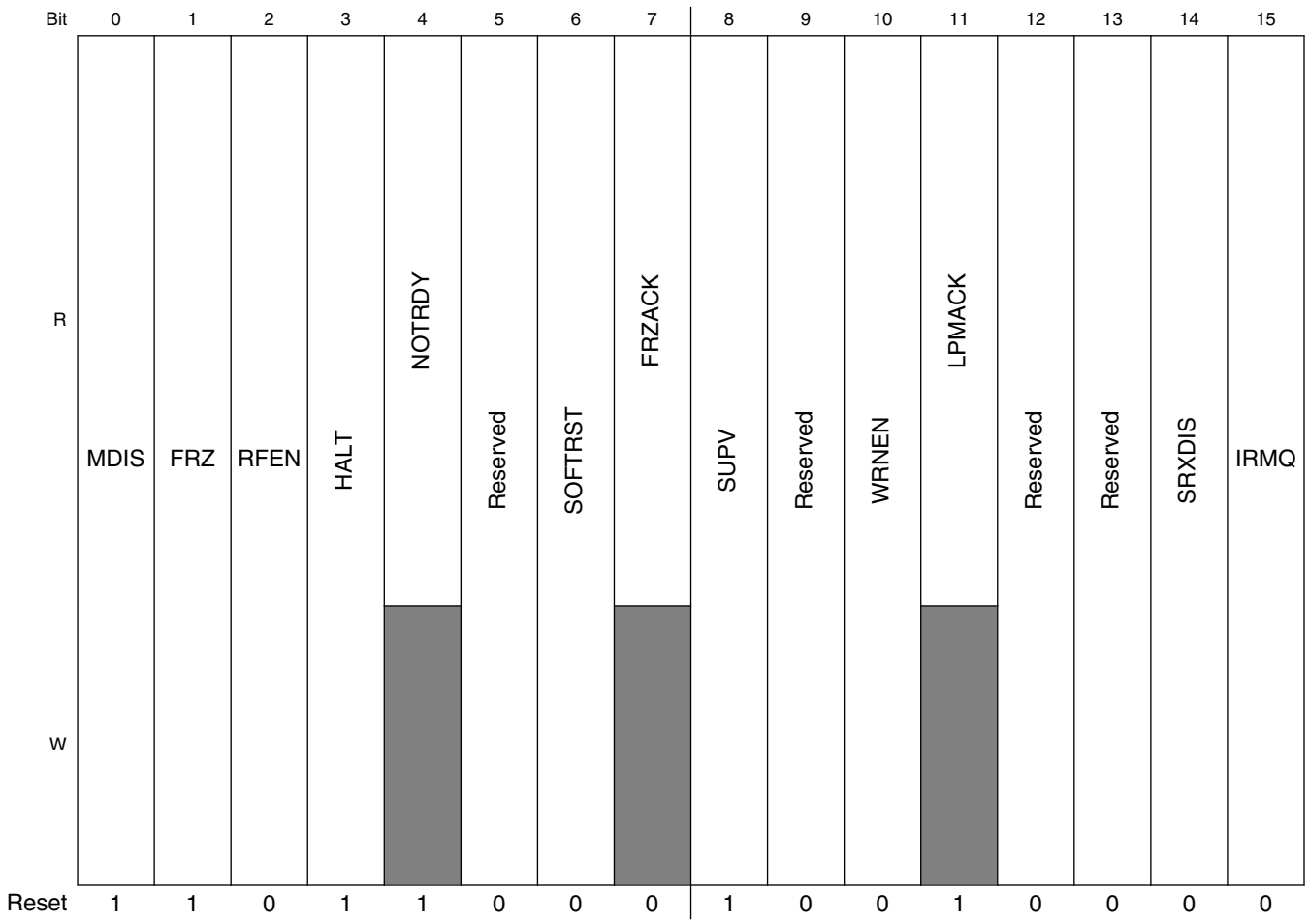
CAN memory map (continued)

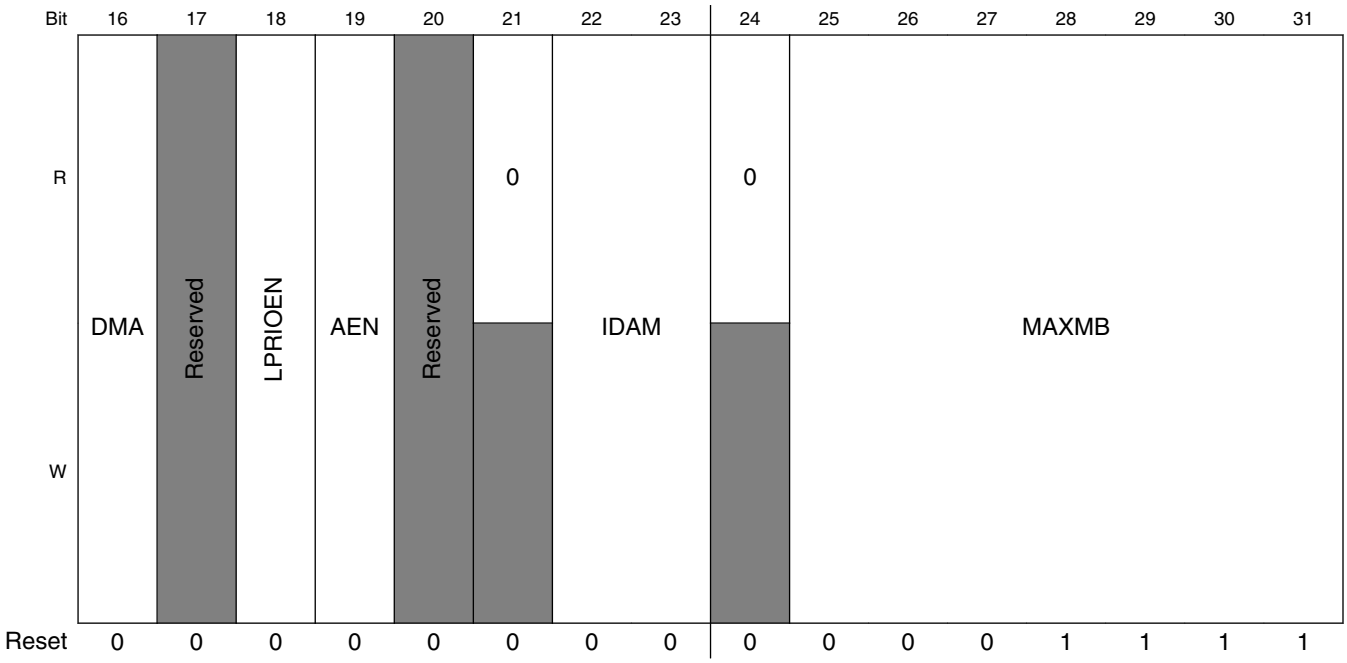
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
AF8	Error Report Syndrome Register (CAN_RERRSYNR)	32	R	0000_0000h	47.4.29/ 1884
AFC	Error Status Register (CAN_ERRSR)	32	R/W	0000_0000h	47.4.30/ 1887

47.4.2 Module Configuration Register (CAN_MCR)

This register defines global system configurations, such as the module operation modes and the maximum message buffer configuration.

Address: 0h base + 0h offset = 0h





CAN_MCR field descriptions

Field	Description
0 MDIS	Module Disable This bit controls whether FlexCAN is enabled or not. When disabled, FlexCAN disables the clocks to the CAN Protocol Engine and Controller Host Interface sub-modules. This bit is not affected by soft reset. 0 Enable the FlexCAN module. 1 Disable the FlexCAN module.
1 FRZ	Freeze Enable The FRZ bit specifies the FlexCAN behavior when the HALT bit in the CAN_MCR Register is set or when Debug mode is requested at chip level. When FRZ is asserted, FlexCAN is enabled to enter Freeze mode. Negation of this bit field causes FlexCAN to exit from Freeze mode. This bit is set by hardware when a non-correctable error is detected and NCEFAFRZ bit in CAN_MECR register is asserted. 0 Not enabled to enter Freeze mode. 1 Enabled to enter Freeze mode.
2 RFEN	Rx FIFO Enable This bit controls whether the Rx FIFO feature is enabled or not. When RFEN is set, MBs 0 to 5 cannot be used for normal reception and transmission because the corresponding memory region (0x80-0xDC) is used by the FIFO engine as well as additional MBs (up to 32, depending on CAN_CTRL2[RFFN] setting) which are used as Rx FIFO ID Filter Table elements. RFEN also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in the table "Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate" (see Arbitration and matching timing). This bit can be written in Freeze mode only because it is blocked by hardware in other modes. 0 Rx FIFO not enabled. 1 Rx FIFO enabled.
3 HALT	Halt FlexCAN

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
	Assertion of this bit puts the FlexCAN module into Freeze mode. The CPU should clear it after initializing the Message Buffers and the Control Registers CAN_CTRL1 and CAN_CTRL2. No reception or transmission is performed by FlexCAN before this bit is cleared. Freeze mode cannot be entered while FlexCAN is in a low power mode. The HALT bit is set by hardware when a non-correctable error is detected and NCEFAFRZ bit in CAN_MECR register is asserted. 0 No Freeze mode request. 1 Enters Freeze mode if the FRZ bit is asserted.
4 NOTRDY	FlexCAN Not Ready This read-only bit indicates that FlexCAN is either in Disable mode, Stop mode or Freeze mode. It is negated once FlexCAN has exited these modes. This bit is not affected by soft reset. 0 FlexCAN module is either in Normal mode, Listen-Only mode or Loop-Back mode. 1 FlexCAN module is either in Disable mode, Stop mode or Freeze mode.
5 Reserved	This field is reserved. When writing to this field, always write the reset value.
6 SOFTRST	Soft Reset When this bit is asserted, FlexCAN resets its internal state machines and some of the memory mapped registers. The SOFTRST bit can be asserted directly by the CPU when it writes to the MCR Register. Because soft reset is synchronous and has to follow a request/acknowledge procedure across clock domains, it may take some time to fully propagate its effect. The SOFTRST bit remains asserted while reset is pending, and is automatically negated when reset completes. Therefore, software can poll this bit to know when the soft reset has completed. Soft reset cannot be applied while clocks are shut down in a low power mode. The module should be first removed from low power mode, and then soft reset can be applied. This bit is not affected by soft reset. 0 No reset request. 1 Resets the registers affected by soft reset.
7 FRZACK	Freeze Mode Acknowledge This read-only bit indicates that FlexCAN is in Freeze mode and its prescaler is stopped. The Freeze mode request cannot be granted until current transmission or reception processes have finished. Therefore the software can poll the FRZACK bit to know when FlexCAN has actually entered Freeze mode. If Freeze Mode request is negated, then this bit is negated after the FlexCAN prescaler is running again. If Freeze mode is requested while FlexCAN is in a low power mode, then the FRZACK bit will be set only when the low-power mode is exited. See Section "Freeze Mode". This bit is not affected by soft reset. **NOTE:** FRZACK will be asserted within 178 CAN bits from the freeze mode request by the CPU, and negated within 2 CAN bits after the freeze mode request removal (see Section "Protocol Timing"). 0 FlexCAN not in Freeze mode, prescaler running. 1 FlexCAN in Freeze mode, prescaler stopped.
8 SUPV	Supervisor Mode This bit configures the FlexCAN to be either in Supervisor or User mode. The registers affected by this bit are marked as S/U in the Access Type column of the module memory map. Reset value of this bit is 1, so the affected registers start with Supervisor access allowance only. This bit can be written only in Freeze mode because it is blocked by hardware in other modes.

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
	0 FlexCAN is in User mode. Affected registers allow both Supervisor and Unrestricted accesses. 1 FlexCAN is in Supervisor mode. Affected registers allow only Supervisor access. Unrestricted access behaves as though the access was done to an unimplemented register location.
9 Reserved	This field is reserved. When writing to this field, always write the reset value.
10 WRNEN	Warning Interrupt Enable When asserted, this bit enables the generation of the TWRNINT and RWRNINT flags in the Error and Status Register 1 (ESR1). If WRNEN is negated, the TWRNINT and RWRNINT flags will always be zero, independent of the values of the error counters, and no warning interrupt will ever be generated. This bit can be written in Freeze mode only because it is blocked by hardware in other modes. 0 TWRNINT and RWRNINT bits are zero, independent of the values in the error counters. 1 TWRNINT and RWRNINT bits are set when the respective error counter transitions from less than 96 to greater than or equal to 96.
11 LPMACK	Low-Power Mode Acknowledge This read-only bit indicates that FlexCAN is in a low-power mode (Disable mode, Stop mode). A low-power mode cannot be entered until all current transmission or reception processes have finished, so the CPU can poll the LPMACK bit to know when FlexCAN has actually entered low power mode. This bit is not affected by soft reset. **NOTE:** LPMACK will be asserted within 180 CAN bits from the low-power mode request by the CPU, and negated within 2 CAN bits after the low-power mode request removal (see Section "Protocol Timing"). 0 FlexCAN is not in a low-power mode. 1 FlexCAN is in a low-power mode.
12 Reserved	This field is reserved. When writing to this field, always write the reset value.
13 Reserved	This field is reserved. When writing to this field, always write the reset value.
14 SRXDIS	Self Reception Disable This bit defines whether FlexCAN is allowed to receive frames transmitted by itself. If this bit is asserted, frames transmitted by the module will not be stored in any MB, regardless if the MB is programmed with an ID that matches the transmitted frame, and no interrupt flag or interrupt signal will be generated due to the frame reception. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Self reception enabled. 1 Self reception disabled.
15 IRMQ	Individual Rx Masking And Queue Enable This bit indicates whether Rx matching process will be based either on individual masking and queue or on masking scheme with CAN_RXMGMASK, CAN_RX14MASK, CAN_RX15MASK and CAN_RXFGMASK. This bit can be written in Freeze mode only because it is blocked by hardware in other modes. 0 Individual Rx masking and queue feature are disabled. For backward compatibility with legacy applications, the reading of C/S word locks the MB even if it is EMPTY. 1 Individual Rx masking and queue feature are enabled.

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
16 DMA	DMA Enable The DMA Enable bit controls whether the DMA feature is enabled or not. The DMA feature can only be used in Rx FIFO, consequently the bit CAN_MCR[RFEN] must be asserted. When DMA and RFEN are set, the CAN_IFLAG1[BUF5I] generates the DMA request and no RX FIFO interrupt is generated. This bit can be written in Freeze mode only as it is blocked by hardware in other modes. 0 DMA feature for RX FIFO disabled. 1 DMA feature for RX FIFO enabled.
17 Reserved	This field is reserved.
18 LPRIOEN	Local Priority Enable This bit is provided for backwards compatibility with legacy applications. It controls whether the local priority feature is enabled or not. It is used to expand the ID used during the arbitration process. With this expanded ID concept, the arbitration process is done based on the full 32-bit word, but the actual transmitted ID still has 11-bit for standard frames and 29-bit for extended frames. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Local Priority disabled. 1 Local Priority enabled.
19 AEN	Abort Enable When asserted, this bit enables the Tx abort mechanism. This mechanism guarantees a safe procedure for aborting a pending transmission, so that no frame is sent in the CAN bus without notification. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. **NOTE:** When CAN_MCR[AEN] is asserted, only the abort mechanism (see Transmission abort mechanism) must be used for updating Mailboxes configured for transmission. **CAUTION:** Writing the Abort code into Rx Mailboxes can cause unpredictable results when the CAN_MCR[AEN] is asserted. 0 Abort disabled. 1 Abort enabled.
20 Reserved	This field is reserved.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 IDAM	ID Acceptance Mode This 2-bit field identifies the format of the Rx FIFO ID Filter Table elements. Note that all elements of the table are configured at the same time by this field (they are all the same format). See Section "Rx FIFO Structure". This field can be written only in Freeze mode because it is blocked by hardware in other modes. 00 Format A: One full ID (standard and extended) per ID Filter Table element. 01 Format B: Two full standard IDs or two partial 14-bit (standard and extended) IDs per ID Filter Table element. 10 Format C: Four partial 8-bit Standard IDs per ID Filter Table element. 11 Format D: All frames rejected.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
25–31 MAXMB	Number Of The Last Message Buffer This 7-bit field defines the number of the last Message Buffers that will take part in the matching and arbitration processes. The reset value (0x0F) is equivalent to a 16 MB configuration. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Number of the last MB = MAXMB **NOTE:** MAXMB must be programmed with a value smaller than or equal to the number of available Message Buffers. Additionally, the definition of MAXMB value must take into account the region of MBs occupied by Rx FIFO and its ID filters table space defined by RFFN bit in CAN_CTRL2 register. MAXMB also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in Table "Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate" (see Arbitration and matching timing).

47.4.3 Control 1 register (CAN_CTRL1)

This register is defined for specific FlexCAN control features related to the CAN bus, such as bit-rate, programmable sampling point within an Rx bit, Loop Back mode, Listen-Only mode, Bus Off recovery behavior and interrupt enabling (Bus-Off, Error, Warning). It also determines the Division Factor for the clock prescaler.

The CAN bit timing variables (PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW) can also be configured in CAN_CBT register, which extends the range of all these variables. If CAN_CBT[BTF] is set, PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW fields of CAN_CTRL1 become read only.

The contents of this register are not affected by soft reset.

NOTE

The CAN bit variables in CAN_CTRL1 and in CAN_CBT are stored in the same register.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PRESDIV								RJW		PSEG1			PSEG2		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BOFFMSK	ERRMSK	Reserved	LPB	TWRNMSK	RWRNMSK	Reserved	Reserved	SMP	BOFFREC	TSYN	LBUF	LOM	PROPSEG		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_CTRL1 field descriptions

Field	Description
0–7 PRESDIV	Prescaler Division Factor This 8-bit field defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency. The Maximum value of this field is 0xFF, that gives a minimum Sclock frequency equal to the PE clock frequency divided by 256. See Section "Protocol Timing". This field can be written only in Freeze mode because it is blocked by hardware in other modes. Sclock frequency = PE clock frequency / (PRESDIV + 1)
8–9 RJW	Resync Jump Width This 2-bit field defines the maximum number of time quanta that a bit time can be changed by one re-synchronization. One time quantum is equal to the Sclock period. The valid programmable values are 0–3. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Resync Jump Width = RJW + 1.
10–12 PSEG1	Phase Segment 1 This 3-bit field defines the length of Phase Segment 1 in the bit time. The valid programmable values are 0–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 1 = (PSEG1 + 1) × Time-Quanta.
13–15 PSEG2	Phase Segment 2 This 3-bit field defines the length of Phase Segment 2 in the bit time. The valid programmable values are 1–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 2 = (PSEG2 + 1) × Time-Quanta.
16 BOFFMSK	Bus Off Interrupt Mask This bit provides a mask for the Bus Off Interrupt BOFFINT in CAN_ESR1 register. 0 Bus Off interrupt disabled. 1 Bus Off interrupt enabled.
17 ERRMSK	Error Interrupt Mask This bit provides a mask for the Error Interrupt ERRINT in the CAN_ESR1 register. 0 Error interrupt disabled. 1 Error interrupt enabled.
18 Reserved	This field is reserved.

Table continues on the next page...

CAN_CTRL1 field descriptions (continued)

Field	Description
19 LPB	Loop Back Mode This bit configures FlexCAN to operate in Loop-Back mode. In this mode, FlexCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic 1). FlexCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. **NOTE:** In this mode, the CAN_MCR[SRXDIS] cannot be asserted because this will impede the self reception of a transmitted message. 0 Loop Back disabled. 1 Loop Back enabled.
20 TWRNMSK	Tx Warning Interrupt Mask This bit provides a mask for the Tx Warning Interrupt associated with the TWRNINT flag in the Error and Status Register 1 (ESR1). This bit is read as zero when CAN_MCR[WRNEN] bit is negated. This bit can be written only if CAN_MCR[WRNEN] bit is asserted. 0 Tx Warning Interrupt disabled. 1 Tx Warning Interrupt enabled.
21 RWRNMSK	Rx Warning Interrupt Mask This bit provides a mask for the Rx Warning Interrupt associated with the RWRNINT flag in the Error and Status Register 1 (ESR1). This bit is read as zero when CAN_MCR[WRNEN] bit is negated. This bit can be written only if CAN_MCR[WRNEN] bit is asserted. 0 Rx Warning Interrupt disabled. 1 Rx Warning Interrupt enabled.
22 Reserved	This field is reserved.
23 Reserved	This field is reserved.
24 SMP	CAN Bit Sampling This bit defines the sampling mode of CAN bits at the Rx input. It can be written in Freeze mode only because it is blocked by hardware in other modes. **NOTE:** For proper operation, to assert SMP it is necessary to guarantee a minimum value of 2 TQs in CAN_CTRL1[PSEG1] (or CAN_CBT[EPSEG1]). 0 Just one sample is used to determine the bit value. 1 Three samples are used to determine the value of the received bit: the regular one (sample point) and 2 preceding samples; a majority rule is used.
25 BOFFREC	Bus Off Recovery This bit defines how FlexCAN recovers from Bus Off state. If this bit is negated, automatic recovering from Bus Off state occurs according to the CAN Specification 2.0B. If the bit is asserted, automatic recovering from Bus Off is disabled and the module remains in Bus Off state until the bit is negated by the user. If the

Table continues on the next page...

CAN_CTRL1 field descriptions (continued)

Field	Description
	negation occurs before 128 sequences of 11 recessive bits are detected on the CAN bus, then Bus Off recovery happens as if the BOFFREC bit had never been asserted. If the negation occurs after 128 sequences of 11 recessive bits occurred, then FlexCAN will re-synchronize to the bus by waiting for 11 recessive bits before joining the bus. After negation, the BOFFREC bit can be re-asserted again during Bus Off, but it will be effective only the next time the module enters Bus Off. If BOFFREC was negated when the module entered Bus Off, asserting it during Bus Off will not be effective for the current Bus Off recovery. 0 Automatic recovering from Bus Off state enabled. 1 Automatic recovering from Bus Off state disabled.
26 TSYN	Timer Sync This bit enables a mechanism that resets the free-running timer each time a message is received in Message Buffer 0. This feature provides means to synchronize multiple FlexCAN stations with a special “SYNC” message, that is, global network time. If the RFEN bit in CAN_MCR is set (Rx FIFO enabled), the first available Mailbox, according to CAN_CTRL2[RFFN] setting, is used for timer synchronization instead of MB0. This bit can be written in Freeze mode only because it is blocked by hardware in other modes. 0 Timer Sync feature disabled 1 Timer Sync feature enabled
27 LBUF	Lowest Buffer Transmitted First This bit defines the ordering mechanism for Message Buffer transmission. When asserted, the CAN_MCR[LPRIOEN] bit does not affect the priority arbitration. This bit can be written in Freeze mode only because it is blocked by hardware in other modes. 0 Buffer with highest priority is transmitted first. 1 Lowest number buffer is transmitted first.
28 LOM	Listen-Only Mode This bit configures FlexCAN to operate in Listen-Only mode. In this mode, transmission is disabled, all error counters described in CAN_ECR register are frozen and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error without changing the receive error counter (RXERRCNT) in CAN_ECR register, as if it was trying to acknowledge the message. Listen-Only mode is acknowledged by the state of CAN_ESR1[FLTCONF] field indicating Passive Error. There can be some delay between the Listen-Only mode request and acknowledge. This bit can be written in Freeze mode only because it is blocked by hardware in other modes. 0 Listen-Only mode is deactivated. 1 FlexCAN module operates in Listen-Only mode.
29–31 PROPSEG	Propagation Segment This 3-bit field defines the length of the Propagation Segment in the bit time. The valid programmable values are 0–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Propagation Segment Time = (PROPSEG + 1) × Time-Quanta. Time-Quantum = one Sclock period.

47.4.4 Free Running Timer (CAN_TIMER)

This register represents a 16-bit free running counter that can be read and written by the CPU. The timer starts from 0x0 after Reset, counts linearly to 0xFFFF, and wraps around.

The timer is incremented by the CAN bit clock, which defines the baud rate on the CAN bus. During a message transmission/reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate. The timer is not incremented during Disable, Stop and Freeze modes.

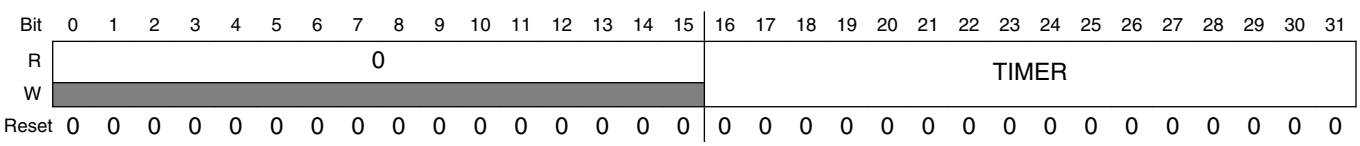
The timer value is captured when the second bit of the identifier field of any frame is on the CAN bus. This captured value is written into the Time Stamp entry in a message buffer after a successful reception or transmission of a message.

If bit CAN_CTRL1[TSYN] is asserted, the Timer is reset whenever a message is received in the first available Mailbox, according to CAN_CTRL2[RFFN] setting.

The CPU can write to this register anytime. However, if the write occurs at the same time that the Timer is being reset by a reception in the first Mailbox, then the write value is discarded.

Reading this register affects the Mailbox Unlocking procedure, see Section "Mailbox Lock Mechanism".

Address: 0h base + 8h offset = 8h



CAN_TIMER field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TIMER	Timer Value Contains the free-running counter value.

47.4.5 Rx Mailboxes Global Mask Register (CAN_RXMGMASK)

This register is located in RAM.

RXMGMASK is provided for legacy application support.

- When the CAN_MCR[IRMQ] bit is negated, RXMGMASK is always in effect (the bits in the MG field will mask the Mailbox filter bits).
- When the CAN_MCR[IRMQ] bit is asserted, RXMGMASK has no effect (the bits in the MG field will not mask the Mailbox filter bits).

RXMGMASK is used to mask the filter fields of all Rx MBs, excluding MBs 14-15, which have individual mask registers.

This register can only be written in Freeze mode as it is blocked by hardware in other modes.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MG																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

CAN_RXMGMASK field descriptions

Field	Description
0–31 MG	Rx Mailboxes Global Mask Bits These bits mask the Mailbox filter bits. Note that the alignment with the ID word of the Mailbox is not perfect as the two most significant MG bits affect the fields RTR and IDE, which are located in the Control and Status word of the Mailbox. The following table shows in detail which MG bits mask each Mailbox filter field.

SMB[RTR] [1]	CAN_CTRL2[RRS]	CAN_CTRL2[EACEN]	Mailbox filter fields			
			MB[RTR]	MB[IDE]	MB[ID]	Reserved
0	-	0	note [2]	note [3]	MG[28:0]	MG[31:29]
0	-	1	MG[31]	MG[30]	MG[28:0]	MG[29]
1	0	-	-	-	-	MG[31:0]
1	1	0	-	-	MG[28:0]	MG[31:29]
1	1	1	MG[31]	MG[30]	MG[28:0]	MG[29]

1. RTR bit of the Incoming Frame. It is saved into an auxiliary MB called Rx Serial Message Buffer (Rx SMB).
2. If the CTRL2[EACEN] bit is negated, the RTR bit of Mailbox is never compared with the RTR bit of the incoming frame.
3. If the CAN_CTRL2[EACEN] bit is negated, the IDE bit of Mailbox is always compared with the IDE bit of the incoming frame.

0 The corresponding bit in the filter is "don't care."
1 The corresponding bit in the filter is checked.

1. RTR bit of the Incoming Frame. It is saved into an auxiliary MB called Rx Serial Message Buffer (Rx SMB).
2. If the CTRL2[EACEN] bit is negated, the RTR bit of Mailbox is never compared with the RTR bit of the incoming frame.
3. If the CAN_CTRL2[EACEN] bit is negated, the IDE bit of Mailbox is always compared with the IDE bit of the incoming frame.

47.4.6 Rx 14 Mask register (CAN_RX14MASK)

This register is located in RAM.

RX14MASK is provided for legacy application support. When the CAN_MCR[IRMQ] bit is asserted, RX14MASK has no effect.

RX14MASK is used to mask the filter fields of Message Buffer 14.

This register can only be programmed while the module is in Freeze mode as it is blocked by hardware in other modes.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RX14M																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

CAN_RX14MASK field descriptions

Field	Description
0–31 RX14M	Rx Buffer 14 Mask Bits Each mask bit masks the corresponding Mailbox 14 filter field in the same way that RXMGMASK masks other Mailboxes' filters. See the description of the CAN_RXMGMASK register. 0 The corresponding bit in the filter is "don't care." 1 The corresponding bit in the filter is checked.

47.4.7 Rx 15 Mask register (CAN_RX15MASK)

This register is located in RAM.

RX15MASK is provided for legacy application support. When the CAN_MCR[IRMQ] bit is asserted, RX15MASK has no effect.

RX15MASK is used to mask the filter fields of Message Buffer 15.

This register can be programmed only while the module is in Freeze mode because it is blocked by hardware in other modes.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RX15M																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

CAN_RX15MASK field descriptions

Field	Description
0–31 RX15M	Rx Buffer 15 Mask Bits Each mask bit masks the corresponding Mailbox 15 filter field in the same way that RXMGMASK masks other Mailboxes' filters. See the description of the CAN_RXMGMASK register. 0 The corresponding bit in the filter is "don't care." 1 The corresponding bit in the filter is checked.

47.4.8 Error Counter (CAN_ECR)

This register has two 8-bit fields reflecting the value of the FlexCAN error counters:

- Transmit Error Counter (TXERRCNT field)
- Receive Error Counter (RXERRCNT field)

The Fault Confinement State (FLTCONF field in Error and Status Register 1 - CAN_ESR1) is updated based on TXERRCNT and RXERRCNT counters. TXERRCNT and RXERRCNT counters can be written in Freeze mode only. The rules for increasing and decreasing these counters are described in the CAN protocol and are completely implemented in the FlexCAN module.

The following are the basic rules for FlexCAN bus state transitions:

- If the value of TXERRCNT or RXERRCNT increases to be greater than or equal to 128, the FLTCONF field in the Error and Status Register is updated to reflect "Error Passive" state.
- If the FlexCAN state is "Error Passive", and either TXERRCNT or RXERRCNT decrements to a value less than or equal to 127 while the other already satisfies this condition, the FLTCONF field in the Error and Status Register is updated to reflect "Error Active" state.
- If the value of TXERRCNT increases to be greater than 255, the FLTCONF field in the Error and Status Register is updated to reflect "Bus Off" state, and an interrupt may be issued. The value of TXERRCNT is then reset to zero.

- If FlexCAN is in "Bus Off' state, then TXERRCNT is cascaded together with another internal counter to count the 128th occurrences of 11 consecutive recessive bits on the bus. Hence, TXERRCNT is reset to zero and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing the TXERRCNT. When TXERRCNT reaches the value of 128, the FLTCONF field in the Error and Status Register is updated to be "Error Active" and both error counters are reset to zero. At any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the TXERRCNT value.
- If during system start-up, only one node is operating, then its TXERRCNT increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the ACKERR bit in the Error and Status Register). After the transition to "Error Passive" state, the TXERRCNT does not increment anymore by acknowledge errors. Therefore the device never goes to the "Bus Off" state.
- If the RXERRCNT increases to a value greater than 127, it is not incremented further, even if more errors are detected while being a receiver. At the next successful message reception, the counter is set to a value between 119 and 127 to resume to "Error Active" state.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved								Reserved								RXERRCNT								TXERRCNT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ECR field descriptions

Field	Description
0–7 Reserved	This field is reserved.
8–15 Reserved	This field is reserved.
16–23 RXERRCNT	Receive Error Counter Receive Error Counter for all errors detected in received messages. The RXERRCNT counter is read-only except in Freeze mode, where it can be written by the CPU.
24–31 TXERRCNT	Transmit Error Counter Transmit Error Counter for all errors detected in transmitted messages. The TXERRCNT counter is read-only except in Freeze mode, where it can be written by the CPU.

47.4.9 Error and Status 1 register (CAN_ESR1)

This register reports various error conditions detected in the reception and transmission of a CAN frame, some general status of the device and it is the source of some interrupts to the CPU.

The reported error conditions are BIT1ERR, BIT0ERR, ACKKERR, CRCERR, FRMERR and STFERR.

An error detected in a single CAN frame may be reported by one or more error flags. Also, error reporting is cummulative in case more error events happen in the next frames while the CPU does not attempt to read this register.

TXWRN, RXWRN, IDLE, TX, FLTCONF, RX and SYNCH are status bits.

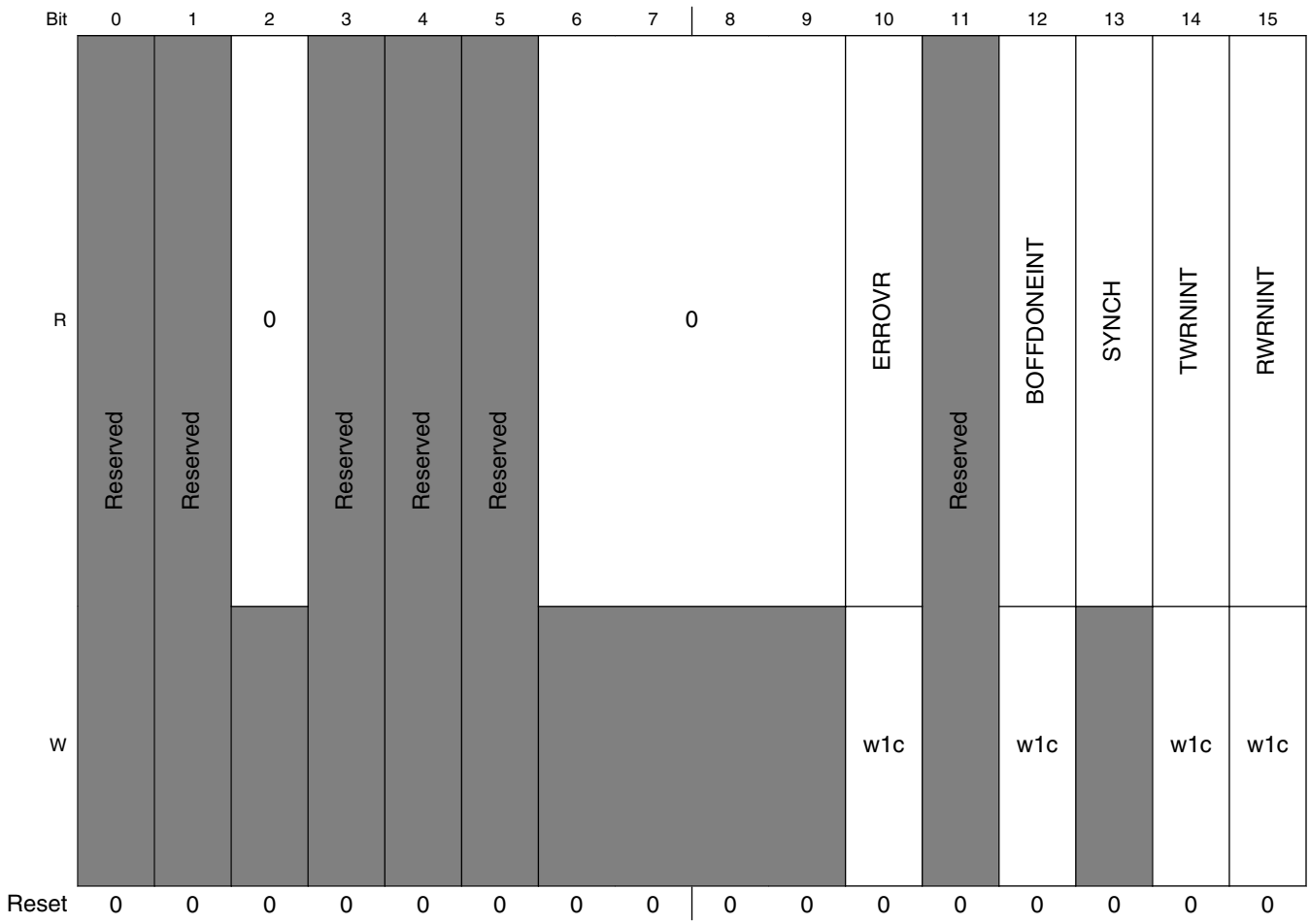
BOFFINT, BOFFDONEINT, ERRINT, TWRNINT and RWRNINT are interrupt bits. It is recommended the CPU to use the following procedure when servicing interrupt requests generated by these bits:

- Read this register to capture all error condition and status bits. This action clear the respective bits that were set since the last read access.
- Write 1 to clear the interrupt bit that has triggered the interrupt request.
- Write 1 to clear the ERR_OVR bit if it is set.

Starting from all error flags cleared, a first error event sets the ERRINT (provided the corresponding mask bit is asserted). If other error events in subsequent frames happen before the CPU to serve the interrupt request, the ERR_OVR bit is set to indicate that errors from different frames had accumulated.

SYNCH	IDLE	TX	RX	FlexCAN State
0	0	0	0	Not synchronized to CAN bus
1	1	x	x	Idle
1	0	1	0	Transmitting
1	0	0	1	Receiving

Address: 0h base + 20h offset = 20h



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BIT1ERR	BIT0ERR	ACKERR	CRCERR	FRMERR	STFERR	TXWRN	RXWRN	IDLE	TX	FLTCONF		RX	BOFFINT	ERRINT	0
W														w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ESR1 field descriptions

Field	Description
0 Reserved	This field is reserved.
1 Reserved	This field is reserved.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 Reserved	This field is reserved.
4 Reserved	This field is reserved.
5 Reserved	This field is reserved.
6–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 ERROVR	Error Overrun bit This bit indicates that an error condition occurred when any error flag is already set. This bit is cleared by writing it to 1.

Table continues on the next page...

CAN_ESR1 field descriptions (continued)

Field	Description
	0 Overrun has not occurred. 1 Overrun has occured.
11 Reserved	This field is reserved.
12 BOFFDONEINT	Bus Off Done Interrupt This bit is set when the Tx Error Counter (TXERRCNT) has finished counting 128 occurrences of 11 consecutive recessive bits on the CAN bus and is ready to leave Bus Off. If the corresponding mask bit in the Control 2 Register (BOFFDONEMSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect. 0 No such occurrence. 1 FlexCAN module has completed Bus Off process.
13 SYNCH	CAN Synchronization Status This read-only flag indicates whether the FlexCAN is synchronized to the CAN bus and able to participate in the communication process. It is set and cleared by the FlexCAN. See the table in the overall CAN_ESR1 register description. 0 FlexCAN is not synchronized to the CAN bus. 1 FlexCAN is synchronized to the CAN bus.
14 TWRNINT	Tx Warning Interrupt Flag If the WRNEN bit in CAN_MCR is asserted, the TWRNINT bit is set when the TXWRN flag transitions from 0 to 1, meaning that the Tx error counter reached 96. If the corresponding mask bit in the Control 1 Register (CAN_CTRL1[TWRNMSK]) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. When WRNEN is negated, this flag is masked. CPU must clear this flag before disabling the bit. Otherwise it will be set when the WRNEN is set again. Writing 0 has no effect. This flag is not generated during Bus Off state. This bit is not updated during Freeze mode. 0 No such occurrence. 1 The Tx error counter transitioned from less than 96 to greater than or equal to 96.
15 RWRNINT	Rx Warning Interrupt Flag If the WRNEN bit in CAN_MCR is asserted, the RWRNINT bit is set when the RXWRN flag transitions from 0 to 1, meaning that the Rx error counters reached 96. If the corresponding mask bit in the Control 1 Register (CAN_CTRL1[RWRNMSK]) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. When WRNEN is negated, this flag is masked. CPU must clear this flag before disabling the bit. Otherwise it will be set when the WRNEN is set again. Writing 0 has no effect. This bit is not updated during Freeze mode. 0 No such occurrence. 1 The Rx error counter transitioned from less than 96 to greater than or equal to 96.
16 BIT1ERR	Bit1 Error This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message. **NOTE:** This bit is not set by a transmitter in case of arbitration field or ACK slot, or in case of a node sending a passive error flag that detects dominant bits. 0 No such occurrence. 1 At least one bit sent as recessive is received as dominant.

Table continues on the next page...

CAN_ESR1 field descriptions (continued)

Field	Description
17 BIT0ERR	Bit0 Error This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message. 0 No such occurrence. 1 At least one bit sent as dominant is received as recessive.
18 ACKERR	Acknowledge Error This bit indicates that an Acknowledge Error has been detected by the transmitter node, that is, a dominant bit has not been detected during the ACK SLOT. 0 No such occurrence. 1 An ACK error occurred since last read of this register.
19 CRCERR	Cyclic Redundancy Check Error This bit indicates that a CRC Error has been detected by the receiver node, that is, the calculated CRC is different from the received. 0 No such occurrence. 1 A CRC error occurred since last read of this register.
20 FRMERR	Form Error This bit indicates that a Form Error has been detected by the receiver node, that is, a fixed-form bit field contains at least one illegal bit. 0 No such occurrence. 1 A Form Error occurred since last read of this register.
21 STFERR	Stuffing Error This bit indicates that a Stuffing Error has been detected by the receiver node. 0 No such occurrence. 1 A Stuffing Error occurred since last read of this register.
22 TXWRN	TX Error Warning This bit indicates when repetitive errors are occurring during message transmission and is affected by the value of TXERRCNT in CAN_ECR register only. This bit is not updated during Freeze mode. 0 No such occurrence. 1 TXERRCNT is greater than or equal to 96.
23 RXWRN	Rx Error Warning This bit indicates when repetitive errors are occurring during message reception and is affected by the value of RXERRCNT in CAN_ECR register only. This bit is not updated during Freeze mode. 0 No such occurrence. 1 RXERRCNT is greater than or equal to 96.
24 IDLE	This bit indicates when CAN bus is in IDLE state. See the table in the overall CAN_ESR1 register description. 0 No such occurrence. 1 CAN bus is now IDLE.

Table continues on the next page...

CAN_ESR1 field descriptions (continued)

Field	Description
25 TX	FlexCAN In Transmission This bit indicates if FlexCAN is transmitting a message. See the table in the overall CAN_ESR1 register description. 0 FlexCAN is not transmitting a message. 1 FlexCAN is transmitting a message.
26–27 FLTCONF	Fault Confinement State This 2-bit field indicates the Confinement State of the FlexCAN module. If the LOM bit in the Control Register 1 is asserted, after some delay that depends on the CAN bit timing the FLTCONF field will indicate “Error Passive”. The very same delay affects the way how FLTCONF reflects an update to CAN_ECR register by the CPU. It may be necessary up to one CAN bit time to get them coherent again. This bit field is affected by soft reset, but if the LOM bit is asserted, its reset value lasts just one CAN bit. After this time, FLTCONF reports "Error Passive". 00 Error Active 01 Error Passive 1x Bus Off
28 RX	FlexCAN In Reception This bit indicates if FlexCAN is receiving a message. See the table in the overall CAN_ESR1 register description. 0 FlexCAN is not receiving a message. 1 FlexCAN is receiving a message.
29 BOFFINT	Bus Off Interrupt This bit is set when FlexCAN enters ‘Bus Off’ state. If the corresponding mask bit in the Control Register 1 (CAN_CTRL1[BOFFMSK]) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect. 0 No such occurrence. 1 FlexCAN module entered Bus Off state.
30 ERRINT	Error Interrupt This bit indicates that at least one of the Error Bits (BIT1ERR, BIT0ERR, ACKERR. CRCERR, FRMERR or STFERR) is set. If the corresponding mask bit CAN_CTRL1[ERRMSK] is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect. 0 No such occurrence. 1 Indicates setting of any Error Bit in the Error and Status Register.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

47.4.10 Interrupt Masks 2 register (CAN_IMASK2)

This register allows any number of a range of the 32 Message Buffer Interrupts to be enabled or disabled for MB63 to MB32. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception, that is, when the corresponding CAN_IFLAG2 bit is set.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	BUF63TO32M																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IMASK2 field descriptions

Field	Description
0–31 BUF63TO32M	Buffer MB $_i$ Mask Each bit enables or disables the corresponding FlexCAN Message Buffer Interrupt for MB63 to MB32. **NOTE:** Setting or clearing a bit in the CAN_IMASK2 Register can assert or negate an interrupt request, if the corresponding IFLAG2 bit is set. 0 The corresponding buffer Interrupt is disabled. 1 The corresponding buffer Interrupt is enabled.

47.4.11 Interrupt Masks 1 register (CAN_IMASK1)

This register allows any number of a range of the 32 Message Buffer Interrupts to be enabled or disabled for MB31 to MB0. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception, that is, when the corresponding CAN_IFLAG1 bit is set.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	BUF31TO0M																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IMASK1 field descriptions

Field	Description
0–31 BUF31TO0M	Buffer MB $_i$ Mask Each bit enables or disables the corresponding FlexCAN Message Buffer Interrupt for MB31 to MB0. **NOTE:** Setting or clearing a bit in the CAN_IMASK1 Register can assert or negate an interrupt request, if the corresponding IFLAG1 bit is set. 0 The corresponding buffer Interrupt is disabled. 1 The corresponding buffer Interrupt is enabled.

47.4.12 Interrupt Flags 2 register (CAN_IFLAG2)

This register defines the flags for the 32 Message Buffer interrupts for MB63 to MB32. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the respective CAN_IFLAG2 bit. If the corresponding CAN_IMASK2 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect.

Before updating CAN_MCR[MAXMB] field, CPU must service the CAN_IFLAG2 bits whose MB value is greater than the MAXMB to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BUF63TO32I																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IFLAG2 field descriptions

Field	Description
0–31 BUF63TO32I	Buffer MB $_i$ Interrupt Each bit flags the corresponding FlexCAN Message Buffer interrupt for MB63 to MB32. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception. 1 The corresponding buffer has successfully completed transmission or reception.

47.4.13 Interrupt Flags 1 register (CAN_IFLAG1)

This register defines the flags for the 32 Message Buffer interrupts for MB31 to MB0. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding CAN_IFLAG1 bit. If the corresponding CAN_IMASK1 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect. There is an exception when DMA for Rx FIFO is enabled, as described below.

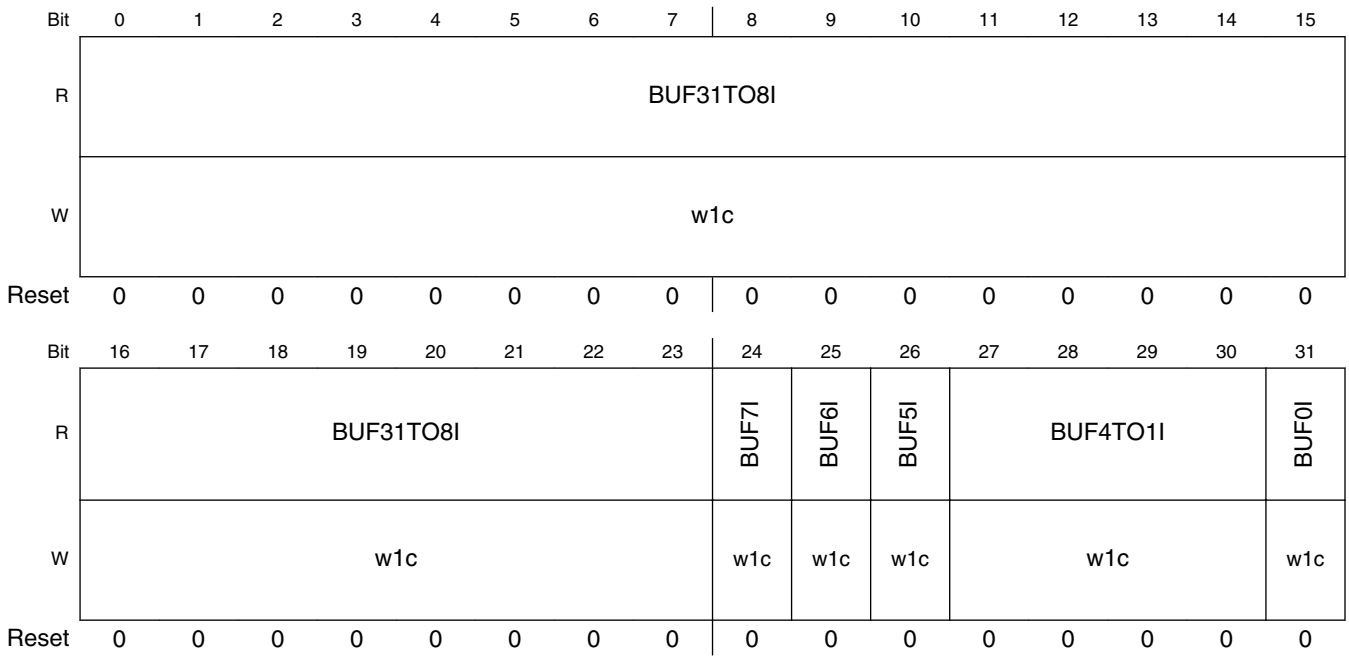
The BUF7I to BUF5I flags are also used to represent FIFO interrupts when the Rx FIFO is enabled. When the bit CAN_MCR[RFEN] is set and the bit CAN_MCR[DMA] is negated, the function of the 8 least significant interrupt flags changes: BUF7I, BUF6I and BUF5I indicate operating conditions of the FIFO, BUF0I is used to empty FIFO, and BUF4I to BUF1I bits are reserved.

Before enabling the CAN_MCR[RFEN], the CPU must service the IFLAG bits asserted in the Rx FIFO region; see Section "Rx FIFO". Otherwise, these IFLAG bits will mistakenly show the related MBs now belonging to FIFO as having contents to be serviced. When the CAN_MCR[RFEN] bit is negated, the FIFO flags must be cleared. The same care must be taken when an CAN_CTRL2[RFFN] value is selected extending Rx FIFO filters beyond MB7. For example, when RFFN is 0x8, the MB0-23 range is occupied by Rx FIFO filters and related IFLAG bits must be cleared.

When both the CAN_MCR[RFEN] and CAN_MCR[DMA] bits are asserted (DMA feature for Rx FIFO enabled), the function of the 8 least significant interrupt flags (BUF7I - BUF0I) are changed to support the DMA operation. BUF7I and BUF6I are not used, as well as, BUF4I to BUF1I. BUF5I indicates operating condition of FIFO, and BUF0I is used to empty FIFO. Moreover, BUF5I does not generate a CPU interrupt, but generates a DMA request. IMASK1 bits in Rx FIFO region are not considered when bit CAN_MCR[DMA] is enabled. In addition the CPU must not clear the flag BUF5I when DMA is enabled. Before enabling the bit CAN_MCR[DMA], the CPU must service the IFLAGs asserted in the Rx FIFO region. When the bit CAN_MCR[DMA] is negated, the FIFO must be empty.

Before updating CAN_MCR[MAXMB] field, CPU must service the CAN_IFLAG1 bits whose MB value is greater than the CAN_MCR[MAXMB] to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.

Address: 0h base + 30h offset = 30h



CAN_IFLAG1 field descriptions

Field	Description
0–23 BUF31TO8I	Buffer MB_i Interrupt Each bit flags the corresponding FlexCAN Message Buffer interrupt for MB31 to MB8. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception. 1 The corresponding buffer has successfully completed transmission or reception.
24 BUF7I	Buffer MB7 Interrupt Or "Rx FIFO Overflow" When the RFEN bit in the CAN_MCR register is cleared (Rx FIFO disabled), this bit flags the interrupt for MB7. **NOTE:** This flag is cleared by the FlexCAN whenever the bit CAN_MCR[RFEN] is changed by CPU writes. The BUF7I flag represents "Rx FIFO Overflow" when CAN_MCR[RFEN] is set. In this case, the flag indicates that a message was lost because the Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a Mailbox. 0 No occurrence of MB7 completing transmission/reception when MCR[RFEN]=0, or of Rx FIFO overflow when MCR[RFEN]=1 1 MB7 completed transmission/reception when MCR[RFEN]=0, or Rx FIFO overflow when MCR[RFEN]=1
25 BUF6I	Buffer MB6 Interrupt Or "Rx FIFO Warning" When the RFEN bit in the CAN_MCR register is cleared (Rx FIFO disabled), this bit flags the interrupt for MB6. **NOTE:** This flag is cleared by the FlexCAN whenever the bit CAN_MCR[RFEN] is changed by CPU writes.

Table continues on the next page...

CAN_IFLAG1 field descriptions (continued)

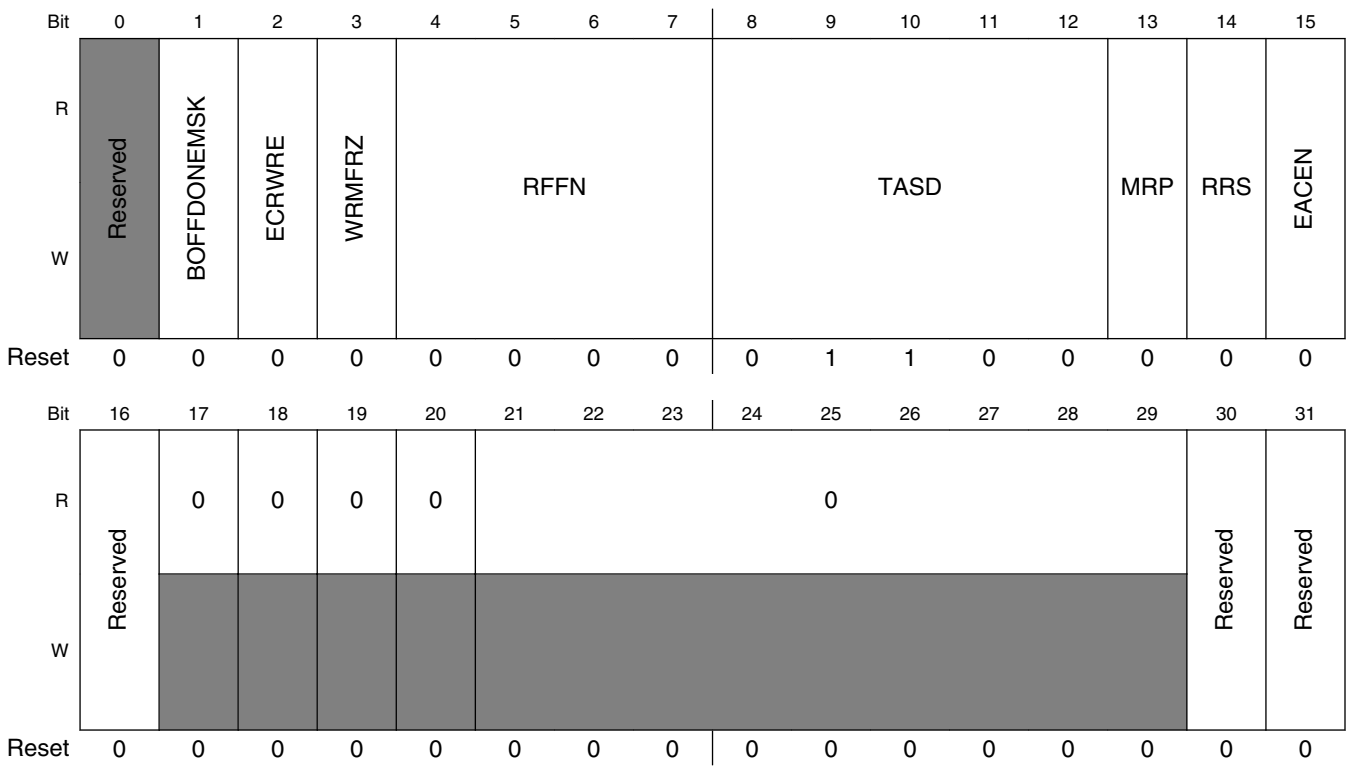
Field	Description
	The BUF6I flag represents "Rx FIFO Warning" when CAN_MCR[RFEN] is set. In this case, the flag indicates when the number of unread messages within the Rx FIFO is increased to 5 from 4 due to the reception of a new one, meaning that the Rx FIFO is almost full. Note that if the flag is cleared while the number of unread messages is greater than 4, it does not assert again until the number of unread messages within the Rx FIFO is decreased to be equal to or less than 4. 0 No occurrence of MB6 completing transmission/reception when MCR[RFEN]=0, or of Rx FIFO almost full when MCR[RFEN]=1 1 MB6 completed transmission/reception when MCR[RFEN]=0, or Rx FIFO almost full when MCR[RFEN]=1
26 BUF5I	Buffer MB5 Interrupt Or "Frames available in Rx FIFO" When the RFEN bit in the MCR is cleared (Rx FIFO disabled), this bit flags the interrupt for MB5. **NOTE:** This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by CPU writes. When MCR[RFEN] is set (Rx FIFO enabled), the BUF5I flag represents "Frames available in Rx FIFO" and indicates that at least one frame is available to be read from the Rx FIFO. When the MCR[DMA] bit is enabled, this flag generates a DMA request and the CPU must not clear this bit by writing 1 in BUF5I. 0 No occurrence of MB5 completing transmission/reception when MCR[RFEN]=0, or of frame(s) available in the FIFO, when MCR[RFEN]=1 1 MB5 completed transmission/reception when MCR[RFEN]=0, or frame(s) available in the Rx FIFO when MCR[RFEN]=1. It generates a DMA request in case of MCR[RFEN] and MCR[DMA] are enabled.
27–30 BUF4TO1I	Buffer MB $_i$ Interrupt Or "reserved" When the RFEN bit in the CAN_MCR register is cleared (Rx FIFO disabled), these bits flag the interrupts for MB4 to MB1. **NOTE:** These flags are cleared by the FlexCAN whenever the bit CAN_MCR[RFEN] is changed by CPU writes. The BUF4TO1I flags are reserved when CAN_MCR[RFEN] is set. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception when MCR[RFEN]=0. 1 The corresponding buffer has successfully completed transmission or reception when MCR[RFEN]=0.
31 BUF0I	Buffer MB0 Interrupt Or Clear FIFO bit When the RFEN bit in MCR is cleared (Rx FIFO disabled), this bit flags the interrupt for MB0. If the Rx FIFO is enabled, this bit is used to trigger the clear FIFO operation. This operation empties FIFO contents. Before performing this operation the CPU must service all FIFO related IFLAGs. When the bit MCR[DMA] is enabled this operation also clears the BUF5I flag and consequently abort the DMA request. The clear FIFO operation occurs when the CPU writes 1 in BUF0I. It is only allowed in Freeze Mode and is blocked by hardware in other conditions. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception when MCR[RFEN]=0. 1 The corresponding buffer has successfully completed transmission or reception when MCR[RFEN]=0.

47.4.14 Control 2 register (CAN_CTRL2)

This register complements Control1 Register providing control bits for memory write access in Freeze Mode, for extending FIFO filter quantity, and for adjust the operation of internal FlexCAN processes like matching and arbitration.

The contents of this register are not affected by soft reset.

Address: 0h base + 34h offset = 34h



CAN_CTRL2 field descriptions

Field	Description
0 Reserved	This field is reserved.
1 BOFFDONEMSK	Bus Off Done Interrupt Mask This bit provides a mask for the Bus Off Done Interrupt in CAN_ESR1 register. 0 Bus Off Done interrupt disabled. 1 Bus Off Done interrupt enabled.
2 ECRWRE	Error-correction Configuration Register Write Enable This bit enables the CAN_MECR register to be updated. This bit is automatically set to 0 if the protocol described in section Detection and Correction of Memory Errors is not followed.

Table continues on the next page...

CAN_CTRL2 field descriptions (continued)

Field	Description
	0 Disable update. 1 Enable update.
3 WRMFRZ	Write-Access To Memory In Freeze Mode Enable unrestricted write access to FlexCAN memory in Freeze mode. This bit can only be written in Freeze mode and has no effect out of Freeze mode. CAN_MCR[RFEN] bit must not be set during FlexCAN memory initialization. 0 Maintain the write access restrictions. 1 Enable unrestricted write access to FlexCAN memory.
4–7 RFFN	Number Of Rx FIFO Filters This 4-bit field defines the number of Rx FIFO filters, as shown in the following table. The maximum selectable number of filters is determined by the chip. This field can only be written in Freeze mode as it is blocked by hardware in other modes. This field must not be programmed with values that make the number of Message Buffers occupied by Rx FIFO and ID Filter exceed the number of Mailboxes present, defined by CAN_MCR[MAXMB]. **NOTE:** Each group of eight filters occupies a memory space equivalent to two Message Buffers which means that the more filters are implemented the less Mailboxes will be available. Considering that the Rx FIFO occupies the memory space originally reserved for MB0-5, RFFN should be programmed with a value corresponding to a number of filters not greater than the number of available memory words which can be calculated as follows: (SETUP_MB - 6) × 4 where SETUP_MB is the least between the parameter NUMBER_OF_MB and CAN_MCR[MAXMB]. The number of remaining Mailboxes available will be: (SETUP_MB - 8) - (RFFN × 2) If the Number of Rx FIFO Filters programmed through RFFN exceeds the SETUP_MB value (memory space available) the exceeding ones will not be functional. **NOTE:** • The number of the last remaining available mailboxes is defined by the least value between the NUMBER_OF_MB minus 1 and the CAN_MCR[MAXMB] field. • If Rx Individual Mask Registers are not enabled then all Rx FIFO filters are affected by the Rx FIFO Global Mask.

RFFN[3:0]	Number of Rx FIFO filter elements	Message Buffers occupied by Rx FIFO and ID Filter Table	Remaining Available Mailboxes	Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks	Rx FIFO ID Filter Table Elements Affected by Rx FIFO Global Mask
0x0	8	MB 0-7	MB 8-63	Elements 0-7	none
0x1	16	MB 0-9	MB 10-63	Elements 0-9	Elements 10-15
0x2	24	MB 0-11	MB 12-63	Elements 0-11	Elements 12-23
0x3	32	MB 0-13	MB 14-63	Elements 0-13	Elements 14-31
0x4	40	MB 0-15	MB 16-63	Elements 0-15	Elements 16-39
0x5	48	MB 0-17	MB 18-63	Elements 0-17	Elements 18-47
0x6	56	MB 0-19	MB 20-63	Elements 0-19	Elements 20-55

Table continues on the next page...

CAN_CTRL2 field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>RFFN[3:0]</th><th>Number of Rx FIFO filter elements</th><th>Message Buffers occupied by Rx FIFO and ID Filter Table</th><th>Remaining Available Mailboxes</th><th>Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks</th><th>Rx FIFO ID Filter Table Elements Affected by Rx FIFO Global Mask</th></tr>
<tr><td>0x7</td><td>64</td><td>MB 0-21</td><td>MB 22-63</td><td>Elements 0-21</td><td>Elements 22-63</td></tr>
<tr><td>0x8</td><td>72</td><td>MB 0-23</td><td>MB 24-63</td><td>Elements 0-23</td><td>Elements 24-71</td></tr>
<tr><td>0x9</td><td>80</td><td>MB 0-25</td><td>MB 26-63</td><td>Elements 0-25</td><td>Elements 26-79</td></tr>
<tr><td>0xA</td><td>88</td><td>MB 0-27</td><td>MB 28-63</td><td>Elements 0-27</td><td>Elements 28-87</td></tr>
<tr><td>0xB</td><td>96</td><td>MB 0-29</td><td>MB 30-63</td><td>Elements 0-29</td><td>Elements 30-95</td></tr>
<tr><td>0xC</td><td>104</td><td>MB 0-31</td><td>MB 32-63</td><td>Elements 0-31</td><td>Elements 32-103</td></tr>
<tr><td>0xD</td><td>112</td><td>MB 0-33</td><td>MB 34-63</td><td>Elements 0-31</td><td>Elements 32-111</td></tr>
<tr><td>0xE</td><td>120</td><td>MB 0-35</td><td>MB 36-63</td><td>Elements 0-31</td><td>Elements 32-119</td></tr>
<tr><td>0xF</td><td>128</td><td>MB 0-37</td><td>MB 38-63</td><td>Elements 0-31</td><td>Elements 32-127</td></tr>
</table>
</td></tr>
<tr><td>8–12
TASD</td><td>Tx Arbitration Start Delay

This 5-bit field indicates how many CAN bits the Tx arbitration process start point can be delayed from the first bit of CRC field on CAN bus. See Tx Arbitration start delay for more details. This field can be written only in Freeze mode because it is blocked by hardware in other modes.</td></tr>
<tr><td>13
MRP</td><td>Mailboxes Reception Priority

If this bit is set the matching process starts from the Mailboxes and if no match occurs the matching continues on the Rx FIFO. This bit can be written only in Freeze mode because it is blocked by hardware in other modes.

0 Matching starts from Rx FIFO and continues on Mailboxes.
1 Matching starts from Mailboxes and continues on Rx FIFO.</td></tr>
<tr><td>14
RRS</td><td>Remote Request Storing

If this bit is asserted Remote Request Frame is submitted to a matching process and stored in the corresponding Message Buffer in the same fashion of a Data Frame. No automatic Remote Response Frame will be generated.

If this bit is negated the Remote Request Frame is submitted to a matching process and an automatic Remote Response Frame is generated if a Message Buffer with CODE=0b1010 is found with the same ID.

This bit can be written only in Freeze mode because it is blocked by hardware in other modes.

0 Remote Response Frame is generated.
1 Remote Request Frame is stored.</td></tr>
<tr><td>15
EACEN</td><td>Entire Frame Arbitration Field Comparison Enable For Rx Mailboxes

This bit controls the comparison of IDE and RTR bits within Rx Mailboxes filters with their corresponding bits in the incoming frame by the matching process. This bit does not affect matching for Rx FIFO. This bit can be written only in Freeze mode because it is blocked by hardware in other modes.

0 Rx Mailbox filter’s IDE bit is always compared and RTR is never compared despite mask bits.
1 Enables the comparison of both Rx Mailbox filter’s IDE and RTR bit with their corresponding bits within the incoming frame. Mask bits do apply.</td></tr>
</table>

Table continues on the next page...

CAN_CTRL2 field descriptions (continued)

Field	Description
16 Reserved	This field is reserved. When writing to this field, always write the reset value.
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 Reserved	This field is reserved. When writing to this field, always write the reset value.
31 Reserved	This field is reserved. When writing to this field, always write the reset value.

47.4.15 Error and Status 2 register (CAN_ESR2)

This register reports some general status information.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0									LPTM						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	VPS	IMB	0												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ESR2 field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 LPTM	Lowest Priority Tx Mailbox If CAN_ESR2[VPS] is asserted, this field indicates the lowest number inactive Mailbox (see the CAN_ESR2[IMB] bit description). If there is no inactive Mailbox then the Mailbox indicated depends on CAN_CTRL1[LBUF] bit value. If CAN_CTRL1[LBUF] bit is negated then the Mailbox indicated is the one that has the greatest arbitration value (see the "Highest priority Mailbox first" section). If CAN_CTRL1[LBUF] bit is asserted then the Mailbox indicated is the highest number active Tx Mailbox. If a

Table continues on the next page...

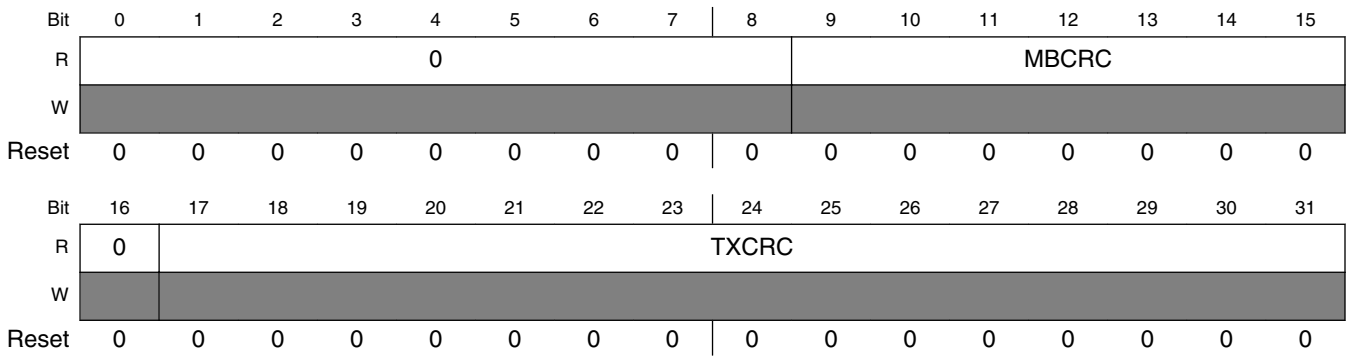
CAN_ESR2 field descriptions (continued)

Field	Description
	Tx Mailbox is being transmitted it is not considered in LPTM calculation. If CAN_ESR2[IMB] is not asserted and a frame is transmitted successfully, LPTM is updated with its Mailbox number.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 VPS	Valid Priority Status This bit indicates whether CAN_ESR2[IMB] and CAN_ESR2[LPTM] contents are currently valid or not. It is asserted upon every complete Tx arbitration process unless the CPU writes to Control and Status word of a Mailbox that has already been scanned, that is, it is behind Tx Arbitration Pointer, during the Tx arbitration process. If there is no inactive Mailbox and only one Tx Mailbox that is being transmitted then VPS is not asserted. This bit is negated upon the start of every Tx arbitration process or upon a write to Control and Status word of any Mailbox. **NOTE:** CAN_ESR2[VPS] is not affected by any CPU write into Control Status (C/S) of a MB that is blocked by abort mechanism. When CAN_MCR[AEN] is asserted, the abort code write in C/S of a MB that is being transmitted (pending abort), or any write attempt into a Tx MB with CAN_IFLAG set is blocked. 0 Contents of IMB and LPTM are invalid. 1 Contents of IMB and LPTM are valid.
18 IMB	Inactive Mailbox If ESR2[VPS] is asserted, this bit indicates whether there is any inactive Mailbox (CODE field is either 0b1000 or 0b0000). This bit is asserted in the following cases: • During arbitration, if an CAN_ESR2[LPTM] is found and it is inactive. • If CAN_ESR2[IMB] is not asserted and a frame is transmitted successfully. This bit is cleared in all start of arbitration (see Section "Arbitration process"). **NOTE:** CAN_ESR2[LPTM] mechanism have the following behavior: if an MB is successfully transmitted and CAN_ESR2[IMB]=0 (no inactive Mailbox), then CAN_ESR2[VPS] and CAN_ESR2[IMB] are asserted and the index related to the MB just transmitted is loaded into CAN_ESR2[LPTM]. 0 If ESR2[VPS] is asserted, the ESR2[LPTM] is not an inactive Mailbox. 1 If ESR2[VPS] is asserted, there is at least one inactive Mailbox. LPTM content is the number of the first one.
19–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

47.4.16 CRC Register (CAN_CRCR)

This register provides information about the CRC of transmitted messages. This register is updated at the same time the Tx Interrupt Flag is asserted.

Address: 0h base + 44h offset = 44h



CAN_CRCR field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 MBCRC	CRC Mailbox This field indicates the number of the Mailbox corresponding to the value in CAN_CRCR[TXCRC] field.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–31 TXCRC	Transmitted CRC value This field indicates the CRC value of the last transmitted message.

47.4.17 Rx FIFO Global Mask register (CAN_RXFGMASK)

This register is located in RAM.

If Rx FIFO is enabled, RXFGMASK is used to mask the Rx FIFO ID Filter Table elements that do not have a corresponding RXIMR according to CAN_CTRL2[RFFN] field setting.

This register can only be written in Freeze mode as it is blocked by hardware in other modes.

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	FGM																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

CAN_RXFGMASK field descriptions

Field	Description
0–31 FGM	Rx FIFO Global Mask Bits These bits mask the ID Filter Table elements bits in a perfect alignment. The following table shows how the FGM bits correspond to each IDAF field.

Rx FIFO ID Filter Table Elements Format (CAN_MCR[IDAM])	Identifier Acceptance Filter Fields					
	RTR	IDE	RXIDA	RXIDB [1]	RXIDC [2]	Reserved
A	FGM[31]	FGM[30]	FGM[29:1]	-	-	FGM[0]
B	FGM[31], FGM[15]	FGM[30], FGM[14]	-	FGM[29:16], FGM[13:0]		-
C	-	-		-	FGM[31:24], FGM[23:16], FGM[15:8], FGM[7:0]	

1. If CAN_MCR[IDAM] field is equivalent to the format B only the fourteen most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.
2. If CAN_MCR[IDAM] field is equivalent to the format C only the eight most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.

0 The corresponding bit in the filter is "don't care."
1 The corresponding bit in the filter is checked.

1. If CAN_MCR[IDAM] field is equivalent to the format B only the fourteen most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.
2. If CAN_MCR[IDAM] field is equivalent to the format C only the eight most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.

47.4.18 Rx FIFO Information Register (CAN_RXFIR)

RXFIR provides information on Rx FIFO.

This register is the port through which the CPU accesses the output of the RXFIR FIFO located in RAM. The RXFIR FIFO is written by the FlexCAN whenever a new message is moved into the Rx FIFO as well as its output is updated whenever the output of the Rx FIFO is updated with the next message. See Section "Rx FIFO" for instructions on reading this register.

Address: 0h base + 4Ch offset = 4Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																							IDHIT								
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

CAN_RXFIR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 IDHIT	Identifier Acceptance Filter Hit Indicator This field indicates which Identifier Acceptance Filter was hit by the received message that is in the output of the Rx FIFO. If multiple filters match the incoming message ID then the first matching IDAF found (lowest number) by the matching process is indicated. This field is valid only while the CAN_IFLAG1[BUF5I] is asserted.

47.4.19 CAN Bit Timing Register (CAN_CBT)

This register is an alternative way to store the CAN bit timing variables described in CAN_CTRL1 register. EPRESDIV, EPROPSEG, EPSEG1, EPSEG2 and ERJW are extended versions of PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW bit fields respectively.

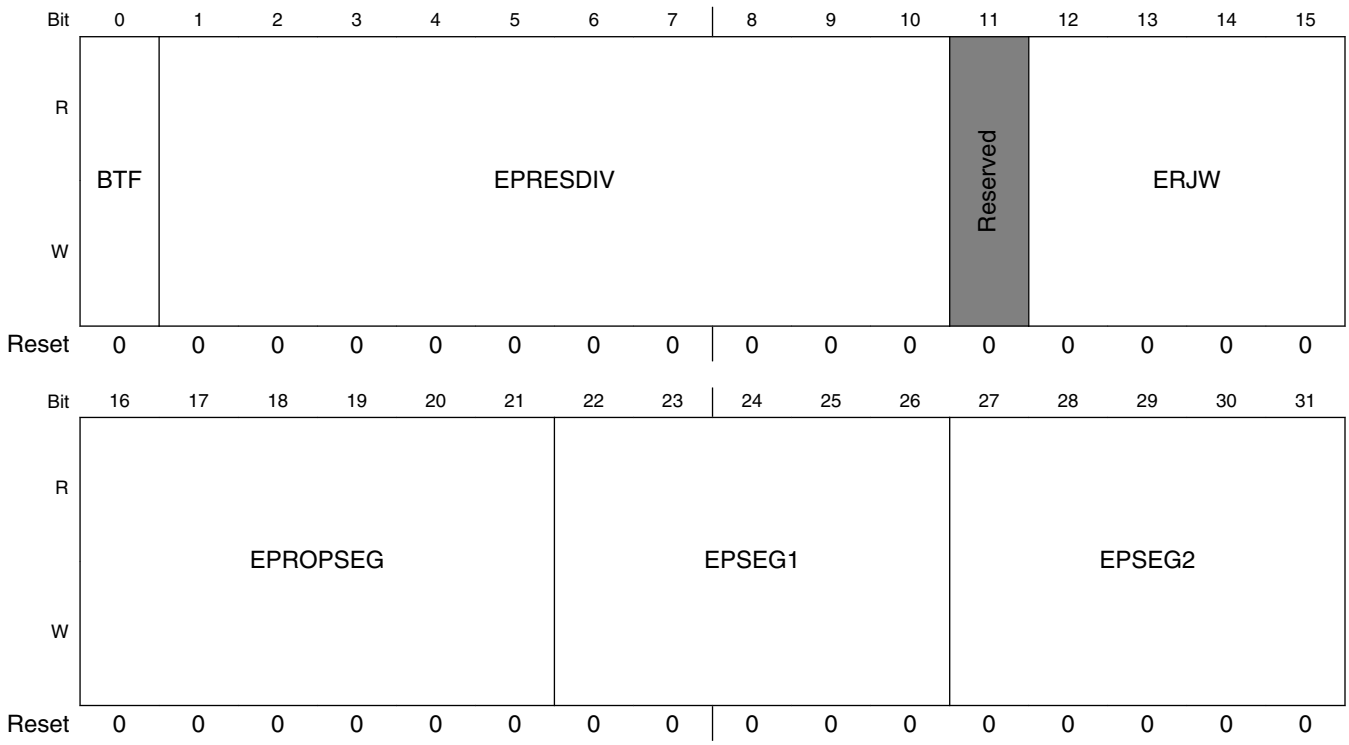
The BTF bit selects the use of the timing variables defined in this register.

The contents of this register are not affected by soft reset.

NOTE

The CAN bit variables in CAN_CTRL1 and in CAN_CBT are stored in the same register.

Address: 0h base + 50h offset = 50h



CAN_CBT field descriptions

Field	Description
0 BTF	Bit Timing Format Enable Enables the use of extended CAN bit timing fields EPRESDIV, EPROPSEG, EPSEG1, EPSEG2 and ERJW replacing the CAN bit timing variables defined in CAN_CTRL1 register. This field can be written in Freeze mode only. 0 Extended bit time definitions disabled. 1 Extended bit time definitions enabled.
1–10 EPRESDIV	Extended Prescaler Division Factor This 10-bit field defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency when CAN_CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CAN_CTRL1[PRESDIV] value range. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency (see Protocol timing). This field can be written only in Freeze mode because it is blocked by hardware in other modes. Sclock frequency = PE clock frequency / (EPRESDIV + 1)
11 Reserved	This field is reserved.
12–15 ERJW	Extended Resync Jump Width This 4-bit field defines the maximum number of time quanta that a bit time can be changed by one re-synchronization when CAN_CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CAN_CTRL1[RJW] value range.

Table continues on the next page...

CAN_CBT field descriptions (continued)

Field	Description
	One time quantum is equal to the Sclock period. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Resync Jump Width = ERJW + 1.
16–21 EPROPSEG	Extended Propagation Segment This 6-bit field defines the length of the Propagation Segment in the bit time when CAN_CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CAN_CTRL1[PROPSEG] value range. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Propagation Segment Time = (EPROPSEG + 1) × Time-Quanta. Time-Quantum = one Sclock period.
22–26 EPSEG1	Extended Phase Segment 1 This 5-bit field defines the length of Phase Segment 1 in the bit time when CAN_CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CAN_CTRL1[PSEG1] value range. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 1 = (EPSEG1 + 1) × Time-Quanta. Time-Quantum = one Sclock period.
27–31 EPSEG2	Extended Phase Segment 2 This 5-bit field defines the length of Phase Segment 2 in the bit time when CAN_CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CAN_CTRL1[PSEG2] value range. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 1 = (EPSEG2 + 1) × Time-Quanta. Time-Quantum = one Sclock period.

47.4.20 Interrupt Masks 3 Register (CAN_IMASK3)

This register allows any number of a range of the 32 Message Buffer Interrupts to be enabled or disabled for MB95 to MB64. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception, that is, when the corresponding CAN_IFLAG3 bit is set.

Address: 0h base + 6Ch offset = 6Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	BUF95TO64M																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IMASK3 field descriptions

Field	Description
0–31 BUF95TO64M	Buffer MB $_i$ Mask Each bit enables or disables the corresponding FlexCAN Message Buffer Interrupt for MB95 to MB64.

CAN_IMASK3 field descriptions (continued)

Field	Description
	NOTE: Setting or clearing a bit in the CAN_IMASK3 Register can assert or negate an interrupt request, if the corresponding CAN_IFLAG3 bit is set. 0 The corresponding buffer Interrupt is disabled. 1 The corresponding buffer Interrupt is enabled.

47.4.21 Interrupt Flags 3 Register (CAN_IFLAG3)

This register defines the flags for the 32 Message Buffer interrupts for MB95 to MB64. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding CAN_IFLAG3 bit. If the corresponding CAN_IMASK3 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect.

Before updating CAN_MCR[MAXMB] field, CPU must service the CAN_IFLAG3 bits whose MB value is greater than the MAXMB to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.

Address: 0h base + 74h offset = 74h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BUF95TO64																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IFLAG3 field descriptions

Field	Description
0–31 BUF95TO64	Buffer MB $_i$ Interrupt Each bit flags the corresponding FlexCAN Message Buffer interrupt for MB95 to MB64. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception 1 The corresponding buffer has successfully completed transmission or reception

47.4.22 Rx Individual Mask Registers (CAN_RXIMR*n*)

The RX Individual Mask Registers are used to store the acceptance masks for ID filtering in Rx MBs and the Rx FIFO.

When the Rx FIFO is disabled (CAN_MCR[RFEN] bit is negated), an individual mask is provided for each available Rx Mailbox on a one-to-one correspondence. When the Rx FIFO is enabled (CAN_MCR[RFEN] bit is asserted), an individual mask is provided for each Rx FIFO ID Filter Table Element on a one-to-one correspondence depending on the setting of CAN_CTRL2[RFFN] (see Rx FIFO).

CAN_RXIMR0 stores the individual mask associated to either MB0 or ID Filter Table Element 0, CAN_RXIMR1 stores the individual mask associated to either MB1 or ID Filter Table Element 1 and so on.

CAN_RXIMR registers can only be accessed by the CPU while the module is in Freeze mode, otherwise, they are blocked by hardware. These registers are not affected by reset. They are located in RAM and must be explicitly initialized prior to any reception.

Address: 0h base + 880h offset + (4d × i), where i=0d to 95d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MI																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

CAN_RXIMR*n* field descriptions

Field	Description
0–31 MI	Individual Mask Bits Each Individual Mask Bit masks the corresponding bit in both the Mailbox filter and Rx FIFO ID Filter Table element in distinct ways. For Mailbox filters, see the RXMGMASK register description. For Rx FIFO ID Filter Table elements, see the RXFGMASK register description. 0 The corresponding bit in the filter is "don't care." 1 The corresponding bit in the filter is checked.

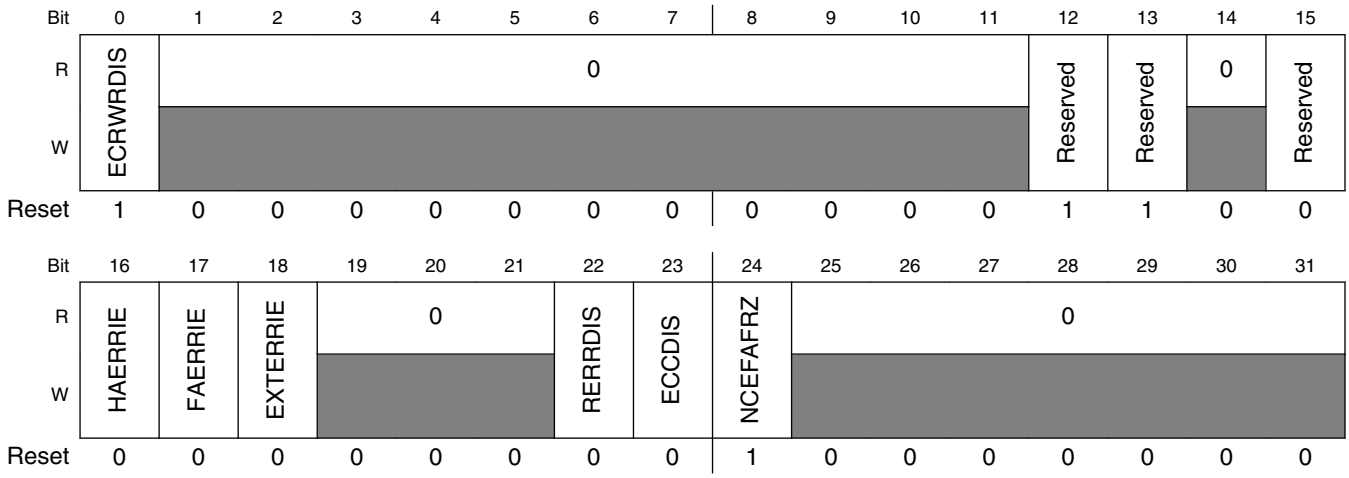
47.4.23 Memory Error Control Register (CAN_MECR)

This register contains control bits for memory error detection and correction (ECC).

NOTE

When bit CTRL2[ECRWRE] is 0, writes in this register are blocked, except in the ECRWRDIS bit.

Address: 0h base + AE0h offset = AE0h



CAN_MECR field descriptions

Field	Description
0 ECRWRDIS	Error Configuration Register Write Disable Disables writes on this register. This bit is automatically set to 1 (disabled) when CTRL2[ECRWRE] is enabled. The protocol described in section Detection and Correction of Memory Errors must be followed. 0 Write is enabled. 1 Write is disabled.
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 Reserved	This field is reserved. When writing to this field, always write the reset value.
13 Reserved	This field is reserved. When writing to this field, always write the reset value.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 Reserved	This field is reserved. When writing to this field, always write the reset value.
16 HAERRIE	Host Access Error Injection Enable Enables the injection of errors only in memory reads issued by the host (CPU). 0 Injection is disabled. 1 Injection is enabled.
17 FAERRIE	FlexCAN Access Error Injection Enable Enables the injection of errors only in memory reads issued by the FlexCAN internal processes. 0 Injection is disabled. 1 Injection is enabled.
18 EXTERRIE	Extended Error Injection Enable Memory accesses performed by internal FlexCAN processes are 64-bit. This bit extends the error injection on 32-bit memory accesses to the complementary 32-bit word using the same 32-bit error injection data

Table continues on the next page...

CAN_MECR field descriptions (continued)

Field	Description
	and parity words. See Error Injection Data Pattern Register (CAN_FCERRIDPR) and Error Injection Parity Pattern Register (CAN_FCERRIPPR) 0 Error injection is applied only to the 32-bit word. 1 Error injection is applied to the 64-bit word.
19–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 RERRDIS	Error Report Disable Disables the update of the error report registers. The update of error-related flags and the generation of bus transfer errors are still active. **NOTE:** When reading the report registers, this bit must be set to assure coherence on the consecutive register reads. 0 Enable updates of the error report registers. 1 Disable updates of the error report registers.
23 ECCDIS	Error Correction Disable Disables completely the memory detection and correction mechanism. Besides disabling the error report mechanism, it also stops the update of the error-related flags and generation of bus transfer errors. The parity bits continue being calculated and written into memory on write transactions. 0 Enable memory error correction. 1 Disable memory error correction.
24 NCEFAFRZ	Non-Correctable Errors In FlexCAN Access Put Device In Freeze Mode Determines the response when a non-correctable error is detected in a memory read performed by FlexCAN internal processes. In this case, entering Freeze mode prevents corrupted data from being treated as valid by FlexCAN internal processes. 0 Keep normal operation. 1 Put FlexCAN in Freeze mode (according to the "Freeze mode" section).
25–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

47.4.24 Error Injection Address Register (CAN_ERRIAR)

This register holds the address where error is to be injected.

See the chip-specific FlexCAN information for the FlexCAN RAM to memory map address mapping table.

Address: 0h base + AE4h offset = AE4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																INJADDR															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ERRIAR field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–31 INJADDR	Error Injection Address This bit field defines the physical RAM address where error is to be injected (see table above). **NOTE:** The two least significant bits are always read as zero.

47.4.25 Error Injection Data Pattern Register (CAN_ERRIDPR)

Holds the error pattern to be injected in the data word read from memory.

Address: 0h base + AE8h offset = AE8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DFLIP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ERRIDPR field descriptions

Field	Description
0–31 DFLIP	Data flip pattern Bits set to 1 in the flip pattern cause the corresponding data bit in the word read from memory to invert.

47.4.26 Error Injection Parity Pattern Register (CAN_ERRIPPR)

Holds the error pattern to be injected in parity bits read from memory along with data word.

Bits set to 1 in the flip pattern cause the corresponding parity bit in the word read from memory to invert.

Address: 0h base + AECh offset = AECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			PFLIP3					0			PFLIP2					0			PFLIP1				0			PFLIP0				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ERRIPPR field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 PFLIP3	Parity Flip Pattern For Byte 3 (most significant)
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 PFLIP2	Parity Flip Pattern For Byte 2
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 PFLIP1	Parity Flip Pattern For Byte 1
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 PFLIP0	Parity Flip Pattern For Byte 0 (Least Significant)

47.4.27 Error Report Address Register (CAN_RERRAR)

Reports the address used for an access in which an error (correctable or non-correctable) was detected, and also reports the identification of the source of that access.

This address is always reported using a 32-bit alignment. Non-aligned accesses (ERRADDR[1:0] non-0) are reported with the address aligned and data is reported in RERRDR accordingly shifted. In case of errors detected in accesses larger than 32-bit (as performed by FlexCAN internal processes), the address of the 32-bit word which the error was detected is reported. In case of errors detected in more than one 32-bit word, only the least significant address is reported.

Address: 0h base + AF0h offset = AF0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							NCE	0					SAID		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		ERRADDR													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_RERRAR field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 NCE	Non-Correctable Error Indicates that the report is due to an non-correctable error. 0 Reporting a correctable-error 1 Reporting a non-correctable error
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 SAID	SAID[2] — Identification of the requestor of the memory read request: • 0 = Requested by FlexCAN internal processes • 1 = Requested by Host (CPU) SAID[1] — Details of FlexCAN operation: • 0 = Move • 1 = Scanning SAID[0] — Operation that requested the memory read: • 0 = Transmission • 1 = Reception
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–31 ERRADDR	Address Where The Error Was Detected See the description of the Error Injection Address Register (ERRIAR).

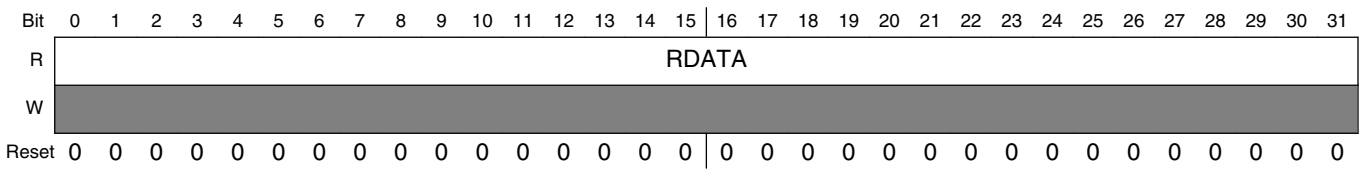
Table 47-130. Source of memory access

SAID[2:0]	Error during...
0	Move-out FlexCAN access
1	Move-in
2	Tx Arbitration
3	Rx Matching
4	Move-out Host access
5-7	Reserved

47.4.28 Error Report Data Register (CAN_RERRDR)

Reports the raw data (unmodified by the correction performed by ECC logic) read from memory with error. The value reported does not represent the transient values of the BUSY bit (see the “Message Buffer Code for Rx buffers” table) when reading a Message Buffer.

Address: 0h base + AF4h offset = AF4h



CAN_RERRDR field descriptions

Field	Description
0–31 RDATA	Raw data word read from memory with error

47.4.29 Error Report Syndrome Register (CAN_RERRSYNR)

Holds the syndrome detected in a memory read with error. It also reports the bytes which were read in this 32-bit read transaction.

Each SYNDn field indicates the type of error and which bit in byte (n) is affected by the error. (SYND3 corresponds to the most significant byte in the data word read from memory; SYND0 corresponds to the least significant.)

Table 47-133. Syndrome Definition

SYNDn (hex)	Type	Bit affected
00	-	none (no error)
01	Code	0
02	Code	1
04	Code	2
07	Data	5
08	Code	3
0E	Data	7
10	Code	4
13	Data	2

Table continues on the next page...

Table 47-133. Syndrome Definition (continued)

SYNDn (hex)	Type	Bit affected
15	Data	6
16	Data	1
19	Data	3
1A	Data	4
1C	Data	0
06	-	All-zeros non-correctable error
1F	-	All-ones non-correctable error
All others	-	Non-correctable error

Each BEn field indicates which byte in the 32-bit word reported was effectively read. The syndrome bits are calculated for all bytes, even for the non-read ones. Errors detected in non-read bytes are indicated (see the "Error Indication" section) and reported (see the "Error Reporting" section).

Address: 0h base + AF8h offset = AF8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BE3	0		SYND3					BE2	0		SYND2				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BE1	0		SYND1					BE0	0		SYND0				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_RERRSYNR field descriptions

Field	Description
0 BE3	Byte Enabled For Byte 3 (Most Significant) 0 The byte was not read. 1 The byte was read.
1–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 SYND3	Error Syndrome For Byte 3 (Most Significant) See the "Syndrome Definition" table.
8 BE2	Byte Enabled For Byte 2 0 The byte was not read. 1 The byte was read.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

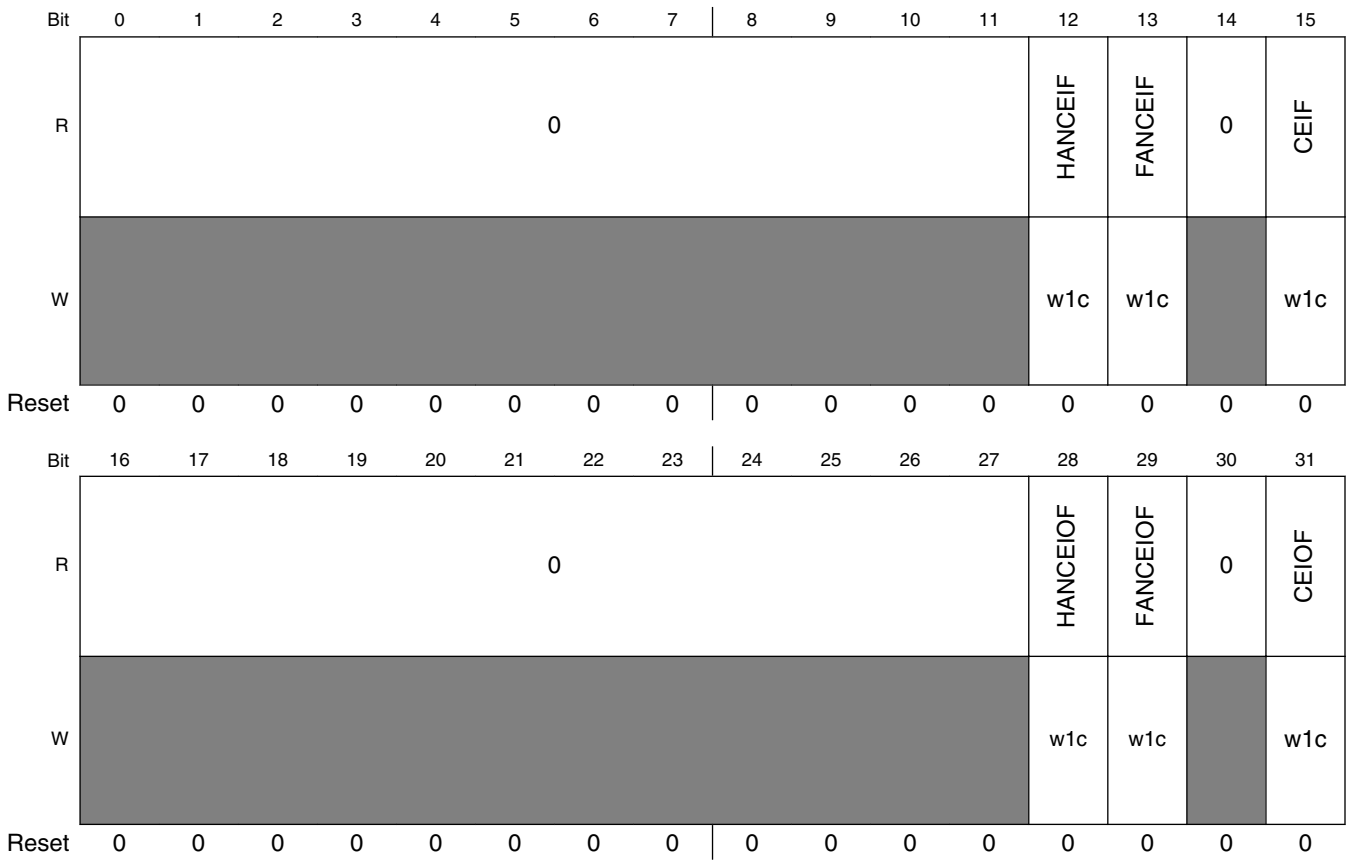
CAN_RERRSYNR field descriptions (continued)

Field	Description
11–15 SYND2	Error Syndrome For Byte 2 See the "Syndrome Definition" table.
16 BE1	Byte Enabled For Byte 1 0 The byte was not read. 1 The byte was read.
17–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 SYND1	Error Syndrome for Byte 1 See the "Syndrome Definition" table.
24 BE0	Byte Enabled For Byte 0 (least significant) 0 The byte was not read. 1 The byte was read.
25–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 SYND0	Error Syndrome For Byte 0 (least significant) See the "Syndrome Definition" table.

47.4.30 Error Status Register (CAN_ERRSR)

Holds the status bits of the error correction and detection operations. These flags can be cleared by writing 1 to them. Writing 0 has no effect.

Address: 0h base + AFCh offset = AFCh



CAN_ERRSR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 HANCEIF	Host Access With Non-Correctable Error Interrupt Flag Indicates that a non-correctable error was detected in a memory read initiated by Host. A bus transfer error is asserted for that access.No interrupt is associated to this flag. 0 No non-correctable errors were detected in Host accesses so far. 1 A non-correctable error was detected in a Host access.
13 FANCEIF	FlexCAN Access With Non-Correctable Error Interrupt Flag

Table continues on the next page...

CAN_ERRSR field descriptions (continued)

Field	Description
	Indicates that a non-correctable error was detected in a memory read initiated by FlexCAN internal processes.No interrupt is associated to this flag. 0 No non-correctable errors were detected in FlexCAN accesses so far. 1 A non-correctable error was detected in a FlexCAN access.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 CEIF	Correctable Error Interrupt Flag Indicates that a correctable error was detected in a memory read. 0 No correctable errors were detected so far. 1 A correctable error was detected.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 HANCEIOF	Host Access With Non-Correctable Error Interrupt Overrun Flag Indicates that a non-correctable error was detected in a memory read initiated by Host when HANCEIF was set. No interrupt is associated to this flag. See Error Indication. 0 No overrun on non-correctable errors in Host access 1 Overrun on non-correctable errors in Host access
29 FANCEIOF	FlexCAN Access With Non-Correctable Error Interrupt Overrun Flag Indicates that a non-correctable error was detected in a memory read initiated by FlexCAN internal processes when FANCEIF was set. No interrupt is associated to this flag. See Error Indication. 0 No overrun on non-correctable errors in FlexCAN access 1 Overrun on non-correctable errors in FlexCAN access
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CEIOF	Correctable Error Interrupt Overrun Flag Indicates that a correctable error was detected in a memory read when CEIF was set. No interrupt is associated to this flag. See Error Indication. 0 No overrun on correctable errors 1 Overrun on correctable errors

47.4.31 Message buffer structure

The message buffer structure used by the FlexCAN module is represented in the following figure. Both Extended (29-bit identifier) and Standard (11-bit identifier) frames used in the CAN specification (Version 2.0 Part B) are represented. Each individual MB is formed by 16 bytes.

The memory area from 0x80 to 0x67F is used by the mailboxes.

Table 47-135. Message buffer structure

	0	1	2	3	4	7	8	9	10	11	12	13	14	15	16	23	24	31
0x0	EDL	BRS	ESI		CODE			SRR	IDE	RTR	DLC				TIME STAMP			
0x4	PRIO				ID (Standard/Extended)										ID (Extended)			
0x8	Data Byte 0							Data Byte 1							Data Byte 2		Data Byte 3	
0xC	Data Byte 4							Data Byte 5							Data Byte 6		Data Byte 7	
	= Unimplemented or Reserved																	

CODE - Message Buffer Code

This 4-bit field can be accessed (read or write) by the CPU and by the FlexCAN module itself, as part of the message buffer matching and arbitration process. The encoding is shown in Table 47-136 and Table 47-137. See Functional description for additional information.

Table 47-136. Message buffer code for Rx buffers

CODE description	Rx code BEFORE receive new frame	SRV[1]	Rx code AFTER successful reception[2]	RRS[3]	Comment
0b0000: INACTIVE - MB is not active.	INACTIVE	-	-	-	MB does not participate in the matching process.
0b0100: EMPTY - MB is active and empty.	EMPTY	-	FULL	-	When a frame is received successfully (after the Move-in) process), the CODE field is automatically updated to FULL.

Table continues on the next page...

Table 47-136. Message buffer code for Rx buffers (continued)

CODE description	Rx code BEFORE receive new frame	SRV[1]	Rx code AFTER successful reception[2]	RRS[3]	Comment
0b0010: FULL - MB is full.	FULL	Yes	FULL	-	The act of reading the C/S word followed by unlocking the MB (SRV) does not make the code return to EMPTY. It remains FULL. If a new frame is moved to the MB after the MB was serviced, the code still remains FULL. See Matching process for matching details related to FULL code.
		No	OVERRUN	-	If the MB is FULL and a new frame is moved to this MB before the CPU services it, the CODE field is automatically updated to OVERRUN. See Matching process for details about overrun behavior.
0b0110: OVERRUN - MB is being overwritten into a full buffer.	OVERRUN	Yes	FULL	-	If the CODE field indicates OVERRUN and CPU has serviced the MB, when a new frame is moved to the MB then the code returns to FULL.
		No	OVERRUN	-	If the CODE field already indicates OVERRUN, and another new frame must be moved, the MB will be overwritten again, and the code will remain OVERRUN. See Matching process for details about overrun behavior.

Table continues on the next page...

Table 47-136. Message buffer code for Rx buffers (continued)

CODE description	Rx code BEFORE receive new frame	SRV[1]	Rx code AFTER successful reception[2]	RRS[3]	Comment
0b1010: RANSWER[4] - A frame was configured to recognize a Remote Request Frame and transmit a Response Frame in return.	RANSWER	-	TANSWER(0b1110)	0	A Remote Answer was configured to recognize a remote request frame received. After that an MB is set to transmit a response frame. The code is automatically changed to TANSWER (0b1110). See Matching process for details. If CAN_CTRL2[RRS] is negated, transmit a response frame whenever a remote request frame with the same ID is received.
		-	-	1	This code is ignored during matching and arbitration process. See Matching process for details.
CODE[0]=1: BUSY - FlexCAN is updating the contents of the MB. The CPU must not access the MB.	BUSY[5]	-	FULL	-	Indicates that the MB is being updated. It will be negated automatically and does not interfere with the next CODE.
		-	OVERRUN	-	

1. SRV: Serviced MB. MB was read and unlocked by reading TIMER or other MB.
2. A frame is considered a successful reception after the frame to be moved to MB (move-in process). See Move-in for details.
3. Remote Request Stored bit, see "Control 2 Register (CAN_CTRL2)" for details.
4. Code 0b1010 is not considered Tx and an MB with this code should not be aborted.
5. Note that for Tx MBs, the BUSY bit should be ignored upon read, except when AEN bit is set in the MCR register. If this bit is asserted, the corresponding MB does not participate in the matching process.

Table 47-137. Message buffer code for Tx buffers

CODE Description	Tx Code BEFORE tx frame	MB RTR	Tx Code AFTER successful transmission	Comment
0b1000: INACTIVE - MB is not active	INACTIVE	-	-	MB does not participate in arbitration process.

Table continues on the next page...

Table 47-137. Message buffer code for Tx buffers (continued)

CODE Description	Tx Code BEFORE tx frame	MB RTR	Tx Code AFTER successful transmission	Comment
0b1001: ABORT - MB is aborted	ABORT	-	-	MB does not participate in arbitration process.
0b1100: DATA - MB is a Tx Data Frame (MB RTR must be 0)	DATA	0	INACTIVE	Transmit data frame unconditionally once. After transmission, the MB automatically returns to the INACTIVE state.
0b1100: REMOTE - MB is a Tx Remote Request Frame (MB RTR must be 1)	REMOTE	1	EMPTY	Transmit remote request frame unconditionally once. After transmission, the MB automatically becomes an Rx Empty MB with the same ID.
0b1110: TANSWER - MB is a Tx Response Frame from an incoming Remote Request Frame	TANSWER	-	RANSWER	This is an intermediate code that is automatically written to the MB by the CHI as a result of a match to a remote request frame. The remote response frame will be transmitted unconditionally once, and then the code will automatically return to RANSWER (0b1010). The CPU can also write this code with the same effect. The remote response frame can be either a data frame or another remote request frame depending on the RTR bit value. See Matching process and Arbitration process for details.

SRR - Substitute Remote Request

Fixed recessive bit, used only in extended format. It must be set to one by the user for transmission (Tx Buffers) and will be stored with the value received on the CAN bus for Rx receiving buffers. It can be received as either recessive or dominant. If FlexCAN receives this bit as dominant, then it is interpreted as an arbitration loss.

1 = Recessive value is compulsory for transmission in extended format frames

0 = Dominant is not a valid value for transmission in extended format frames

IDE - ID Extended Bit

This field identifies whether the frame format is standard or extended.

1 = Frame format is extended

0 = Frame format is standard

RTR - Remote Transmission Request

This bit affects the behavior of remote frames and is part of the reception filter. See Table 47-136, Table 47-137, and the description of the RRS bit in Control 2 Register (CAN_CTRL2) for additional details.

If FlexCAN transmits this bit as '1' (recessive) and receives it as '0' (dominant), it is interpreted as an arbitration loss. If this bit is transmitted as '0' (dominant), then if it is received as '1' (recessive), the FlexCAN module treats it as a bit error. If the value received matches the value transmitted, it is considered a successful bit transmission.

1 = Indicates the current MB may have a remote request frame to be transmitted if MB is Tx. If the MB is Rx then incoming remote request frames may be stored.

0 = Indicates the current MB has a data frame to be transmitted. In Rx MB it may be considered in matching processes.

DLC - Length of Data in Bytes

This 4-bit field is the length (in bytes) of the Rx or Tx data, which is located in offset 0x8 through 0xF of the MB space (see Table 47-135) . In reception, this field is written by the FlexCAN module, copied from the DLC (Data Length Code) field of the received frame. In transmission, this field is written by the CPU and corresponds to the DLC field value of the frame to be transmitted. When RTR = 1, the frame to be transmitted is a remote frame and does not include the data field, regardless of the DLC field (see Table 47-138).

TIME STAMP - Free-Running Counter Time Stamp

This 16-bit field is a copy of the Free-Running Timer, captured for Tx and Rx frames at the time when the beginning of the Identifier field appears on the CAN bus.

PRIO - Local priority

This 3-bit field is used only when LPRIO_EN bit is set in CAN_MCR, and it only makes sense for Tx mailboxes. These bits are not transmitted. They are appended to the regular ID to define the transmission priority. See Arbitration process.

ID - Frame Identifier

In standard frame format, only the 11 most significant bits (3 to 13) are used for frame identification in both receive and transmit cases. The 18 least significant bits are ignored. In extended frame format, all bits are used for frame identification in both receive and transmit cases.

DATA BYTE 0 to 7 - Data Field

Up to eight bytes can be used for a data frame.

For Rx frames, the data is stored as it is received from the CAN bus. DATA BYTE (*n*) is valid only if *n* is less than DLC as shown in the table below.

Table 47-138. DATA BYTEs validity

DLC	Valid DATA BYTEs
0	none
1	DATA BYTE 0
2	DATA BYTE 0 to 1
3	DATA BYTE 0 to 2
4	DATA BYTE 0 to 3
5	DATA BYTE 0 to 4
6	DATA BYTE 0 to 5
7	DATA BYTE 0 to 6
8 or above	DATA BYTE 0 to 7

47.4.32 Rx FIFO structure

When the CAN_MCR[RFEN] bit is set, the memory area from 0x80 to 0xDC (which is normally occupied by MBs 0–5) is used by the reception FIFO engine.

The region 0x80-0x8C contains the output of the FIFO which must be read by the CPU as a message buffer. This output contains the oldest message that has been received but not yet read. The region 0x90-0xDC is reserved for internal use of the FIFO engine.

An additional memory area, which starts at 0xE0 and may extend up to 0x2DC (normally occupied by MBs 6–37) depending on the CAN_CTRL2[RFFN] field setting, contains the ID filter table (configurable from 8 to 128 table elements) that specifies filtering criteria for accepting frames into the FIFO.

Out of reset, the ID filter table flexible memory area defaults to 0xE0 and extends only to 0xFC, which corresponds to MBs 6 to 7 for RFFN = 0, for backward compatibility with previous versions of FlexCAN.

The following shows the Rx FIFO data structure.

Table 47-139. Rx FIFO structure

	0 3 7 8 9 10 11 12 13 14 15 16 23 24 31							
0x80	IDHIT			SRR	IDE	RTR	DLC	TIME STAMP
0x84		ID standard					ID extended	
0x88	Data byte 0		Data byte 1				Data byte 2	Data byte 3
0x8C	Data byte 4		Data byte 5				Data byte 6	Data byte 7
0x90 to 0xDC	Reserved							
0xE0	ID filter table element 0							
0xE4	ID filter table element 1							
0xE8 to 0x2D4	ID filter table elements 2 to 125							
0x2D8	ID filter table element 126							
0x2DC	ID filter table element 127							
	= Unimplemented or reserved							

Each ID filter table element occupies an entire 32-bit word and can be compounded by one, two, or four Identifier Acceptance Filters (IDAF) depending on the CAN_MCR[IDAM] field setting. The following figures show the IDAF indexation.

The following table shows the three different formats of the ID table elements. Note that all elements of the table must have the same format. See Rx FIFO for more information.

Table 47-140. ID table structure

	0 1 2 7 8 15 16 17 18 23 24 30 31						
A	RTR	IDE	RXIDA (standard = 2–12, extended = 2–30)				
B	RTR	IDE	RXIDB_0 (standard = 2–12, extended = 2–15)	RTR	IDE	RXIDB_1 (standard = 18–28, extended = 18–31)	
C	RXIDC_0 (std/ext = 0–7)		RXIDC_1 (std/ext = 8–15)		RXIDC_2 (std/ext = 16–23)		RXIDC_3 (std/ext = 24–31)
	= Unimplemented or Reserved						

RTR — Remote Frame

This bit specifies if Remote Frames are accepted into the FIFO if they match the target ID.

1 = Remote Frames can be accepted and data frames are rejected

0 = Remote Frames are rejected and data frames can be accepted

IDE — Extended Frame

Specifies whether extended or standard frames are accepted into the FIFO if they match the target ID.

1 = Extended frames can be accepted and standard frames are rejected

0 = Extended frames are rejected and standard frames can be accepted

RXIDA — Rx Frame Identifier (Format A)

Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, only the 11 most significant bits (2 to 12) are used for frame identification. In the extended frame format, all bits are used.

RXIDB_0, RXIDB_1 — Rx Frame Identifier (Format B)

Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, the 11 most significant bits (a full standard ID) (2 to 12 and 18 to 28) are used for frame identification. In the extended frame format, all 14 bits of the field are compared to the 14 most significant bits of the received ID.

RXIDC_0, RXIDC_1, RXIDC_2, RXIDC_3 — Rx Frame Identifier (Format C)

Specifies an ID to be used as acceptance criteria for the FIFO. In both standard and extended frame formats, all 8 bits of the field are compared to the 8 most significant bits of the received ID.

IDHIT — Identifier Acceptance Filter Hit Indicator

This 9-bit field indicates which Identifier Acceptance Filter was hit by the received message that is in the output of the Rx FIFO. See Rx FIFO for more information.

47.5 Functional description

The FlexCAN module is a CAN protocol engine with a very flexible mailbox system for transmitting and receiving CAN frames. The mailbox system is composed by a set of Message Buffers (MB) that store configuration and control data, time stamp, message ID and data (see Message buffer structure). The memory corresponding to the first 38 MBs can be configured to support a FIFO reception scheme with a powerful ID filtering

mechanism, capable of checking incoming frames against a table of IDs (up to 128 extended IDs or 256 standard IDs or 512 8-bit ID slices), with individual mask register for up to 32 ID Filter Table elements.

Simultaneous reception through FIFO and mailbox is supported. For mailbox reception, a matching algorithm makes it possible to store received frames only into MBs that have the same ID programmed on its ID field. A masking scheme makes it possible to match the ID programmed on the MB with a range of IDs on received CAN frames. For transmission, an arbitration algorithm decides the prioritization of MBs to be transmitted based on the message ID (optionally augmented by 3 local priority bits) or the MB ordering.

Before proceeding with the functional description, an important concept must be explained. A Message Buffer is said to be "active" at a given time if it can participate in both the Matching and Arbitration processes. An Rx MB with a 0b0000 code is inactive (refer to Table 47-136). Similarly, a Tx MB with a 0b1000 or 0b1001 code is also inactive (refer to Table 47-137).

47.5.1 Transmit process

To transmit a CAN frame, the CPU must prepare a Message Buffer for transmission by executing the following procedure:

1. Check whether the respective interrupt bit is set and clear it.
2. If the MB is active (transmission pending), write the ABORT code (0b1001) to the CODE field of the Control and Status word to request an abortion of the transmission. Wait for the corresponding IFLAG bit to be asserted by polling the CAN_IFLAG register or by the interrupt request if enabled by the respective IMASK bit. Then read back the CODE field to check if the transmission was aborted or transmitted (see Transmission abort mechanism). If backwards compatibility is desired (CAN_MCR[AEN] bit is negated), just write the INACTIVE code (0b1000) to the CODE field to inactivate the MB but then the pending frame may be transmitted without notification (see Mailbox inactivation).
3. Write the ID word.
4. Write the data bytes.
5. Write the DLC, Control, and CODE fields of the Control and Status word to activate the MB.

When the MB is activated, it participates in the arbitration process and is eventually transmitted according to its priority.

At the end of the successful transmission, the value of the Free Running Timer is written into the Time Stamp field, the CODE field in the Control and Status word is updated, the CRC Register is updated, a status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit. The new CODE field after transmission depends on the code that was used to activate the MB (see Table 47-136 and Table 47-137 in Message buffer structure).

When the Abort feature is enabled (CAN_MCR[AEN] is asserted), after the Interrupt Flag is asserted for a Mailbox configured as transmit buffer, the Mailbox is blocked. Therefore the CPU is not able to update it until the Interrupt Flag is negated by CPU. This means that the CPU must clear the corresponding IFLAG bit before starting to prepare this MB for a new transmission or reception.

47.5.2 Arbitration process

The arbitration process scans the Mailboxes searching the Tx one that holds the message to be sent in the next opportunity. This Mailbox is called the *arbitration winner*.

The scan starts from the lowest number Mailbox and runs toward the higher ones.

The arbitration process is triggered in the following events:

- From the CRC field of the CAN frame. The start point depends on the CAN_CTRL2[TASD] field value.
- During the Error Delimiter field of a CAN frame.
- During the Overload Delimiter field of a CAN frame.
- When the winner is inactivated and the CAN bus has still not reached the first bit of the Intermission field.
- When there is CPU write to the C/S word of a winner MB and the CAN bus has still not reached the first bit of the Intermission field.
- When CHI is in Idle state and the CPU writes to the C/S word of any MB.
- When FlexCAN exits Bus Off state.
- Upon leaving Freeze mode or Low Power mode.

If the arbitration process does not manage to evaluate all Mailboxes before the CAN bus has reached the first bit of the Intermission field the temporary arbitration winner is invalidated and the FlexCAN will not compete for the CAN bus in the next opportunity.

The arbitration process selects the winner among the active Tx Mailboxes at the end of the scan according to both CAN_CTRL1[LBUF] and CAN_MCR[LPRIOEN] bits settings.

47.5.2.1 Lowest-number Mailbox first

If CAN_CTRL1[LBUF] bit is asserted the first (lowest number) active Tx Mailbox found is the arbitration winner. CAN_MCR[LPRIOEN] bit has no effect when CAN_CTRL1[LBUF] is asserted.

47.5.2.2 Highest-priority Mailbox first

If CAN_CTRL1[LBUF] bit is negated, then the arbitration process searches the active Tx Mailbox with the highest priority, which means that this Mailbox's frame would have a higher probability to win the arbitration on CAN bus when multiple external nodes compete for the bus at the same time.

The sequence of bits considered for this arbitration is called the *arbitration value* of the Mailbox. The highest-priority Tx Mailbox is the one that has the lowest arbitration value among all Tx Mailboxes.

If two or more Mailboxes have equivalent arbitration values, the Mailbox with the lowest number is the arbitration winner.

The composition of the arbitration value depends on CAN_MCR[LPRIOEN] bit setting.

47.5.2.2.1 Local Priority disabled

If CAN_MCR[LPRIOEN] bit is negated the arbitration value is built in the exact sequence of bits as they would be transmitted in a CAN frame (see the following table) in such a way that the Local Priority is disabled.

Table 47-141. Composition of the arbitration value when Local Priority is disabled

Format	Mailbox Arbitration Value (32 bits)				
Standard (IDE = 0)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	Extended ID[28:18] (11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

47.5.2.2.2 Local Priority enabled

If Local Priority is desired CAN_MCR[LPRIOEN] must be asserted. In this case the Mailbox PRIO field is included at the very left of the arbitration value (see the following table).

Table 47-142. Composition of the arbitration value when Local Priority is enabled

Format	Mailbox Arbitration Value (35 bits)					
Standard (IDE = 0)	PRIO (3 bits)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	PRIO (3 bits)	Extended ID[28:18] (11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

As the PRIO field is the most significant part of the arbitration value Mailboxes with low PRIO values have higher priority than Mailboxes with high PRIO values regardless the rest of their arbitration values.

Note that the PRIO field is not part of the frame on the CAN bus. Its purpose is only to affect the internal arbitration process.

47.5.2.3 Arbitration process (continued)

After the arbitration winner is found, its content is copied to a hidden auxiliary MB called Tx Serial Message Buffer (Tx SMB), which has the same structure as a normal MB but is not user accessible. This operation is called move-out and after it is done, write access to the C/S word of the corresponding MB is blocked (if the AEN bit in CAN_MCR register is asserted). Write access is restored in the following events:

- After the MB is transmitted and the corresponding IFLAG bit is cleared by the CPU
- FlexCAN enters in Freeze mode or Bus Off
- FlexCAN loses the bus arbitration or there is an error during the transmission

At the first opportunity window on the CAN bus, the message on the Tx SMB is transmitted according to the CAN protocol rules.

Arbitration process can be triggered in the following situations:

- During Rx and Tx frames from CAN CRC field to end of frame. CAN_CTRL2[TASD] bit value may be changed to optimize the arbitration start point.

- During CAN BusOff state from TX_ERR_CNT=124 to 128. CAN_CTRL2[TASD] bit value may be changed to optimize the arbitration start point.
- During C/S write by CPU in BusIdle. First C/S write starts arbitration process and a second C/S write during this same arbitration restarts the process. If other C/S writes are performed, Tx arbitration process is pending. If there is no arbitration winner after the arbitration process has finished, then the TX arbitration machine begins a new arbitration process. If there is a pending arbitration and BusIdle state starts then an arbitration process is triggered. In this case the first and second C/S write in BusIdle will not restart the arbitration process. It is possible that there is not enough time to finish arbitration in WaitForBusIdle state and the next state is Idle. In this case the scan is not interrupted, and it is completed during BusIdle state. During this arbitration C/S write does not cause arbitration restart.
- Arbitration winner deactivation during a valid arbitration window.
- Upon exiting Freeze mode (first bit of the WaitForBusIdle state). If there is a re-synchronization during WaitForBusIdle, the arbitration process is restarted.

Arbitration process stops in the following situations:

- All Mailboxes were scanned
- A Tx active Mailbox is found in case of Lowest Buffer feature enabled
- Arbitration winner inactivation or abort during any arbitration process
- There was not enough time to finish Tx arbitration process (for instance, when a deactivation was performed near the end of frame). In this case arbitration process is pending.
- Error or Overload flag in the bus
- Low Power or Freeze mode request in Idle state

Arbitration is considered pending as described below:

- It was not possible to finish arbitration process in time
- C/S write during arbitration if write is performed in a MB whose number is lower than the Tx arbitration pointer
- Any C/S write if there is no Tx Arbitration process in progress
- Rx Match has just updated a Rx Code to Tx Code
- Entering Busoff state

C/S write during arbitration has the following effect:

- If C/S write is performed in the arbitration winner, a new process is restarted immediately.
- If C/S write is performed in a MB whose number is higher than the Tx arbitration pointer, the ongoing arbitration process will scan this MB as normal.

47.5.3 Receive process

To be able to receive CAN frames into a Mailbox, the CPU must prepare it for reception by executing the following steps:

1. If the Mailbox is active (either Tx or Rx) inactivate the Mailbox (see Mailbox inactivation), preferably with a safe inactivation (see Transmission abort mechanism).
2. Write the ID word
3. Write the EMPTY code (0b0100) to the CODE field of the Control and Status word to activate the Mailbox.

After the MB is activated, it will be able to receive frames that match the programmed filter. At the end of a successful reception, the Mailbox is updated by the *move-in* process (see Move-in) as follows:

1. The received Data field (8 bytes at most for Classical CAN message format) is stored.
2. The received Identifier field is stored.
3. The value of the Free Running Timer at the time of the second bit of frame's Identifier field is written into the Mailbox's Time Stamp field.
4. The received SRR, IDE, RTR and DLC fields are stored.
5. The CODE field in the Control and Status word is updated (see Table 47-136 and Table 47-137 in Section Message buffer structure).
6. A status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit.

The recommended way for CPU servicing (read) the frame received in an Mailbox is using the following procedure:

1. Read the Control and Status word of that Mailbox.
2. Check if the BUSY bit is deasserted, indicating that the Mailbox is locked. Repeat step 1) while it is asserted. See Mailbox lock mechanism.
3. Read the contents of the Mailbox. Once Mailbox is locked now, its contents won't be modified by FlexCAN Move-in processes. See Move-in.

4. Acknowledge the proper flag at IFLAG registers.
5. Read the Free Running Timer. It is optional but recommended to unlock Mailbox as soon as possible and make it available for reception.

The CPU should poll for frame reception by the status flag bit for the specific Mailbox in one of the IFLAG Registers and not by the CODE field of that Mailbox. Polling the CODE field does not work because once a frame was received and the CPU services the Mailbox (by reading the C/S word followed by unlocking the Mailbox), the CODE field will not return to EMPTY. It will remain FULL, as explained in Table 47-136 . If the CPU tries to workaround this behavior by writing to the C/S word to force an EMPTY code after reading the Mailbox without a prior *safe inactivation*, a newly received frame matching the filter of that Mailbox may be lost.

CAUTION

In summary: *never do polling by reading directly the C/S word of the Mailboxes. Instead, read the IFLAG registers.*

Note that the received frame's Identifier field is always stored in the matching Mailbox, thus the contents of the ID field in an Mailbox may change if the match was due to masking. When CAN_MCR[SRXDIS] bit is asserted, FlexCAN will not store frames transmitted by itself in any MB, even if it contains a matching Rx Mailbox, and no interrupt flag or interrupt signal will be generated. Otherwise, when CAN_MCR[SRXDIS] bit is deasserted, FlexCAN can receive frames transmitted by itself if there exists a matching Rx Mailbox.

To be able to receive CAN frames through the Rx FIFO, the CPU must enable and configure the Rx FIFO during Freeze mode (see Rx FIFO). Upon receiving the Frames Available in Rx FIFO interrupt (see the description of the BUF5I bit "Frames available in Rx FIFO" bit in the CAN_IFLAG1 register), the CPU should service the received frame using the following procedure:

1. Read the Control and Status word (optional: needed only if a mask was used for IDE and RTR bits)
2. Read the ID field (optional: needed only if a mask was used)
3. Read the Data field
4. Read the CAN_RXFIR register (optional)
5. Clear the Frames Available in Rx FIFO interrupt by writing 1 to CAN_IFLAG1[BUF5I] bit (mandatory: releases the MB and allows the CPU to read the next Rx FIFO entry)

When CAN_MCR[DMA] is asserted, upon receiving a frame in FIFO, CAN_IFLAG1[BUF5I] generates a DMA request and does not generate a CPU interrupt (see Rx FIFO under DMA Operation). The CAN_IMASK1 bits in Rx FIFO region are not used.

The DMA controller must service the received frame using the following procedure:

1. Read the Control and Status word (read 0x80 address, optional)
2. Read the ID field (read 0x84 address, optional)
3. Read all Data Bytes (start read at 0x88 address, optional)
4. Read the last Data Bytes (read 0x8C address is mandatory)

47.5.4 Matching process

The matching process scans the MB memory looking for Rx MBs programmed with the same ID as the one received from the CAN bus. If the FIFO is enabled, the priority of scanning can be selected between Mailboxes and FIFO filters. The matching starts from the lowest number Message Buffer toward the higher ones. If no match is found within the first structure then the other is scanned subsequently. In the event that the FIFO is full, the matching algorithm always looks for a matching MB outside the FIFO region.

As the frame is being received, it is stored in a hidden auxiliary MB called Rx Serial Message Buffer (Rx SMB).

The matching process start point depends on the following conditions:

- If the received frame is a remote frame, the start point is the CRC field of the frame
- If the received frame is a data frame with DLC field equal to zero, the start point is the CRC field of the frame
- If the received frame is a data frame with DLC field different than zero, the start point is the DATA field of the frame

If a matching ID is found in the FIFO table or in one of the Mailboxes, the contents of the Rx SMB are transferred to the FIFO or to the matched Mailbox by the move-in process. If any CAN protocol error is detected then no match results are transferred to the FIFO or to the matched Mailbox at the end of reception.

The matching process scans all matching elements of both Rx FIFO (if enabled) and the active Rx Mailboxes (CODE is EMPTY, FULL, OVERRUN or RANSWER) in search of a successful comparison with the matching elements of the Rx SMB that is receiving the

frame on the CAN bus. The Rx SMB has the same structure of a Mailbox. The reception structures (Rx FIFO or Mailboxes) associated with the matching elements that had a successful comparison are the *matched structures*. The *matching winner* is selected at the end of the scan among those matched structures and depends on conditions described ahead. See the following table.

Table 47-143. Matching architecture

Structure	SMB[RTR]	CTRL2[RRS]	CTRL2[EACEN]	MB[IDE]	MB[RTR]	MB[ID[1]]	MB[CODE]
Mailbox	0	-	0	cmp[2]	no_cmp[3]	cmp_msk[4]	EMPTY or FULL or OVERRUN
Mailbox	0	-	1	cmp_msk	cmp_msk	cmp_msk	EMPTY or FULL or OVERRUN
Mailbox	1	0	-	cmp	no_cmp	cmp	RANSWER
Mailbox	1	1	0	cmp	no_cmp	cmp_msk	EMPTY or FULL or OVERRUN
Mailbox	1	1	1	cmp_msk	cmp_msk	cmp_msk	EMPTY or FULL or OVERRUN
FIFO[5]	-	-	-	cmp_msk	cmp_msk	cmp_msk	-

1. For Mailbox structure, If SMB[IDE] is asserted, the ID is 29 bits (ID Standard + ID Extended). If SMB[IDE] is negated, the ID is only 11 bits (ID Standard). For FIFO structure, the ID depends on IDAM.
2. cmp: Compares the Rx SMB contents with the MB contents regardless the masks.
3. no_cmp: The Rx SMB contents are not compared with the MB contents.
4. cmp_msk: Compares the Rx SMB contents with MB contents taking into account the masks.
5. SMB[IDE] and SMB[RTR] are not taken into account when IDAM is type C.

A reception structure is *free-to-receive* when any of the following conditions is satisfied:

- The CODE field of the Mailbox is EMPTY
- The CODE field of the Mailbox is either FULL or OVERRUN and it has already been serviced (the C/S word was read by the CPU and unlocked as described in Mailbox lock mechanism)
- The CODE field of the Mailbox is either FULL or OVERRUN and an inactivation (see Mailbox inactivation) is performed
- The Rx FIFO is not full

The scan order for Mailboxes and Rx FIFO is from the matching element with lowest number to the higher ones.

The matching winner search for Mailboxes is affected by the CAN_MCR[IRMQ] bit. If it is negated, the matching winner is the first matched Mailbox regardless if it is free-to-receive or not. If it is asserted, the matching winner is selected according to the priority below:

1. the first free-to-receive matched Mailbox;
2. the last non free-to-receive matched Mailbox.

It is possible to select the priority of scan between Mailboxes and Rx FIFO by the CAN_CTRL2[MRP] bit.

If the selected priority is Rx FIFO first:

- If the Rx FIFO is a matched structure and is free-to-receive, then the Rx FIFO is the matching winner regardless of the scan for Mailboxes
- Otherwise (the Rx FIFO is not a matched structure or is not free-to-receive), then the matching winner is searched among Mailboxes as described above

If the selected priority is Mailboxes first:

- If a free-to-receive matched Mailbox is found, it is the matching winner regardless of the scan for Rx FIFO
- If no matched Mailbox is found, then the matching winner is searched in the scan for the Rx FIFO
- If both conditions above are not satisfied and a non free-to-receive matched Mailbox is found, then the matching winner determination is conditioned by the CAN_MCR[IRMQ] bit:
- - If CAN_MCR[IRMQ] bit is negated, the matching winner is the first matched Mailbox
  - If CAN_MCR[IRMQ] bit is asserted, the matching winner is the Rx FIFO if it is a free-to-receive matched structure; otherwise, the matching winner is the last non free-to-receive matched Mailbox

See the following table for a summary of matching possibilities.

Table 47-144. Matching possibilities and resulting reception structures

RFEN	IRMQ	MRP	Matched in MB	Matched in FIFO	Reception structure	Description
No FIFO, only MB, match is always MB first						
0	0	X[1]	None[2]	-[3]	None	Frame lost by no match
0	0	X	Free[4]	-	FirstMB	
0	1	X	None	-	None	Frame lost by no match
0	1	X	Free	-	FirstMb	
0	1	X	NotFree	-	LastMB	Overrun
FIFO enabled, no match in FIFO is as if FIFO does not exist						
1	0	X	None	None[5]	None	Frame lost by no match
1	0	X	Free	None	FirstMB	

Table continues on the next page...

Table 47-144. Matching possibilities and resulting reception structures (continued)

RFEN	IRMQ	MRP	Matched in MB	Matched in FIFO	Reception structure	Description
1	1	X	None	None	None	Frame lost by no match
1	1	X	Free	None	FirstMb	
1	1	X	NotFree	None	LastMB	Overrun
FIFO enabled, Queue disabled						
1	0	0	X	NotFull[6]	FIFO	
1	0	0	None	Full[7]	None	Frame lost by FIFO full (FIFO Overflow)
1	0	0	Free	Full	FirstMB	
1	0	0	NotFree	Full	FirstMB	
1	0	1	None	NotFull	FIFO	
1	0	1	None	Full	None	Frame lost by FIFO full (FIFO Overflow)
1	0	1	Free	X	FirstMB	
1	0	1	NotFree	X	FirtsMb	Overrun
FIFO enabled, Queue enabled						
1	1	0	X	NotFull	FIFO	
1	1	0	None	Full	None	Frame lost by FIFO full (FIFO Overflow)
1	1	0	Free	Full	FirstMB	
1	1	0	NotFree	Full	LastMb	Overrun
1	1	1	None	NotFull	FIFO	
1	1	1	Free	X	FirstMB	
1	1	1	NotFree	NotFull	FIFO	
1	1	1	NotFree	Full	LastMb	Overrun

1. This is a don't care condition.
2. Matched in MB "None" means that the frame has not matched any MB (free-to-receive or non-free-to-receive).
3. This is a forbidden condition.
4. Matched in MB "Free" means that the frame matched at least one MB free-to-receive regardless of whether it has matched MBs non-free-to-receive.
5. Matched in FIFO "None" means that the frame has not matched any filter in FIFO. It is as if the FIFO didn't exist (CAN_CTRL2[RFEN]=0).
6. Matched in FIFO "NotFull" means that the frame has matched a FIFO filter and has empty slots to receive it.
7. Matched in FIFO "Full" means that the frame has matched a FIFO filter but couldn't store it because it has no empty slots to receive it.

If a non-safe Mailbox inactivation (see Mailbox inactivation) occurs during matching process and the Mailbox inactivated is the temporary matching winner, then the temporary matching winner is invalidated. The matching elements scan is not stopped nor

restarted, it continues normally. The consequence is that the current matching process works as if the matching elements compared before the inactivation did not exist, therefore a message may be lost.

Suppose, for example, that the FIFO is disabled, IRMQ is enabled and there are two MBs with the same ID, and FlexCAN starts receiving messages with that ID. Let us say that these MBs are the second and the fifth in the array. When the first message arrives, the matching algorithm finds the first match in MB number 2. The code of this MB is EMPTY, so the message is stored there. When the second message arrives, the matching algorithm finds MB number 2 again, but it is not "free-to-receive", so it keeps looking, finds MB number 5 and stores the message there. If yet another message with the same ID arrives, the matching algorithm finds out that there are no matching MBs that are "free-to-receive", so it decides to overwrite the last matched MB, which is number 5. In doing so, it sets the CODE field of the MB to indicate OVERRUN.

The ability to match the same ID in more than one MB can be exploited to implement a reception queue (in addition to the full featured FIFO) to allow more time for the CPU to service the MBs. By programming more than one MB with the same ID, received messages are queued into the MBs. The CPU can examine the Time Stamp field of the MBs to determine the order in which the messages arrived.

Matching to a range of IDs is possible by using ID Acceptance Masks. FlexCAN supports individual masking per MB. See the description of the Rx Individual Mask Registers (CAN_RXIMRx). During the matching algorithm, if a mask bit is asserted, then the corresponding ID bit is compared. If the mask bit is negated, the corresponding ID bit is a "don't care". Note that the Individual Mask Registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed while the module is in Freeze mode; otherwise, they are blocked by hardware.

FlexCAN also supports an alternate masking scheme with only four mask registers (CAN_RXFGMASK, CAN_RXMGMASK, CAN_RX14MASK and CAN_RX15MASK) for backwards compatibility with legacy applications. This alternate masking scheme is enabled when the IRMQ bit in the CAN_MCR Register is negated.

47.5.5 Move process

There are two types of move process: move-in and move-out.

47.5.5.1 Move-in

The move-in process is the copy of a message received by an Rx SMB to a Rx Mailbox or FIFO that has matched it. If the move destination is the Rx FIFO, attributes of the message are also copied to the CAN_RXFIR FIFO. Each Rx SMB has its own move-in process, but only one is performed at a given time as described ahead. The move-in starts only when the message held by the Rx SMB has a corresponding matching winner (see Matching process) and all of the following conditions are true:

- The CAN bus has reached or let past either:
  - The second bit of Intermission field next to the frame that carried the message that is in the Rx SMB
  - The first bit of an overload frame next to the frame that carried the message that is in the Rx SMB
- There is no ongoing matching process
- The destination Mailbox is not locked by the CPU
- There is no ongoing move-in process from another Rx SMB. If more than one move-in processes are to be started at the same time both are performed and the newest substitutes the oldest.

The term *pending move-in* is used throughout the documentation and stands for a move-to-be that still does not satisfy all of the aforementioned conditions.

The move-in is cancelled and the Rx SMB is able to receive another message if any of the following conditions is satisfied:

- The destination Mailbox is inactivated after the CAN bus has reached the first bit of Intermission field next to the frame that carried the message and its matching process has finished
- There is a previous pending move-in to the same destination Mailbox
- The Rx SMB is receiving a frame transmitted by the FlexCAN itself and the self-reception is disabled (CAN_MCR[SRXDIS] bit is asserted)
- Any CAN protocol error is detected

Note that the pending move-in is not cancelled if the module enters Freeze or Low-Power mode. It only stays on hold waiting for exiting Freeze and Low-Power mode and to be unlocked. If an MB is unlocked during Freeze mode, the move-in happens immediately.

The move-in process is the execution by the FlexCAN of the following steps:

1. Push IDHIT into the RXFIR FIFO if the message is destined to the Rx FIFO.
2. Read DATA0-3 and DATA4-7 words from the Rx SMB.
3. Write DATA0-3 and DATA4-7 words to the Rx Mailbox
4. Read the Control/Status and ID words from the Rx SMB.
5. Write Control/Status and ID words to the Rx Mailbox, and update the CODE field.

The move-in process is not atomic, in such a way that it is immediately cancelled by the inactivation of the destination Mailbox (see Mailbox inactivation) and in this case the Mailbox may be left partially updated, thus incoherent. The exception is if the move-in destination is an Rx FIFO Message Buffer, then the process cannot be cancelled.

The BUSY Bit (least significant bit of the CODE field) of the destination Message Buffer is asserted while the move-in is being performed to alert the CPU that the Message Buffer content is temporarily incoherent.

47.5.5.2 Move-out

The move-out process is the copy of the content from a Tx Mailbox to the Tx SMB when a message for transmission is available (see Section "Arbitration process"). The move-out occurs in the following conditions:

- The first bit of Intermission field
- During Bus Off state when TX Error Counter is in the 124 to 128 range
- During Bus Idle state
- During Wait For Bus Idle state

The move-out process is not atomic. Only the CPU has priority to access the memory concurrently out of Bus Idle state. In Bus Idle, the move-out has the lowest priority to the concurrent memory accesses.

47.5.6 Data coherence

In order to maintain data coherency and FlexCAN proper operation, the CPU must obey the rules described in Transmit process and Receive process.

47.5.6.1 Transmission abort mechanism

The abort mechanism provides a safe way to request the abortion of a pending transmission. A feedback mechanism is provided to inform the CPU if the transmission was aborted or if the frame could not be aborted and was transmitted instead.

Two primary conditions must be fulfilled in order to abort a transmission:

- CAN_MCR[AEN] bit must be asserted
- The first CPU action must be the writing of abort code (0b1001) into the CODE field of the Control and Status word.

Active MBs configured for transmission must be aborted first before they can be updated. If the abort code is written to a Mailbox that is currently being transmitted or to a Mailbox that was already loaded into the Tx SMB for transmission, the write operation is blocked and the transmission is not disturbed. However, the abort request is captured and kept pending until one of the following conditions is satisfied:

- The module loses the bus arbitration
- There is an error during the transmission
- The module is put into Freeze mode
- The module enters the BusOff state
- There is an overload frame

If none of the conditions above are reached, the MB is transmitted correctly, the interrupt flag is set in the IFLAG register, and an interrupt to the CPU is generated (if enabled). The abort request is automatically cleared when the interrupt flag is set. On the other hand, if one of the above conditions is reached, the frame is not transmitted; therefore, the abort code is written into the CODE field, the interrupt flag is set in the IFLAG, and an interrupt is (optionally) generated to the CPU.

If the CPU writes the abort code before the transmission begins internally, then the write operation is not blocked; therefore, the MB is updated and the interrupt flag is set. In this way the CPU just needs to read the abort code to make sure the active MB was *safely inactivated*. Although the AEN bit is asserted and the CPU wrote the abort code, in this case the MB is inactivated and not aborted, because the transmission did not start yet. One Mailbox is only aborted when the abort request is captured and kept pending until one of the previous conditions are satisfied.

The abort procedure can be summarized as follows:

- CPU checks the corresponding IFLAG and clears it, if asserted.
- CPU writes 0b1001 into the CODE field of the C/S word.
- CPU waits for the corresponding IFLAG indicating that the frame was either transmitted or aborted.
- CPU reads the CODE field to check if the frame was either transmitted (CODE=0b1000) or aborted (CODE=0b1001).
- It is necessary to clear the corresponding IFLAG in order to allow the MB to be reconfigured.

47.5.6.2 Mailbox inactivation

Inactivation is a mechanism provided to protect the Mailbox against updates by the FlexCAN internal processes, thus allowing the CPU to rely on Mailbox data coherence after having updated it, even in Normal mode.

Inactivation of transmission Mailboxes must be performed just when MCR[AEN] bit is deasserted.

If a Mailbox is inactivated, it participates in neither the arbitration process nor the matching process until it is reactivated. See Transmit process and Receive process for more detailed instructions on how to inactivate and reactivate a Mailbox.

To inactivate a Mailbox, the CPU must update its CODE field to INACTIVE (either 0b0000 or 0b1000).

Because the user is not able to synchronize the CODE field update with the FlexCAN internal processes, an inactivation can have the following consequences:

- A frame in the bus that matches the filtering of the inactivated Rx Mailbox may be lost without notice, even if there are other Mailboxes with the same filter
- A frame containing the message within the inactivated Tx Mailbox may be transmitted without setting the respective IFLAG

In order to perform a *safe inactivation* and avoid the above consequences for Tx Mailboxes, the CPU must use the Transmission Abort mechanism (see Transmission abort mechanism).

The inactivation automatically unlocks the Mailbox (see Mailbox lock mechanism).

NOTE

Message Buffers that are part of the Rx FIFO cannot be inactivated. There is no write protection on the FIFO region by FlexCAN. CPU must maintain data coherency in the FIFO region when RFEN is asserted.

47.5.6.3 Mailbox lock mechanism

Other than Mailbox inactivation, FlexCAN has another data coherence mechanism for the receive process. When the CPU reads the Control and Status word of an Rx MB with codes FULL or OVERRUN, FlexCAN assumes that the CPU wants to read the whole MB in an atomic operation, and therefore it sets an internal lock flag for that MB. The lock is released when the CPU reads the Free Running Timer (global unlock operation),

or when it reads the Control and Status word of another MB regardless of its code. A CPU write into the C/S word also unlocks the MB, but this procedure is not recommended for normal unlock use because it cancels a pending-move and potentially may lose a received message. The MB locking prevents a new frame from being written into the MB while the CPU is reading it.

NOTE

The locking mechanism applies only to Rx MBs that are not part of the FIFO and have a code different than INACTIVE (0b0000) or EMPTY[1] (0b0100). Also, Tx MBs can not be locked.

Suppose, for example, that the FIFO is disabled and the second and the fifth MBs of the array are programmed with the same ID, and FlexCAN has already received and stored messages into these two MBs. Suppose now that the CPU decides to read MB number 5 and at the same time another message with the same ID is arriving. When the CPU reads the Control and Status word of MB number 5, this MB is locked. The new message arrives and the matching algorithm finds out that there are no "free-to-receive" MBs, so it decides to override MB number 5. However, this MB is locked, so the new message can not be written there. It will remain in the Rx SMB waiting for the MB to be unlocked, and only then will be written to the MB.

If the MB is not unlocked in time and yet another new message with the same ID arrives, then the new message overwrites the one on the Rx SMB and there will be no indication of lost messages either in the CODE field of the MB or in the Error and Status Register.

While the message is being moved-in from the Rx SMB to the MB, the BUSY bit on the CODE field is asserted. If the CPU reads the Control and Status word and finds out that the BUSY bit is set, it should defer accessing the MB until the BUSY bit is negated.

Note

If the BUSY bit is asserted or if the MB is empty, then reading the Control and Status word does not lock the MB.

Inactivation takes precedence over locking. If the CPU inactivates a locked Rx MB, then its lock status is negated and the MB is marked as invalid for the current matching round. Any pending message on the Rx SMB will not be transferred anymore to the MB. An MB is unlocked when the CPU reads the Free Running Timer Register (see Section "Free Running Timer Register (CAN_TIMER)"), or the C/S word of another MB.

Lock and unlock mechanisms have the same functionality in both Normal and Freeze modes.

1. In previous FlexCAN versions, reading the C/S word locked the MB even if it was EMPTY. This behavior is maintained when the IRMQ bit is negated.

An unlock during Normal or Freeze mode results in the move-in of the pending message. However, the move-in is postponed if an unlock occurs during a low power mode (see Modes of operation), and it takes place only when the module resumes to Normal or Freeze modes.

47.5.7 Rx FIFO

The Rx FIFO is receive-only and is enabled by asserting the CAN_MCR[RFEN] bit. The reset value of this bit is zero to maintain software backward compatibility with previous versions of the module that did not have the FIFO feature.

The FIFO is 6-message deep. The memory region occupied by the FIFO structure (both Message Buffers and FIFO engine) is described in Rx FIFO structure. The CPU can read the received messages sequentially, in the order they were received, by repeatedly reading a Message Buffer structure at the output of the FIFO.

The CAN_IFLAG1[BUF5I] (Frames available in Rx FIFO) is asserted when there is at least one frame available to be read from the FIFO. An interrupt is generated if it is enabled by the corresponding mask bit. Upon receiving the interrupt, the CPU can read the message (accessing the output of the FIFO as a Message Buffer) and the CAN_RXFIR register and then clear the interrupt. If there are more messages in the FIFO the act of clearing the interrupt updates the output of the FIFO with the next message and update the CAN_RXFIR with the attributes of that message, reissuing the interrupt to the CPU. Otherwise, the flag remains negated. The output of the FIFO is only valid whilst the CAN_IFLAG1[BUF5I] is asserted.

The CAN_IFLAG1[BUF6I] (Rx FIFO Warning) is asserted when the number of unread messages within the Rx FIFO is increased to 5 from 4 due to the reception of a new one, meaning that the Rx FIFO is almost full. The flag remains asserted until the CPU clears it.

The CAN_IFLAG1[BUF7I] (Rx FIFO Overflow) is asserted when an incoming message was lost because the Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a Mailbox. The flag remains asserted until the CPU clears it.

Clearing one of those three flags does not affect the state of the other two.

An interrupt is generated if an IFLAG bit is asserted and the corresponding mask bit is asserted too.

A powerful filtering scheme is provided to accept only frames intended for the target application, reducing the interrupt servicing work load. The filtering criteria is specified by programming a table of up to 128 32-bit registers, according to CAN_CTRL2[RFFN] setting, that can be configured to one of the following formats (see also Rx FIFO structure):

- Format A: 128 IDAFs (extended or standard IDs including IDE and RTR)
- Format B: 256 IDAFs (standard IDs or extended 14-bit ID slices including IDE and RTR)
- Format C: 512 IDAFs (standard or extended 8-bit ID slices)

Note

A chosen format is applied to all entries of the filter table. It is not possible to mix formats within the table.

Every frame available in the FIFO has a corresponding IDHIT (Identifier Acceptance Filter Hit Indicator) that can read in the IDHIT field from C/S word, as shown in the Rx FIFO Structure description. Another way the CPU can obtain this information is by accessing the CAN_RXFIR register. The CAN_RXFIR[IDHIT] field refers to the message at the output of the FIFO and is valid while the CAN_IFLAG1[BUF5I] flag is asserted. The CAN_RXFIR register must be read only before clearing the flag, which guarantees that the information refers to the correct frame within the FIFO.

Up to 32 elements of the filter table are individually affected by the Individual Mask Registers (CAN_RXIMRx), according to the setting of CAN_CTRL2[RFFN], allowing very powerful filtering criteria to be defined. If the CAN_MCR[IRMQ] bit is negated, then the FIFO filter table is affected by CAN_RXFGMASK.

47.5.7.1 Rx FIFO under DMA Operation

The receive-only FIFO can support DMA, this feature is enabled by asserting both the CAN_MCR[RFEN] and CAN_MCR[DMA] bits. The reset value of CAN_MCR[DMA] bit is zero to maintain backward compatibility with previous versions of the module that did not have the DMA feature.

The DMA controller can read the received message by reading a Message Buffer structure at the FIFO output port at the 0x80-0x8C address range.

When CAN_MCR[DMA] is asserted the CPU must not access the FIFO output port address range. Before enabling the CAN_MCR[DMA], the CPU must service the IFLAGs asserted in the Rx FIFO region. Otherwise, these IFLAGs may show that the FIFO has data to be serviced, and mistakenly generate a DMA request. Before disabling the CAN_MCR[DMA], the CPU must perform a clear FIFO operation.

The CAN_IFLAG1[BUF5I] (Frames available in Rx FIFO) is asserted when there is at least one frame available to be read from the FIFO, consequently a DMA request is generated simultaneously. Upon receiving the request, the DMA controller can read the message (accessing the output of the FIFO as a Message Buffer). The DMA reading process must end by reading address 0x8C, which clears the CAN_IFLAG1[BUF5I] and updates both the FIFO output with the next message (if FIFO is not empty) and the CAN_RXFIR register with the attributes of the new message. If there are more messages stored in the FIFO, the CAN_IFLAG1[BUF5I] will be re-asserted and another DMA request is issued. Otherwise, the flag remains negated.

NOTE

CAN_RXFIR register contents cannot be read after DMA completes the FIFO read. The IDHIT information is also available in the C/S word at address 0x080 (see Rx FIFO structure.

The CAN_IFLAG1[BUF6I] and CAN_IFLAG1[BUF7I] are not used when the DMA feature is enabled.

When FlexCAN is working with DMA, the CPU does not receive any Rx FIFO interruption and must not clear the related IFLAGs. In addition, the related IMASKs are not used to mask the generation of DMA requests.

47.5.7.2 Clear FIFO Operation

When CAN_MCR[RFEN] is asserted, the clear FIFO operation is a feature used to empty FIFO contents. With CAN_MCR[RFEN] asserted the Clear FIFO occurs when the CPU writes 1 in CAN_IFLAG1[BUF0I]. This operation can only be performed in Freeze Mode and is blocked by hardware in other modes. This operation does not clear the FIFO IFLAGs, consequently the CPU must service all FIFO IFLAGs before execute the clear FIFO task.

When Rx FIFO is working with DMA, the clear FIFO operation clears the CAN_IFLAG1[BUF5I] and the DMA request is canceled.

CAUTION

Clear FIFO operation does not clear IFLAGs, except when CAN_MCR[DMA] is asserted, in this case only the CAN_IFLAG1[BUF5I] is cleared.

47.5.8 CAN protocol related features

This section describes the CAN protocol related features.

47.5.8.1 Remote frames

Remote frame is a special kind of frame. The user can program a mailbox to be a Remote Request Frame by configuring the mailbox as Transmit with the RTR bit set to '1'. After the remote request frame is transmitted successfully, the mailbox becomes a Receive Message Buffer, with the same ID as before.

When a remote request frame is received by FlexCAN, it can be treated in three ways, depending on Remote Request Storing (CTRL2[RRS]) and Rx FIFO Enable (MCR[RFEN]) bits:

- If RRS is negated the frame's ID is compared to the IDs of the Transmit Message Buffers with the CODE field 0b1010. If there is a matching ID, then this mailbox frame will be transmitted. Note that if the matching mailbox has the RTR bit set, then FlexCAN will transmit a remote frame as a response. The received remote request frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame should match. In the case that a remote request frame is received and matches a mailbox, this message buffer immediately enters the internal arbitration process, but is considered as a normal Tx mailbox, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.
- If RRS is asserted the frame's ID is compared to the IDs of the receive mailboxes with the CODE field 0b0100, 0b0010 or 0b0110. If there is a matching ID, then this mailbox will store the remote frame in the same fashion of a data frame. No automatic remote response frame will be generated. The mask registers are used in the matching process.
- If RFEN is asserted FlexCAN will not generate an automatic response for remote request frames that match the FIFO filtering criteria. If the remote frame matches one of the target IDs, it will be stored in the FIFO and presented to the CPU. Note that for

filtering formats A and B, it is possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted (if they match the ID). Remote Request Frames are considered as normal frames, and generate a FIFO overflow when a successful reception occurs and the FIFO is already full.

47.5.8.2 Overload frames

FlexCAN does transmit overload frames due to detection of following conditions on CAN bus:

- Detection of a dominant bit in the first/second bit of Intermission
- Detection of a dominant bit at the 7th bit (last) of End of Frame field (Rx frames)
- Detection of a dominant bit at the 8th bit (last) of Error Frame Delimiter or Overload Frame Delimiter

47.5.8.3 Time stamp

The value of the Free Running Timer is sampled at the beginning of the Identifier field on the CAN bus, and is stored at the end of "move-in" in the TIME STAMP field, providing network behavior with respect to time.

The Free Running Timer is clocked by the FlexCAN bit-clock, which defines the baud rate on the CAN bus. During a message transmission/reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate.

The Free Running Timer is not incremented during Disable, Stop, and Freeze modes. It can be reset upon a specific frame reception, enabling network time synchronization. See the TSYN description in Control 1 Register (CAN_CTRL1).

47.5.8.4 Protocol timing

The following figure shows the structure of the clock generation circuitry that feeds the CAN Protocol Engine (PE) submodule.

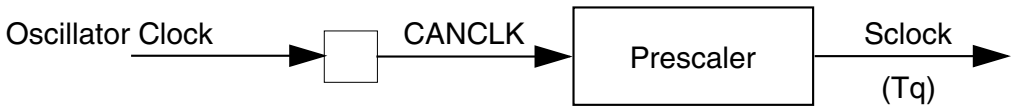


Figure 47-128. CAN engine clocking scheme

The FlexCAN module supports a variety of means to setup bit timing parameters that are required by the CAN protocol. The Control 1 Register (CAN_CTRL1) has various fields used to control bit timing parameters: PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW.

The CAN Bit Timing register (CAN_CBT) extends the range of the CAN bit timing variables in CAN_CTRL1.

The PRESDIV field (as well as its extended range EPRESDIV) defines the Prescaler Value (see the equation below) that generates the Serial Clock (Sclock), whose period defines the 'time quantum' used to compose the CAN waveform. A time quantum (Tq) is the atomic unit of time handled by the CAN engine.

$$Tq = \frac{(\text{PRESDIV} + 1)}{f_{\text{CANCLK}}}$$

The bit rate, which defines the rate the CAN message is either received or transmitted, is given by the formula:

$$\text{CAN Bit Time} = (\text{Number of Time Quanta in 1 bit time}) * Tq$$

$$\text{Bit Rate} = \frac{1}{\text{CAN Bit Time}}$$

A bit time is subdivided into three segments[2] (see Figure 47-129 and Table 47-145):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section

2. For further explanation of the underlying concepts, see ISO 11898-1. See also the CAN 2.0A/B protocol specification for bit timing.

- Time Segment 1: This segment includes the Propagation Segment and the Phase Segment 1 of the CAN standard. It can be programmed by setting the PROPSEG and the PSEG1 fields of the CAN_CTRL1 Register so that their sum (plus 2) is in the range of 4 to 16 time quanta. When CAN_CBT[BTF] bit is asserted, FlexCAN uses EPROPSEG and EPSEG1 fields from CAN_CBT register so that their sum (plus 2) is in the range of 2 to 96 time quanta.
- Time Segment 2: This segment represents the Phase Segment 2 of the CAN standard. It can be programmed by setting the PSEG2 field of the CAN_CTRL1 Register (plus 1) to be 2 to 8 time quanta long. When CAN_CBT[BTF] bit is asserted, FlexCAN uses EPSEG2 fields of CAN_CBT register so that its value (plus 1) is in the range of 2 to 32 time quanta. The Time Segnment 2 cannot be smaller than the Information Processing Time (IPT), which value is 2 time quanta in FlexCAN.

NOTE

The bit time defined by the above time segments must not be smaller than 5 time quanta. For bit time calculations, use an Information Processing Time (IPT) of 2, which is the value implemented in the FlexCAN module.

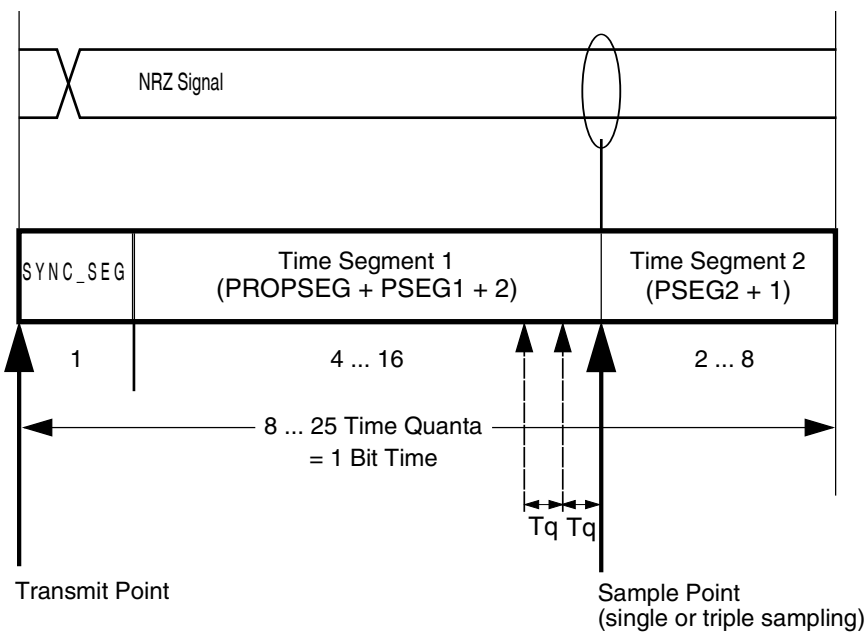


Figure 47-129. Segments within the bit time (example using CAN_CTRL1 bit timing variables for Classical CAN format)

Table 47-145. Time segment syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the bus during this period.
TSEG1	Corresponds to the sum of PROPSEG and PSEG1.
TSEG2	Corresponds to the PSEG2 value.
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

The following table gives some examples of the CAN compliant segment settings for Classical CAN format (Bosch CAN 2.0B) messages.

Table 47-146. Bosch CAN 2.0B standard compliant bit time segment settings

Time segment 1	Time segment 2	Re-synchronization jump width
5 .. 10	2	1 .. 2
4 .. 11	3	1 .. 3
5 .. 12	4	1 .. 4
6 .. 13	5	1 .. 4
7 .. 14	6	1 .. 4
8 .. 15	7	1 .. 4
9 .. 16	8	1 .. 4

Note

The user must ensure the bit time settings are in compliance with the CAN Protocol standard (ISO 11898-1).

Whenever CAN bit is used as a measure of time duration (e.g. estimating the occurrence of a CAN bit event in a message), the number of peripheral clocks in one CAN bit (NumClkBit) can be calculated as:

$$\text{NumClkBit} = \frac{f_{SYS}}{f_{CANCLK}} \text{ X } (\text{PRESDIV} + 1) \text{ X } (\text{PROPSEG} + \text{PSEG1} + \text{PSEG2} + 4)$$

where:

- NumClkBit is the number of peripheral clocks in one CAN bit;
- f_{CANCLK} is the Protocol Engine (PE) Clock (see Figure "CAN Engine Clocking Scheme"), in Hz;
- f_{SYS} is the frequency of operation of the system (CHI) clock, in Hz;

- PSEG1 is the value in CAN_CTRL1[PSEG1] field;
- PSEG2 is the value in CAN_CTRL1[PSEG2] field;
- PROPSEG is the value in CAN_CTRL1[PROPSEG] field;
- PRESDIV is the value in CAN_CTRL1[PRESDIV] field.

The formula above is also applicable to the alternative CAN bit timing variables described in the CAN Bit Timing Register (CAN_CBT).

For example, 180 CAN bits = (180 x NumClkBit) peripheral clock periods.

47.5.8.5 Arbitration and matching timing

During normal reception and transmission, the matching, arbitration, move-in and move-out processes are executed during certain time windows inside the CAN frame, as shown in the following figures.

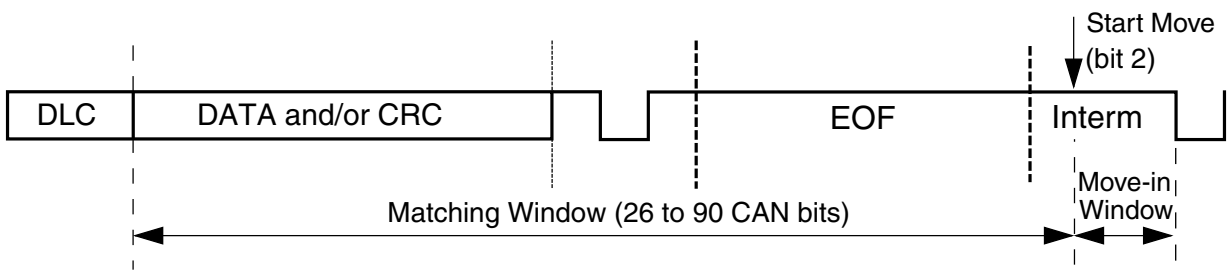


Figure 47-130. Matching and move-in time windows

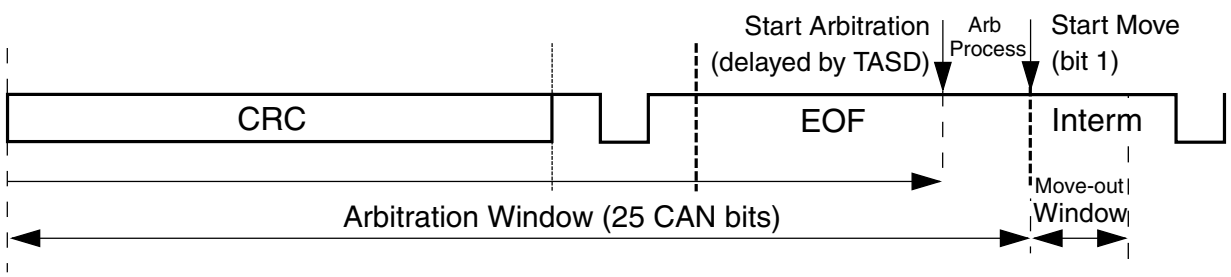


Figure 47-131. Arbitration and move-out time windows

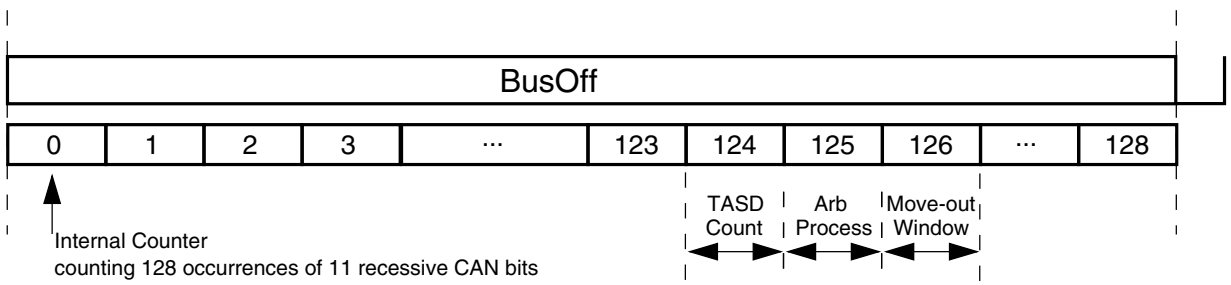


Figure 47-132. Arbitration at the end of bus off and move-out time windows

NOTE

In the preceding figures, the matching and arbitration timing does not take into account the delay caused by the concurrent memory access due to the CPU or other internal FlexCAN sub-blocks.

47.5.8.6 Tx Arbitration start delay

The Tx Arbitration Start Delay (TASD) bit field in Control 2 register (CAN_CTRL2[TASD]) is a variable that indicates the number of CAN bits used by FlexCAN to delay the Tx Arbitration process start point from the first bit of CRC field of the current frame. This variable can be written only in Freeze mode because it is blocked by hardware in other modes.

The transmission performance is impacted by the ability of the CPU to reconfigure Message Buffers (MBs) for transmission after the end of the internal Arbitration process, where FlexCAN finds the winner MB for transmission (see Arbitration process). If the Arbitration ends too early before the first bit of Intermission field, then there is a chance that the CPU reconfigures some Tx MBs and the winner MB is no longer the best candidate to be transmitted.

TASD is useful to optimize the transmission performance by defining the Arbitration start point, as shown in the next figure, based on factors such as:

- The peripheral-to-oscillator clock ratio
- CAN bit timing variables that determine the CAN bit rate
- The number of Message Buffers (MBs) in use by the Matching and Arbitration processes.

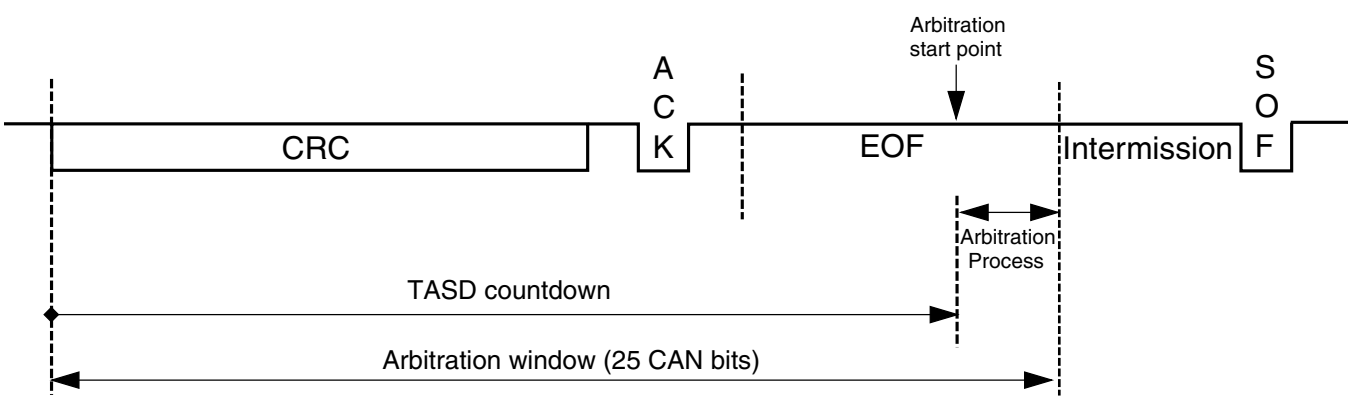


Figure 47-133. Optimal Tx Arbitration start point

The duration of an Arbitration process, in terms of CAN bits, is directly proportional to the number of available MBs and to the CAN bit rate, and inversely proportional to the peripheral clock frequency.

The optimal Arbitration timing is that in which the last MB is scanned right before the first bit of the Intermission field of a CAN frame. For instance, if there are few MBs and the peripheral/oscillator clock ratio is high and the CAN baud rate is low, then the Arbitration can be placed closer to the frame's end, adding more delay to its start point, and vice-versa.

If TASD is set to 0 then the Arbitration start is not delayed and more time is reserved for Arbitration. On the other hand, if TASD is close to 24 then the CPU can configure a Tx MB later and less time is reserved for Arbitration. If too little time is reserved for Arbitration the FlexCAN may be not be able to find a winner MB in time to be transmitted with the best chance to win the bus arbitration against external nodes on the CAN bus.

The optimal TASD value can be calculated as follows:

$$TASD = 25 - \left(\frac{f_{CANCLK} \times [\, MAXMB + 3 - (RFEN \times 8) - (RFEN \times RFFN \times 2)\,] \times 2}{f_{SYS} \times [1 + (PSEG1 + 1) + (PSEG2 + 1) + (PROPSEG + 1)\,] \times (PRESDIV + 1)} \right)$$

where:

- MAXMB is the value in CAN_CTRL1[MAXMB] field
- f_{CANCLK} is the oscillator clock, in Hz
- f_{SYS} is the peripheral clock, in Hz
- RFEN is the value in CAN_CTRL1[RFEN] bit
- RFFN is the value in CAN_CTRL2[RFFN] field
- PSEG1 is the value in CAN_CTRL1[PSEG1] field
- PSEG2 is the value in CAN_CTRL1[PSEG2] field
- PROPSEG is the value in CAN_CTRL1[PROPSEG] field
- PRESDIV is the value in CAN_CTRL1[PRESDIV] field

See also Protocol timing for more details.

47.5.9 Clock domains and restrictions

The FlexCAN module has two clock domains asynchronous to each other:

- The Bus Domain feeds the Control Host Interface (CHI) submodule and is derived from the peripheral clock.
- The Oscillator Domain feeds the CAN Protocol Engine (PE) submodule and is derived directly from a crystal oscillator clock, so that very low jitter performance can be achieved on the CAN bus.

When CAN_CTRL1[CLKSRC] bit is set, synchronous operation occurs because both domains are connected to the peripheral clock (creating a 1:1 ratio between the peripheral and oscillator clocks).

When the two domains are connected to clocks with different frequencies and/or phases, there are restrictions on the frequency relationship between the two clock domains. In the case of asynchronous operation, the Bus Domain clock frequency must always be greater than the Oscillator Domain clock frequency.

NOTE

Asynchronous operation with a 1:1 ratio between peripheral and oscillator clocks is not allowed.

When doing matching and arbitration, FlexCAN needs to scan the whole Message Buffer memory during the time slot of one CAN frame, comprised of a number of CAN bits. In order to have sufficient time to do that, the following requirements must be observed:

- The peripheral clock frequency can not be smaller than the oscillator clock frequency
- There must be a minimum number of peripheral clocks per CAN bit, as specified in the table shown below

Table 47-147. Minimum number of peripheral clocks per CAN bit

Number of Mailboxes	Value of CAN_MCR[RFEN] bit	Minimum number of peripheral clocks per CAN bit
16	0	16
32	0	16
64	0	25
96	0	37
16	1	16
32	1	17
64	1	30
96	1	42

The minimum number of peripheral clocks per CAN bit specified in the preceding table determines the minimum peripheral clock frequency for a given number of Mailboxes and for an expected CAN bit rate. The CAN bit rate depends on the number of time quanta in a CAN bit, that can be defined by adjusting one or more of the bit timing values

contained in either the Control 1 Register (CAN_CTRL1) or CAN Bit Time register (CAN_CBT). The time quantum (Tq) is defined in Protocol timing. The minimum number of time quanta per CAN bit must be 8; therefore, the oscillator clock frequency should be at least 8 times the CAN bit rate.

47.5.10 Modes of operation details

The FlexCAN module has functional modes and low-power modes. See Modes of operation for an introductory description of all the modes of operation. The following sub-sections contain functional details on Freeze mode and the low-power modes.

CAUTION

"Permanent Dominant" failure on CAN Bus line is not supported by FlexCAN. If a Low-Power request or Freeze mode request is done during a "Permanent Dominant", the corresponding acknowledge can never be asserted.

47.5.10.1 Freeze mode

This mode is requested either by the CPU through the assertion of the HALT bit in the CAN_MCR Register or when the chip is put into Debug mode. In both cases it is also necessary that the FRZ bit is asserted in the CAN_MCR Register and the module is not in a low-power mode. This mode is also requested by FlexCAN through the automatic assertion of both HALT and FRZ bits when CAN_MECR[NCEFAFRZ] bit is set and a non-correctable error is detected in a memory read access performed by FlexCAN internal processes (see Response to Errors).

The acknowledgement is obtained through the assertion by the FlexCAN of FRZ_ACK bit in the same register. The CPU must only consider the FlexCAN in Freeze mode when both request and acknowledgement conditions are satisfied.

When Freeze mode is requested, FlexCAN does the following:

- Waits to be in either Intermission, Passive Error, Bus Off or Idle state
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in does not prevent going to Freeze mode.
- Ignores the Rx input pin and drives the Tx pin as recessive
- Stops the prescaler, thus halting all CAN protocol activities

- Grants write access to the Error Counters Register, which is read-only in other modes
- Sets the NOT_RDY and FRZ_ACK bits in CAN_MCR

After requesting Freeze mode, the user must wait for the FRZ_ACK bit to be asserted in CAN_MCR before executing any other action, otherwise FlexCAN may operate in an unpredictable way. In Freeze mode, all memory mapped registers are accessible.

Exiting Freeze mode is done in one of the following ways:

- CPU negates the FRZ bit in the CAN_MCR Register
- The chip is removed from Debug Mode and/or the HALT bit is negated

The FRZ_ACK bit is negated after the protocol engine recognizes the negation of the freeze request. When out of Freeze mode, FlexCAN tries to re-synchronize to the CAN bus by waiting for 11 consecutive recessive bits.

47.5.10.2 Module Disable mode

This low power mode is normally used to temporarily disable a complete FlexCAN block, with no power consumption. It is requested by the CPU through the assertion of the CAN_MCR[MDIS] bit, and the acknowledgement is obtained through the assertion by the FlexCAN of the CAN_MCR[LPMACK] bit. The CPU must only consider the FlexCAN in Disable mode when both request and acknowledgement conditions are satisfied.

If the module is disabled during Freeze mode, it requests to disable the clocks to the PE and CHI sub-modules, sets the LPMACK bit and negates the FRZACK bit.

If the module is disabled during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and then checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in is not taken into account.
- Ignores its Rx input pin and drives its Tx pin as recessive
- Shuts down the clocks to the PE and CHI sub-modules
- Sets the NOTRDY and LPMACK bits in CAN_MCR

The Bus Interface Unit continues to operate, enabling the CPU to access memory mapped registers, except the Rx Mailboxes Global Mask Registers, the Rx Buffer 14 Mask Register, the Rx Buffer 15 Mask Register, the Rx FIFO Global Mask Register. The Rx FIFO Information Register, the Message Buffers, the Rx Individual Mask Registers, and the reserved words within RAM may not be accessed when the module is in Disable Mode. Exiting from this mode is done by negating the MDIS bit by the CPU, which causes the FlexCAN to request to resume the clocks and negate the LPMACK bit after the CAN protocol engine recognizes the negation of disable mode requested by the CPU.

47.5.10.3 Stop mode

This is a system low-power mode in which all chip clocks can be stopped for maximum power savings. The Stop mode is globally requested by the CPU and the acknowledgement is obtained through the assertion by the FlexCAN of a Stop Acknowledgement signal. The CPU must only consider the FlexCAN in Stop mode when both request and acknowledgement conditions are satisfied.

If FlexCAN receives the global Stop mode request during Freeze mode, it sets the LPMACK bit, negates the FRZACK bit and then sends the Stop Acknowledge signal to the CPU, in order to shut down the clocks globally.

If Stop mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in is not taken into account.
- Ignores its Rx input pin and drives its Tx pin as recessive
- Sets the NOTRDY and LPMACK bits in CAN_MCR
- Sends a Stop Acknowledge signal to the CPU, so that it can shut down the clocks globally

Stop mode is exited when the CPU resumes the clocks and removes the Stop Mode request.

After the CAN protocol engine recognizes the negation of the Stop mode request, the FlexCAN negates the LPMACK bit. FlexCAN will then wait for 11 consecutive recessive bits to synchronize to the CAN bus.

47.5.11 Interrupts

The module has many interrupt sources: interrupts due to message buffers and interrupts due to the ORed interrupts from MBs, Bus Off, Bus Off Done, Error, Tx Warning, and Rx Warning.

Each one of the message buffers can be an interrupt source, if its corresponding IMASK bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception. Each of the buffers has an assigned flag bit in the CAN_IFLAG registers. The bit is set when the corresponding buffer completes a successful transfer and is cleared when the CPU writes it to 1 (unless another interrupt is generated at the same time).

Note

It must be guaranteed that the CPU clears only the bit causing the current interrupt. For this reason, bit manipulation instructions (BSET) must not be used to clear interrupt flags. These instructions may cause accidental clearing of interrupt flags which are set after entering the current interrupt service routine.

If the Rx FIFO is enabled (CAN_MCR[RFEN] = 1) and DMA is disabled (CAN_MCR[DMA] = 0), the interrupts corresponding to MBs 0 to 7 have different meanings. Bit 7 of the CAN_IFLAG1 register becomes the "FIFO Overflow" flag; bit 6 becomes the "FIFO Warning" flag, bit 5 becomes the "Frames Available in FIFO" flag and bits 4-0 are unused. See the description of the Interrupt Flags 1 Register (CAN_IFLAG1) for more information.

If both Rx FIFO and DMA are enabled (CAN_MCR[RFEN] and CAN_MCR[DMA] = 1) the FlexCAN does not generate any FIFO interrupt. The bit 5 of the CAN_IFLAG1 register still indicates "Frames Available in FIFO" and generates a DMA request. The bits 7, 6, 4-0 are unused.

For a combined interrupt where multiple MB interrupt sources are OR'd together, the interrupt is generated when any of the associated MBs (or FIFO, if applicable) generates an interrupt. In this case, the CPU must read the CAN_IFLAG registers to determine which MB or FIFO source caused the interrupt.

The interrupt sources for Bus Off, Bus Off Done, Error, Tx Warning and Rx Warning generate interrupts like the MB interrupt sources, and can be read from CAN_ESR1 register. The Bus Off, Error, Tx Warning, and Rx Warning interrupt mask bits are located in the CAN_CTRL1 Register.

47.5.12 Bus interface

The CPU access to FlexCAN registers are subject to the following rules:

- Unrestricted read and write access to supervisor registers (registers identified with S/U in Table "Module Memory Map" in Supervisor Mode or with S only) results in access error.
- Read and write access to implemented reserved address space results in access error.
- Write access to positions whose bits are all currently read-only results in access error. If at least one of the bits is not read-only then no access error is issued. Write permission to positions or some of their bits can change depending on the mode of operation or transitory state. Refer to register and bit descriptions for details.
- Read and write access to unimplemented address space results in access error.
- Read and write access to RAM located positions during Low Power Mode results in access error.
- If MAXMB in CAN MCR register is programmed with a value smaller than the available number of MBs, then the unused memory space can be used as general purpose RAM space. Note that reserved words within RAM cannot be used. As an example, suppose FlexCAN's RAM can support up to 16 MBs, CAN_CTRL2[RFFN] is 0x0, and CAN_MCR[MAXMB] is programmed with zero. The maximum number of MBs in this case becomes one. The RAM starts at 0x0080, and the space from 0x0080 to 0x008F is used by the one MB. The memory space from 0x0090 to 0x017F is available. The space between 0x0180 and 0x087F is reserved. The space from 0x0880 to 0x0883 is used by the one Individual Mask and the available memory in the Mask Registers space would be from 0x0884 to 0x08BF. From 0x08C0 through 0x09DF there are reserved words for internal use which cannot be used as general purpose RAM. As a general rule, free memory space for general purpose depends only on MAXMB.

47.5.13 Detection and Correction of Memory Errors

FlexCAN supports detection and correction of errors in memory read accesses. Each byte of FlexCAN memory is associated to 5 parity bits that ensure a Hamming distance of 4. The error correction mechanism ensures that in this 13-bit word, errors in one bit can be corrected (correctable errors) and errors in 2 bits can be detected but not corrected (non-correctable errors). Errors in more than 2 bits may not be detected. In case of non-

correctable errors, the corrupted data is not changed by the error correction logic. When a read access is performed, the parity bits are used to calculate a syndrome, which indicates the error in each byte.

FlexCAN detects a non-correctable error in the event either an all-zeros or an all-ones read occurs. See CAN_RERRSYNR register description.

Memory errors are indicated to the Host through status register (Error Status Register (CAN_ERRSR)) and bus transfer errors, and reported through report registers (Error Report Address Register (CAN_RERRAR), Error Report Data Register (CAN_RERRDR) and Error Report Syndrome Register (CAN_RERRSYNR)).

The error detection and correction mechanism can be activated or not, controlled by the ECCDIS bit in Memory Error Control Register (CAN_MECR). When disabled, updates on indications and reporting registers are stopped, but the parity bits are still calculated and written along with data in memory write operations to ensure that memory has consistent parity bits associated to the data.

NOTE

All FlexCAN memory must be initialized before starting its operation in order to have the parity bits in memory properly updated. The WRMFRZ bit in Control 2 Register (CAN_CTRL2) grants write access to all memory positions that require initialization, ranging from 0x080 to 0xADF. The RXMGMASK, RX14MASK, RX15MASK, and RXFGMASK registers need to be initialized as well. The CAN_MCR[RFEN] bit must not be set during memory initialization.

To avoid accidentally changing the critical error correction configuration, this protocol must be followed to enable the update of the Memory Error Control Register (CAN_MECR):

1. By default, ECRWRE bit in Control 2 Register (CAN_CTRL2) is 0 and ECRWRDIS bit in Memory Error Control Register (CAN_MECR) is 1.
2. Set ECRWRE bit in Control 2 Register (CAN_CTRL2).
3. Clear ECRWRDIS bit in Memory Error Control Register (CAN_MECR).
4. All writes to Memory Error Control Register (CAN_MECR) must keep ECRWRDIS cleared.
5. After configuration is done, lock the Memory Error Control Register (CAN_MECR) by either setting ECRWRDIS or clearing ECRWRE.

47.5.13.1 Sources of the Memory Access

The FlexCAN memory can be accessed by two major sources (or requestors):

- by Host (CPU): the largest word accessed is 32-bit wide
- by FlexCAN internals processes (Rx Matching, Tx Arbitration, Move-in on reception, Move-out on transmission): the largest word accessed is 64-bit wide

The way that non-correctable errors are indicated and reported depends on the source of access.

47.5.13.2 Error Indication

Memory errors are indicated by flags HANCEIF, FANCEIF, CEIF in the Error Status Register (CAN_ERRSR). Non-correctable errors detected in memory reads requested by Host are indicated separatedly than the ones detected in requests by FlexCAN internal processes. FlexCAN makes no distinction of the source of the access when correctable errors are detected. There are 3 independent flags for these 3 cases. If both non-correctable and correctable errors are found in different bytes in the same read operation, both flags are set.

A non-correctable error detected in Host access is also indicated as a bus transfer error. A bus wait request may be asserted to extend the memory transaction to the moment the report registers are updated. This indication cannot be masked.

Each indication flag has one Overrun flag in Error Status Register (CAN_ERRSR). The Overrun flags do not request interrupts. Overrun flags for non-correctable errors indicate that other errors of the same nature were detected after current error being treated, while overrun flags for correctable errors indicate that other errors of the same nature were detected before the current error being treated. This is the recommended handling sequence for error indication:

1. Get error report information from report registers
2. Use this information to take proper measures in the application
3. Clear the HANCEIF, FANCEIF, CEIF flags
4. If the Overrun flag is active:
    a. Alert application that at least one error could not be handled
    b. Clear the Overrun flag

The FlexCAN internal processes can access memory in transactions larger than 32-bits. For the indication, this kind of access is considered a consecutive sequence of 32-bit accesses. If errors are found in 2 or more 32-bit words the Interrupt and Overrun flags are set simultaneously.

47.5.13.3 Error Reporting

The report registers Error Report Address Register (CAN_RERRAR), Error Report Data Register (CAN_RERRDR) and Error Report Syndrome Register (CAN_RERRSYNR) provide detailed information about the address read, raw data and syndrome read with error and indicated by the flags described in Error Indication. The address, data and syndrome registers are updated simultaneously along with the error flags, according to these rules:

1. If any of the 2 non-correctable error flags is currently set, the report registers are not updated (the previous non-correctable error reporting is preserved)
2. Otherwise (either no error flag is currently set or only the correctable error flag is currently set), the report registers are updated according to the new error; or according to the most severe of new errors if non-correctable and correctable errors are simultaneously detected

Reporting of errors detected in accesses larger than 32-bit follows the rules described in Error Indication and in the Error Report Address Register (CAN_RERRAR) description.

The address reported in CAN_RERRAR and defined in CAN_ERRIAR are not the same listed in the module memory map. The relation between the reported addresses and the respective ones in the module memory map is shown in the Error Injection Address Register (CAN_ERRIAR) description.

Addresses reported when reading memory portions organized as FIFOs, such as the Rx FIFO Structure and the Rx FIFO Information Register (CAN_RXFIR), refer to the address of the specific entry accessed in the FIFO, not to the FIFO base address.

To assure coherence of the error report registers it is necessary to turn off the report update by setting the CAN_MECR[RERRDIS] bit before reading the report registers.

47.5.13.4 Response to Errors

Correctable errors have no consequence on FlexCAN operation because affected data is corrected before its use by the host or FlexCAN internal processes.

For host-initiated reads, a non-correctable error may affect the host, but does not affect FlexCAN operation.

Non-correctable errors detected on memory reads requested by the FlexCAN internal processes may result in incorrect operation depending on the state of the NCEFAFRZ bit in CAN_MECR register, as follows:

- During reception (either Matching or Move-in processes), when a non-correctable error occurs, an incorrect destination may be selected to store the incoming frame, a corrupted frame may be stored in the correct destination, or both. In case NCEFAFRZ is set, FlexCAN stops operation automatically and enters in Freeze mode to prevent corrupted data from being treated as valid by FlexCAN internal processes. When NCEFAFRZ is negated, FlexCAN continues working and a corrupted frame is received.
- During Arbitration process, when a non-correctable error occurs, either a non-highest priority Tx Message Buffer may be mistakenly selected for transmission or its data may be corrupted. In case NCEFAFRZ is set, FlexCAN stops operation automatically and enters in Freeze mode before starting the Move-out. When NCEFAFRZ is negated, FlexCAN proceeds to Move-out with a corrupted frame that will be transmitted on the CAN bus.
- During Move-out process, when a non-correctable error occurs, a corrupted frame is copied from the selected Tx MB that won the Arbitration to the Tx SMB (hardware buffer not located in RAM) for transmission. In case NCEFAFRZ is set, FlexCAN stops operation automatically and enters in Freeze mode before starting the transmission. When NCEFAFRZ is negated, the corrupted frame is transferred from the Tx SMB to the Protocol Engine (PE) sub-block and is transmitted on the CAN bus.

When NCEFAFRZ is set and FlexCAN enters in Freeze Mode, only the CPU can cause FlexCAN to exit Freeze mode and resume Normal Mode. The assertion of the CAN_MECR[NCEFAFRZ] bit is the only way to prevent corrupted frames from being transmitted on the CAN bus up to the Move-out internal process.

The error report registers can provide information to the application for a customized handling of these situations.

47.5.13.5 Error Injection

The error injection registers CAN_ERRIAR, CAN_ERRIDPR, and CAN_ERRIPPR are used to inject errors in memory reads in order to force errors and consequently update the indication and reporting registers. The relation between the error injection addresses and the respective ones in the module memory map is shown in the CAN_ERRIAR description.

The injection is done by flipping the data and parity bits correspondent to the bits in 1 in CAN_ERRIDPR and CAN_ERRIPPR. Injection can be selected specifically for memory accesses requested by Host or by FlexCAN internal processes.

In case of accesses larger than 32-bits, the EXTERRIE bit in Memory Error Control Register (CAN_MECR) extends the injection pattern, replicating it in 32-bit words to fill the width of the access.

NOTE

It is very unlikely, but error injection may correct a bit with error. This will not raise the error flags and reports as expected.

To ensure coherence among error injection registers and avoid spurious error injections, the HAERRIE and FAERRIE bits in Memory Error Control Register (CAN_MECR) must be cleared while configuring the memory injection registers.

47.6 Initialization/application information

This section provide instructions for initializing the FlexCAN module.

47.6.1 FlexCAN initialization sequence

The FlexCAN module may be reset in three ways:

- Chip level hard reset, which resets all memory mapped registers asynchronously
- SOFTRST bit in MCR, which resets some of the memory mapped registers synchronously. See Table 47-6 to see what registers are affected by soft reset.

Soft reset is synchronous and has to follow an internal request/acknowledge procedure across clock domains. Therefore, it may take some time to fully propagate its effects. The CAN_MCR[SOFTRST] bit remains asserted while soft reset is pending, so software can poll this bit to know when the reset has completed. Also, soft reset can not be applied while clocks are shut down in a low power mode. The low power mode should be exited and the clocks resumed before applying soft reset.

When the module is enabled (CAN_MCR[MDIS] bit negated), FlexCAN automatically goes to Freeze mode. In Freeze mode, FlexCAN is un-synchronized to the CAN bus, the HALT and FRZ bits in CAN_MCR Register are set, the internal state machines are disabled and the FRZACK and NOTRDY bits in the CAN_MCR Register are set. The Tx pin is in recessive state and FlexCAN does not initiate any transmission or reception of CAN frames. Note that the Message Buffers and the Rx Individual Mask Registers are not affected by reset, so they are not automatically initialized.

For any configuration change/initialization it is required that FlexCAN is put into Freeze mode (see Freeze mode). The following is a generic initialization sequence applicable to the FlexCAN module:

- Initialize the Module Configuration Register (CAN_MCR)
  - Enable the individual filtering per MB and reception queue features by setting the IRMQ bit
  - Enable the warning interrupts by setting the WRNEN bit
  - If required, disable frame self reception by setting the SRXDIS bit
  - Enable the Rx FIFO by setting the RFEN bit
  - If Rx FIFO is enabled and DMA is required, set DMA bit
  - Enable the abort mechanism by setting the AEN bit
  - Enable the local priority feature by setting the LPRIOEN bit
- Initialize the Control 1 Register (CAN_CTRL1) and optionally the CAN Bit Timing Register (CAN_CBT).
  - Determine the bit timing parameters: PROPSEG, PSEG1, PSEG2, RJW
  - Optionally determine the bit timing parameters: EPROPSEG, EPSEG1, EPSEG2, ERJW
  - Determine the bit rate by programming the PRESDIV field and optionally the EPRESDIV field
  - Determine the internal arbitration mode (LBUF bit)
- Initialize the Message Buffers
  - The Control and Status word of all Message Buffers must be initialized
  - If Rx FIFO was enabled, the ID filter table must be initialized
  - Other entries in each Message Buffer should be initialized as required
- Initialize the Rx Individual Mask Registers (CAN_RXIMRn)
- Set required interrupt mask bits in the CAN_IMASK Registers (for all MB interrupts) and in CAN_CTRL1 / CAN_CTRL2 Registers (for Bus Off and Error interrupts)
- Negate the HALT bit in CAN_MCR

After the last step listed above, FlexCAN attempts to synchronize to the CAN bus.

Chapter 48
De-serial Serial Peripheral Interface (DSPI)

48.1 DSPI chip-specific module information

The device contains seven instances of DSPI:

- DSPI_0
- DSPI_1
- DSPI_2
- DSPI_3
- DSPI_4
- DSPI_M0
- DSPI_M1

DSPI_M0 and DSPI_M1 implement the microsecond bus functionality and support master mode only. DSPI_0, DSPI_1, DSPI_2, DSPI_3 and DSPI_4 are intended for SPI usage and support both master and slave modes.

Note

De-serialization and serial/parallel chaining mode are described and referenced within this chapter; however, these features are not implemented on this chip.

Note

Very strong drive into serial_clock_out pad of DSPI master will affect full-duplex communication. Received data pattern will be corrupted for both master and slave side of DSPI instances.

48.1.1 Number of CTARs

DSPI CTAR registers define different transfer attribute configurations. The DSPI module supports up to eight CTAR registers.

48.1.2 Number of PCS signals

The following table shows the number of peripheral chip select signals available in a DSPI module. CS support is dependent on package size (QFP176/BGA252).

Table 48-1. DSPI PCS signals

DSPI Module	PCS Signals
DSPI_0	PCS0-5
DSPI_1	PCS0-4
DSPI_2	PCS0-3
DSPI_3	PCS0-5
DSPI_4	PCS0-5
DSPI_M0	PCS0-3
DSPI_M1	PCS0-3

48.1.3 IP Instantiation Parameters

The following table shows the full featured DSPI configuration.

Table 48-2. DSPI Module DSPI[4:0]

Feature	Value	Comment
Num_CTAR	8	Number of CTAR Register
TX_FIFO	4	Number of Transmit FIFO registers
RX_FIFO	4	Number of Receive FIFO registers
XSPI	Included	Extended (32-bit) SPI Mode Support
CMD_FIFO	4	Command FIFO depth

Table 48-3. DSPI Module DSPI_M0

Feature	Value	Comment
Num_CTAR	8	Number of CTAR Register
TX_FIFO	4	Number of Transmit FIFO registers
RX_FIFO	4	Number of Receive FIFO registers
XSPI	Included	Extended (32-bit) SPI Mode Support
CMD_FIFO	4	Command FIFO depth
DSI_EXISTS	Included	DSI mode supported.
DSI_64	Included	64 bit DSI Mode Support
TSB_MODE	Included	Timed Serial Bus Mode
ITSB_MODE	Included	Interleaved Timed Serial Bus Mode

Table 48-4. DSPI Module DSPI_M1

Feature	Value	Comment
Num_CTAR	8	Number of CTAR Register
TX_FIFO	4	Number of Transmit FIFO registers
RX_FIFO	4	Number of Receive FIFO registers
XSPI	Included	Extended (32-bit) SPI Mode Support
CMD_FIFO	4	Command FIFO depth
DSI_EXISTS	Included	DSI mode supported.
TSB_MODE	Included	Timed Serial Bus Mode
ITSB_MODE	Included	Interleaved Timed Serial Bus Mode

48.2 Introduction

The deserial serial peripheral interface (DSPI) module provides a synchronous serial bus for communication between an MCU and an external peripheral device. The module supports MCU pin count reduction through serialization and deserialization of MCU internal signals transmitted over the DSPI serial link.

48.2.1 Block Diagram

The block diagram of this module is as follows:

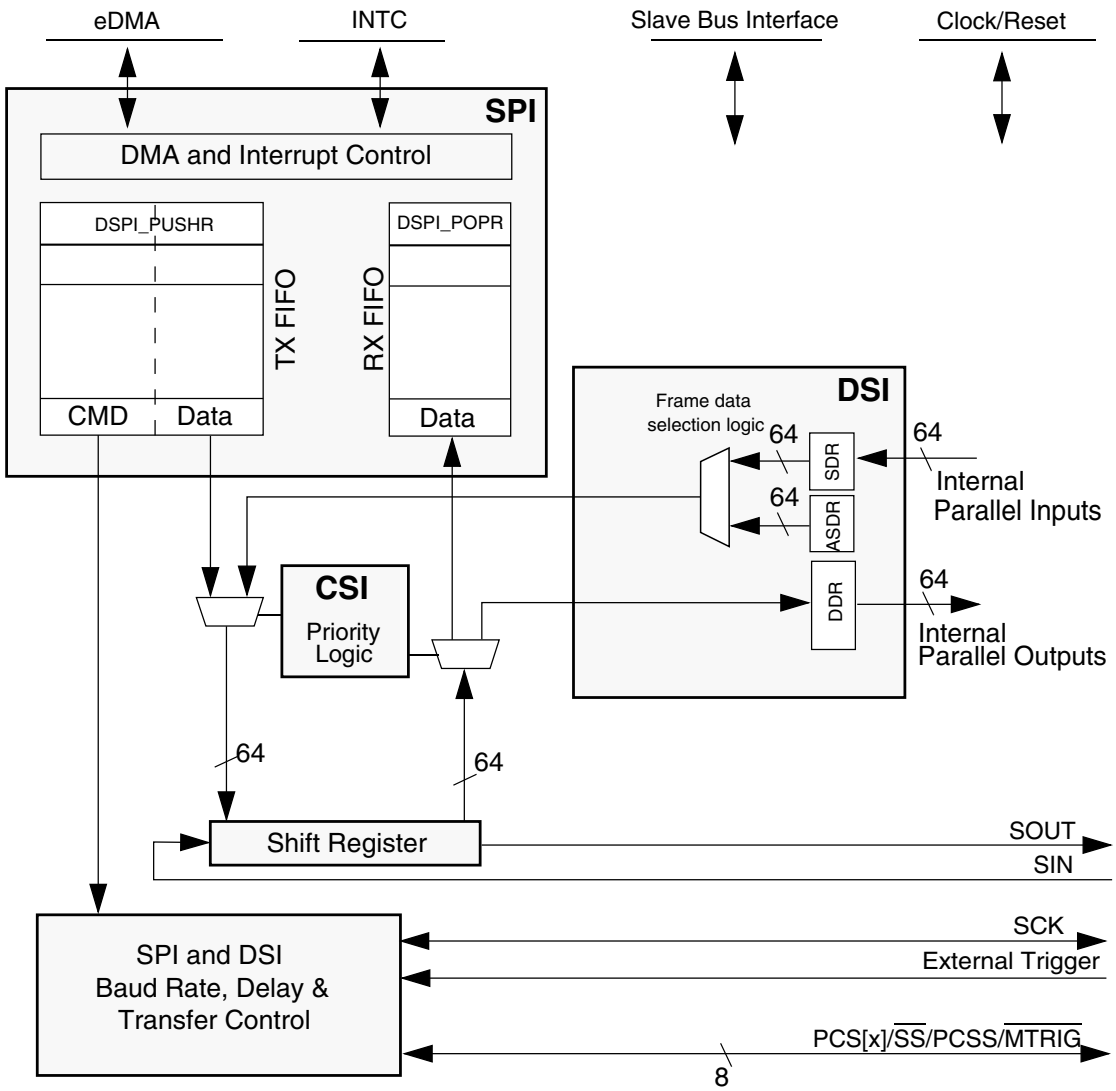


Figure 48-1. DSPI Block Diagram

48.2.2 Features

The module supports the following features:

- Full-duplex, three-wire synchronous transfers
- Master and Slave modes
  - Data streaming operation in Slave mode with continuous slave selection
- Buffered transmit operation using the transmit first in first out (TX FIFO) with depth of four entries

- Buffered receive operation using the receive FIFO (RX FIFO) with depth of four entries
- TX and RX FIFOs can be disabled individually for low-latency updates to SPI queues
- Visibility into TX and RX FIFOs for ease of debugging
- Programmable transfer attributes on a per-frame basis:
  - eight transfer attribute and extended transfer attribute registers
  - Serial clock (SCK) with programmable polarity and phase
  - Various programmable delays
  - Programmable serial frame size of 4–64 bits, expandable by software control
    - SPI frames longer than 32 bits can be supported using the continuous selection format
  - Continuously held chip select capability
  - Parity control
- 6 peripheral chip selects (PCSs), expandable to 64 with external demultiplexer
- Deglitching support for up to 32 PCS with external demultiplexer
- DMA support for adding entries to TX FIFO and removing entries from RX FIFO:
  - TX FIFO is not full (TFFF)
  - RX FIFO is not empty (RFDF)
  - CMD FIFO is not full (CMDFFF)
- Interrupt conditions:
  - End of Queue reached (EOQF)
  - TX FIFO is not full (TFFF)
  - CMD FIFO is not full (CMDFFF)
  - Transfer of current frame complete (TCF)
  - Transfers due from current command frame complete (CMDTCF)
  - Attempt to transmit with an empty Transmit FIFO (TFUF)
  - RX FIFO is not empty (RFDF)

    - Frame received while Receive FIFO is full (RFOF)
    - SPI Parity Error (SPEF)
    - Data present in TX FIFO while CMD FIFO is empty (TFIWF)
- Modified SPI transfer formats for communication with slower peripheral devices
- Power-saving architectural features:
    - Support for Stop mode

The DSPI also supports pin reduction through serialization if enabled for the module.

- Two sources of serialized data:
    - The module memory-mapped register
    - Parallel Input signals
    - Programmable selection of source data on bit basis
- Transfer initiation conditions:
    - Continuous
    - Change in data
- Pin serialization with interleaved SPI frames for control and diagnostics
- Continuous serial communications clock

DSPI supports a combined SPI and DSI modes of operation known as the Combined Serial Interface (CSI). It also supports the downstream Micro Second Channel in the Timed Serial Bus (TSB) configuration or Interleaved Frames Configuration; configurable by software.

Both TSB and Interleaved TSB modes have the following common features::

- Transmission of frames is done at frame boundaries
- Frames from SPI and DSI are identifiable by a bit transmitted at the start of each frame
- Separate interrupts for frame completion from SPI and DSI

48.2.3 SPI Configuration

The SPI configuration allows the module to send and receive serial data. This configuration allows the module to operate as a basic SPI block with internal FIFOs supporting external queue operation. Transmitted data and received data reside in separate FIFOs. The host CPU or a DMA controller read the received data from the Receive FIFO and write transmit data to the Transmit FIFO.

For queued operations, the SPI queues can reside in system RAM, external to the module. Data transfers between the queues and the module FIFOs are accomplished by a DMA controller or host CPU. The following figure shows a system example with DMA, DSPI, and external queues in system RAM.

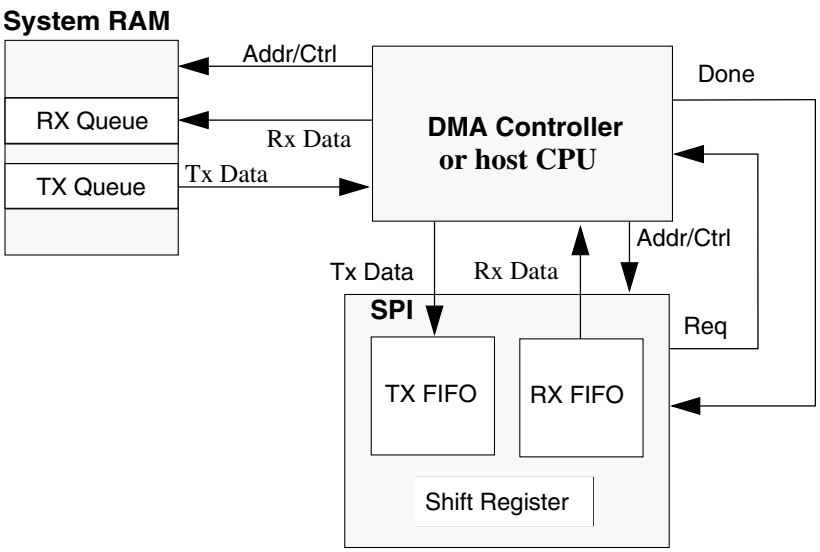


Figure 48-2. SPI with queues and DMA

48.2.4 DSI Configuration

In the DSI configuration the DSPI serializes up to 64 Parallel Input signals or register bits. The DSPI also deserializes the received data to Parallel Output signals or to a memory-mapped register. The data is transferred using a SPI-like protocol.

TSB mode, detailed on Timed Serial Bus (TSB), provides the Micro Second downstream Channel support (MSC), serializing from 4 to 64 Parallel Input signals or register bits.

48.2.5 CSI Configuration

There are three configurations available in this mode.

- The normal CSI configuration is a combination of the SPI and DSI configurations. In this configuration the DSPI interleaves DSI data frames with SPI data frames. Interleaving is done on the frame boundaries.
- In the TSB configuration, transmission of SPI data has higher priority than DSI data.
- In the Interleaved TSB (ITSB) configuration, the frames from SPI and DSI are interleaved without priority. On every trigger, frames from DSI are sent when there are no frames in the SPI or the previous transmission was a frame from SPI. ITSB is detailed in Interleaved TSB (ITSB) Mode.

48.2.6 Modes of Operation

The module supports the following modes of operation that can be divided into two categories:

- Module-specific modes:
  - Master mode
  - Slave mode
  - Module Disable mode
- MCU-specific modes:
  - External Stop mode
  - Debug mode

The module enters module-specific modes when the host writes a module register. The MCU-specific modes are controlled by signals external to the module. The MCU-specific modes are modes that an MCU may enter in parallel to the block-specific modes.

48.2.6.1 Master Mode

Master mode allows the module to initiate and control serial communication. In this mode, the SCK signal and the PCS[*x*] signals are controlled by the module and configured as outputs.

48.2.6.2 Slave Mode

Slave mode allows the module to communicate with SPI/DSI bus masters. In this mode, the module responds to externally controlled serial transfers. The SCK signal and the PCS[0]/$\overline{SS}$ signals are configured as inputs and driven by an SPI bus master.

48.2.6.3 Module Disable Mode

The Module Disable mode can be used for MCU power management. The clock to the non-memory mapped logic in the module can be stopped while in the Module Disable mode.

48.2.6.4 External Stop Mode

External Stop mode is used for MCU power management. The module supports the Peripheral Bus Stop mode mechanism. When a request is made to enter External Stop mode, it acknowledges the request and completes the transfer that is in progress. When the module reaches the frame boundary, it signals that the protocol clock to the module may be shut off.

48.2.6.5 Debug Mode

Debug mode is used for system development and debugging. The MCR[FRZ] bit controls module behavior in the Debug mode:

- If the bit is set, the module stops all serial transfers, when the MCU is in debug mode.
- If the bit is cleared, the MCU debug mode has no effect on the module.

48.3 Module signal descriptions

This section provides description of the module signals.

The following table lists the signals that may connect off chip depending on device implementation.

Table 48-5. Module signal descriptions

Signal	Master Mode	Slave Mode	Port Direction
PCS0/$\overline{SS}$	Peripheral Chip Select 0 output	Slave Select input	I/O

Table continues on the next page...

Table 48-5. Module signal descriptions (continued)

Signal	Master Mode	Slave Mode	Port Direction
PCS[3:1]	Peripheral Chip Select 1 – 3	Unused	O
PCS4/ $\overline{\text{MTRIG}}$	Peripheral Chip Select 4	Master Trigger	O
PCS5/ $\overline{\text{PCSS}}$	Peripheral Chip Select 5 /Peripheral Chip Select Strobe	Unused	O
SIN	Serial Data In	Serial Data In	I
SOUT	Serial Data Out	Serial Data Out	O
SCK	Master mode: Serial Clock (output)	Serial Clock (input)	I/O

48.3.1 PCS0/$\overline{\text{SS}}$ — Peripheral Chip Select/Slave Select

In Master mode, the PCS0 signal is an output that selects which slave device the current transmission is intended for.

In Slave mode, the active low $\overline{\text{SS}}$ signal is an input signal that allows an SPI master to select the module as the target for transmission.

48.3.2 PCS1 – PCS3 — Peripheral Chip Selects 1 – 3

PCS1 – PCS3 are output signals in Master mode.

In Slave mode, these signals are unused.

48.3.3 PCS4/$\overline{\text{MTRIG}}$ — Peripheral Chip Select 4/Master Trigger

In Master mode, PCS4 is an output signal.

In slave mode, the active low $\overline{\text{MTRIG}}$ is an output trigger signal that indicates that a change in data to be serialized has occurred. The $\overline{\text{MTRIG}}$ provides a pulse in DSI Configuration when a change in data to be serialized occurs. The $\overline{\text{MTRIG}}$ pulse is four protocol clock cycles in duration. If the DSPI is in slave mode and the MTO is disabled, the PCS4/$\overline{\text{MTRIG}}$ signal is unused.

48.3.4 PCS5/$\overline{\text{PCSS}}$ — Peripheral Chip Select 5/Peripheral Chip Select Strobe

PCS5 is a Peripheral Chip Select output signal. When the module is in Master mode and the MCR[PCSSE] bit is cleared, this signal selects the slave device the current transfer is intended for.

When the module is in Master mode and the MCR[PCSSE] bit is set, the $\overline{\text{PCSS}}$ signal acts as a strobe to an external peripheral chip select demultiplexer, which decodes the PCS0 – PCS4 signals, preventing glitches on the demultiplexer outputs.

This signal is not used in Slave mode.

48.3.5 SIN — Serial Input

SIN is a serial data input signal.

NOTE

Serial Data Out output buffers are controller through SIUL, and cannot be controller through the module

48.3.6 SOUT — Serial Output

SOUT is a serial data output signal.

NOTE

Serial Data Out output buffers are controller through SIUL, and cannot be controller through the module.

48.3.7 SCK — Serial Clock

SCK is a serial communication clock signal.

In Master mode, the module generates the SCK.

In Slave mode, SCK is an input from an external bus master.

48.4 Memory Map/Register Definition

Register accesses to memory addresses that are reserved or undefined result in a transfer error. Write access to the POPR also results in a transfer error.

DSPI memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (DSPI_MCR)	32	R/W	0000_4001h	48.4.1/1950
8	Transfer Count Register (DSPI_TCR)	32	R/W	0000_0000h	48.4.2/1953
C	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR0)	32	R/W	7800_0000h	48.4.3/1954
C	Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTAR0_SLAVE)	32	R/W	7800_0000h	48.4.4/1959
10	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR1)	32	R/W	7800_0000h	48.4.3/1954
10	Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTAR1_SLAVE)	32	R/W	7800_0000h	48.4.4/1959
14	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR2)	32	R/W	7800_0000h	48.4.3/1954
18	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR3)	32	R/W	7800_0000h	48.4.3/1954
1C	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR4)	32	R/W	7800_0000h	48.4.3/1954
20	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR5)	32	R/W	7800_0000h	48.4.3/1954
24	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR6)	32	R/W	7800_0000h	48.4.3/1954
28	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR7)	32	R/W	7800_0000h	48.4.3/1954
2C	Status Register (DSPI_SR)	32	R/W	0201_0000h	48.4.5/1961
30	DMA/Interrupt Request Select and Enable Register (DSPI_RSER)	32	R/W	0000_0000h	48.4.6/1965
34	PUSH TX FIFO Register In Master Mode (DSPI_PUSHR)	32	R/W	0000_0000h	48.4.7/1968
34	PUSH TX FIFO Register In Slave Mode (DSPI_PUSHR_SLAVE)	32	R/W	0000_0000h	48.4.8/1970
38	POP RX FIFO Register (DSPI_POPR)	32	R	0000_0000h	48.4.9/1971
3C	Transmit FIFO Registers (DSPI_TXFR0)	32	R	0000_0000h	48.4.10/ 1971
40	Transmit FIFO Registers (DSPI_TXFR1)	32	R	0000_0000h	48.4.10/ 1971
44	Transmit FIFO Registers (DSPI_TXFR2)	32	R	0000_0000h	48.4.10/ 1971
48	Transmit FIFO Registers (DSPI_TXFR3)	32	R	0000_0000h	48.4.10/ 1971
7C	Receive FIFO Registers (DSPI_RXFR0)	32	R	0000_0000h	48.4.11/ 1972
80	Receive FIFO Registers (DSPI_RXFR1)	32	R	0000_0000h	48.4.11/ 1972
84	Receive FIFO Registers (DSPI_RXFR2)	32	R	0000_0000h	48.4.11/ 1972

Table continues on the next page...

DSPI memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
88	Receive FIFO Registers (DSPI_RXFR3)	32	R	0000_0000h	48.4.11/ 1972
BC	DSI Configuration Register 0 (DSPI_DSICR0)	32	R/W	0000_0000h	48.4.12/ 1973
C0	DSI Serialization Data Register 0 (DSPI_SDR0)	32	R	0000_0000h	48.4.13/ 1975
C4	DSI Alternate Serialization Data Register 0 (DSPI_ASDR0)	32	R/W	0000_0000h	48.4.14/ 1976
C8	DSI Transmit Comparison Register 0 (DSPI_COMPR0)	32	R	0000_0000h	48.4.15/ 1976
D0	DSI Configuration Register 1 (DSPI_DSICR1)	32	R/W	0000_0000h	48.4.16/ 1977
D4	DSI Serialization Source Select Register 0 (DSPI_SSR0)	32	R/W	0000_0000h	48.4.17/ 1979
F0	DSI Serialization Data Register 1 (DSPI_SDR1)	32	R	0000_0000h	48.4.18/ 1979
F4	DSI Alternate Serialization Data Register 1 (DSPI_ASDR1)	32	R/W	0000_0000h	48.4.19/ 1980
F8	DSI Transmit Comparison Register 1 (DSPI_COMPR1)	32	R	0000_0000h	48.4.20/ 1981
FC	DSI Deserialization Data Register 1 (DSPI_DDR1)	32	R	0000_0000h	48.4.21/ 1981
100	DSI Serialization Source Select Register 1 (DSPI_SSR1)	32	R/W	0000_0000h	48.4.22/ 1982
114	DSI Deserialized Data Interrupt Mask Register 1 (DSPI_DIMR1)	32	R/W	0000_0000h	48.4.23/ 1983
118	DSI Deserialized Data Polarity Interrupt Register 1 (DSPI_DPIR1)	32	R/W	0000_0000h	48.4.24/ 1983
11C	Clock and Transfer Attributes Register Extended (DSPI_CTARE0)	32	R/W	0000_0001h	48.4.25/ 1984
120	Clock and Transfer Attributes Register Extended (DSPI_CTARE1)	32	R/W	0000_0001h	48.4.25/ 1984
124	Clock and Transfer Attributes Register Extended (DSPI_CTARE2)	32	R/W	0000_0001h	48.4.25/ 1984
128	Clock and Transfer Attributes Register Extended (DSPI_CTARE3)	32	R/W	0000_0001h	48.4.25/ 1984
12C	Clock and Transfer Attributes Register Extended (DSPI_CTARE4)	32	R/W	0000_0001h	48.4.25/ 1984
130	Clock and Transfer Attributes Register Extended (DSPI_CTARE5)	32	R/W	0000_0001h	48.4.25/ 1984
134	Clock and Transfer Attributes Register Extended (DSPI_CTARE6)	32	R/W	0000_0001h	48.4.25/ 1984
138	Clock and Transfer Attributes Register Extended (DSPI_CTARE7)	32	R/W	0000_0001h	48.4.25/ 1984

Table continues on the next page...

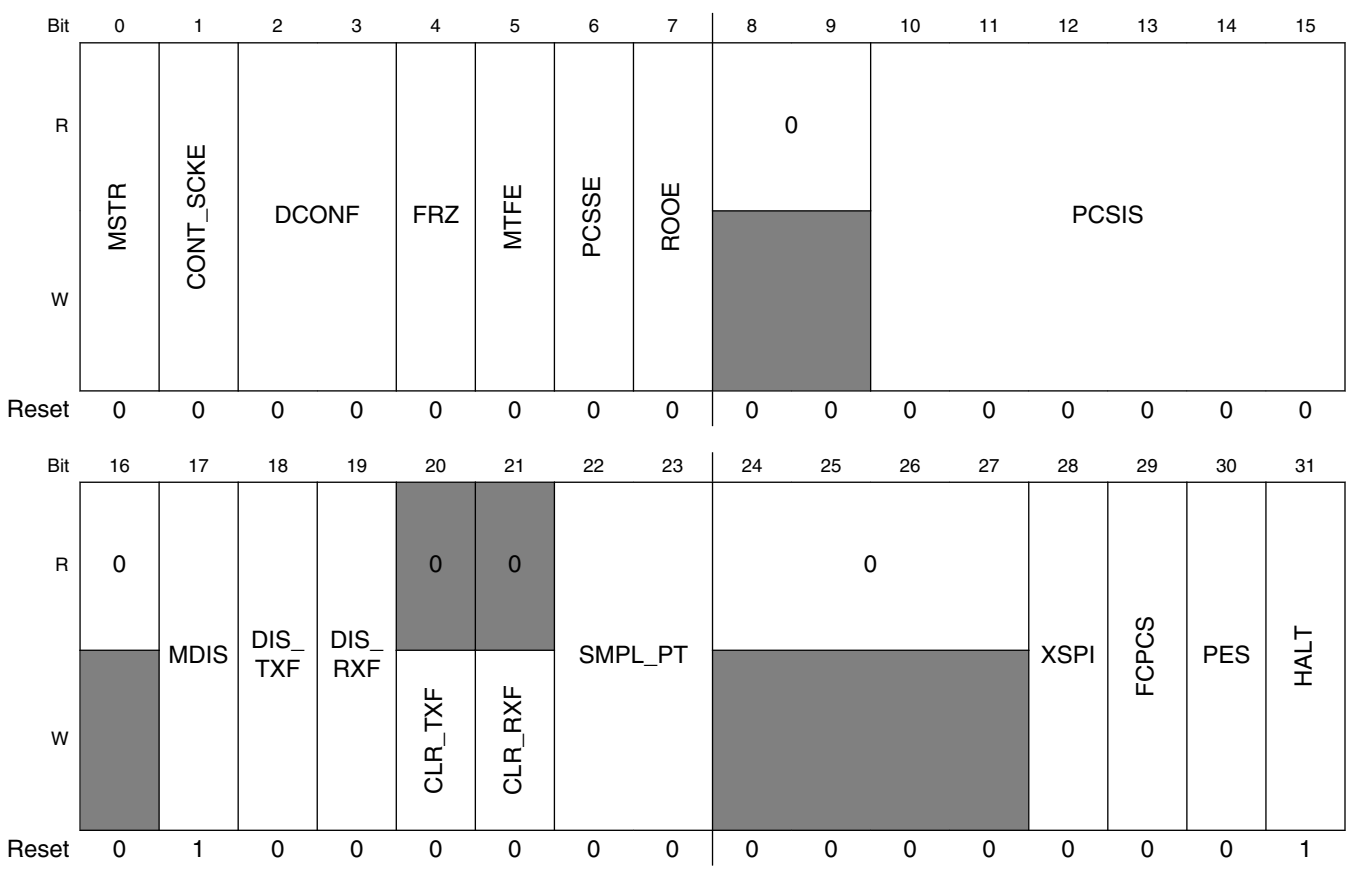
DSPI memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13C	Status Register Extended (DSPI_SREX)	32	R	0000_0000h	48.4.26/ 1985
140	Trigger Register (DSPI_TRIG)	32	R/W	0000_0000h	48.4.27/ 1986
150	Time Slot Length Register (DSPI_TSL)	32	R/W	0000_0000h	48.4.28/ 1987
154	Time Slot Configuration Register (DSPI_TS_CONF)	32	R/W	0000_0000h	48.4.29/ 1988

48.4.1 Module Configuration Register (DSPI_MCR)

Contains bits to configure various attributes associated with the module operations. The HALT and MDIS bits can be changed at any time, but the effect takes place only on the next frame boundary. Only the HALT and MDIS bits in the MCR can be changed, while the module is in the Running state.

Address: 0h base + 0h offset = 0h



DSPI_MCR field descriptions

Field	Description
0 MSTR	Master/Slave Mode Select Configures the module for either Master mode or Slave mode. 0 The module is in Slave mode. 1 The module is in Master mode.
1 CONT_SCKE	Continuous SCK Enable Enables the Serial Communication Clock (SCK) to run continuously. 0 Continuous SCK disabled. 1 Continuous SCK enabled.
2–3 DCONF	DSPI Configuration Selects among the different configurations of the module. 00 SPI 01 DSI 10 CSI 11 Reserved
4 FRZ	Freeze Enables the DSPI transfers to be stopped on the next frame boundary when the device enters Debug mode. 0 Do not halt serial transfers in Debug mode. 1 Halt serial transfers in Debug mode.
5 MTFE	Modified Timing Format Enable Enables a modified transfer format to be used. 0 Modified SPI transfer format disabled. 1 Modified SPI transfer format enabled.
6 PCSSE	Peripheral Chip Select Strobe Enable Enables the PCS5/ $\overline{\text{PCSS}}$ to operate as a PCS Strobe output signal. 0 PCS5/ $\overline{\text{PCSS}}$ is used as the Peripheral Chip Select[5] signal. 1 PCS5/ $\overline{\text{PCSS}}$ is used as an active-low PCS Strobe signal.
7 ROOE	Receive FIFO Overflow Overwrite Enable In the RX FIFO overflow condition, configures the module to ignore the incoming serial data or overwrite existing data. If the RX FIFO is full and new data is received, the data from the transfer, generating the overflow, is ignored or shifted into the shift register. 0 Incoming data is ignored. 1 Incoming data is shifted into the shift register.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 PCSIS	Peripheral Chip Select x Inactive State Determines the inactive state of PCSx.

Table continues on the next page...

DSPI_MCR field descriptions (continued)

Field	Description
	0 The inactive state of PCSx is low. 1 The inactive state of PCSx is high.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 MDIS	Module Disable Allows the clock to be stopped to the non-memory mapped logic in the module effectively putting it in a software-controlled power-saving state. The reset value of the MDIS bit is parameterized, with a default reset value of 0. When DSPI is used in Slave Mode, it is recommended to leave this bit set to '0', since a slave doesn't have control over master transactions. 0 Enables the module clocks. 1 Allows external logic to disable the module clocks.
18 DIS_TXF	Disable Transmit FIFO When the TX FIFO is disabled, the transmit part of the module operates as a simplified double-buffered SPI. This bit can be written only when the MDIS bit is cleared. 0 TX FIFO is enabled. 1 TX FIFO is disabled.
19 DIS_RXF	Disable Receive FIFO When the RX FIFO is disabled, the receive part of the module operates as a simplified double-buffered SPI. This bit can only be written when the MDIS bit is cleared. 0 RX FIFO is enabled. 1 RX FIFO is disabled.
20 CLR_TXF	Clear TX FIFO Flushes the TX FIFO. Writing a 1 to CLR_TXF clears the TX FIFO Counter. The CLR_TXF bit is always read as zero. 0 Do not clear the TX FIFO counter. 1 Clear the TX FIFO counter.
21 CLR_RXF	Flushes the RX FIFO. Writing a 1 to CLR_RXF clears the RX Counter. The CLR_RXF bit is always read as zero. 0 Do not clear the RX FIFO counter. 1 Clear the RX FIFO counter.
22–23 SMPL_PT	Sample Point Controls when the module master samples SIN in Modified Transfer Format. This field is valid only when CPHA bit in CTARn[CPHA] is 0. 00 0 protocol clocks between SCK edge and SIN sample 01 1 protocol clock between SCK edge and SIN sample 10 2 protocol clocks between SCK edge and SIN sample 11 Reserved
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

DSPI_MCR field descriptions (continued)

Field	Description
28 XSPI	Extended SPI Mode This bit enables usage of CTARE (Command and Transfer Attribute Register Extended) Registers. CTARE registers allow the user to send up to 32 bit SPI frames. Command Cycling is also enabled which allows the user to send mutiple Data Frames using a single Command Frame. When MCR[DIS_TXF] is asserted, the Extended SPI Mode cannot be used to transmit SPI frames which are more than 16 bits in size. 0 Normal SPI Mode. Frame size can be up to 16 bits. Command Cycling is not available in this mode. 1 Extended SPI Mode. Up to 32 bit SPI Frames along with Command Cycling is Enabled.
29 FCPCS	Fast Continuous PCS Mode. This bit enables the masking of “After SCK (t_{ASC})” and “PCS to SCK (t_{CSC})” delays when operating in Continuous PCS mode. This masking is not available if Continuous SCK mode is enabled. The individual delay masks are selected via bits MASC and MCSC of the PUSHR register. The firmware should select appropriate masks when providing continuous frames via the PUSHR register. 0 Normal or Slow Continuous PCS mode. Masking of delays is disabled. 1 Fast Continuous PCS mode. Delays masked via control bits in PUSHR register.
30 PES	Parity Error Stop Controls SPI operation when a parity error is detected in a received SPI frame. 0 SPI frame transmission continues. 1 SPI frame transmission stops.
31 HALT	Halt The HALT bit starts and stops DSPI transfers. See Start and Stop of module transfers. 0 Start transfers. 1 Stop transfers.

48.4.2 Transfer Count Register (DSPI_TCR)

TCR contains a counter that indicates the number of SPI transfers made. The transfer counter is intended to assist in queue management. Do not write the TCR when the module is in the Running state.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SPI_TCNT																0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_TCR field descriptions

Field	Description
0–15 SPI_TCNT	SPI Transfer Counter Counts the number of SPI transfers the module makes. The SPI_TCNT field increments every time the last bit of an SPI frame is transmitted. A value written to SPI_TCNT presets the counter to that value. SPI_TCNT is reset to zero at the beginning of the frame when the CTCNT field is set in the executing SPI command. The Transfer Counter wraps around; incrementing the counter past 65535 resets the counter to zero.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.3 Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR*n*)

CTAR registers are used to define different transfer attributes. Do not write to the CTAR registers while the module is in the Running state.

In Master mode, the CTAR registers define combinations of transfer attributes such as frame size, clock phase and polarity, data bit ordering, baud rate, and various delays. In slave mode, a subset of the bitfields in CTAR0 and CTAR1 are used to set the slave transfer attributes.

When the module is configured as an SPI master, the CTAS field in the command portion of the TX FIFO entry selects which of the CTAR registers is used. When the module is configured as an SPI bus slave, the CTAR0 register is used.

When the module is configured as a DSI master, the DSICTAS field in the DSI Configuration Register 0 (DSICR0), selects which of the CTAR register is used. When the module is configured as a DSI bus slave, the CTAR1 register is used.

In CSI Configuration, the transfer attributes are selected based on whether the current frame is SPI data or DSI data. SPI transfers in CSI Configuration follow the protocol described for SPI Configuration, and DSI transfers in CSI Configuration follow the protocol described for DSI Configuration. CSI Configuration is valid only in conjunction with master mode.

TSB mode sets some limitations on transfer attributes:

- Clock phase is forced to be CPHA = 1 and the CPHA bit setting has no effect.
- PCS lines are driven at the driving edge of the SCK clock together with SOUT, so PCS assertion and negation delays control is unavailable and PCSSCK, PASC, CSSCK and ASC fields have no effect.
- Delay after transfer can be set from 1 to 64 serial clocks with help of PDT and DT fields

Address: 0h base + Ch offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DBR	FMSZ				CPOL	CPHA	LSBFE	PCSSCK		PASC		PDT		PBR	
W																
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CSSCK				ASC				DT				BR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_CTAR*n* field descriptions

Field	Description
0 DBR	Double Baud Rate Doubles the effective baud rate of the Serial Communications Clock (SCK). This field is used only in master mode. It effectively halves the Baud Rate division ratio, supporting faster frequencies, and odd division ratios for the Serial Communications Clock (SCK). When the DBR bit is set, the duty cycle of the Serial Communications Clock (SCK) depends on the value in the Baud Rate Prescaler and the Clock Phase bit as listed in the following table. See the BR field description for details on how to compute the baud rate.
1–4 FMSZ	Frame Size The number of bits transferred per frame is equal to the FMSZ field value plus 1. Regardless of the transmission mode, the minimum valid frame size value is 4. There is a constraint on the minimum allowable Frame size, which depends on the Register Read/Write clock to the System Protocol Clock Ratio. The minimum Frame size (FMSZ + 1) is decided by the following equation. Upper Ceiling must be applied for non-integer values.

Table 48-12. DSPI SCK Duty Cycle

DBR	CPHA	PBR	SCK Duty Cycle
0	any	any	50/50
1	0	00	50/50
1	0	01	33/66
1	0	10	40/60
1	0	11	43/57
1	1	00	50/50
1	1	01	66/33
1	1	10	60/40
1	1	11	57/43

0 The baud rate is computed normally with a 50/50 duty cycle.
1 The baud rate is doubled with the duty cycle depending on the Baud Rate Prescaler.

Table continues on the next page...

DSPI_CTAR*n* field descriptions (continued)

Field	Description
	Min. Frame Size = ((4 x f_p) + (3 x f_r)) / (n x f_r) f_p = Protocol Clock Frequency f_r = Bus clock (register interface clock) Frequency n = (PBR x BR) / (1 + DBR) = Multiple of protocol clock required to get Baud (SCK) Clock However, the minimum frame size can never be less than 4. There is no constraint on the maximum programmable frame size. Also note that `fp' can be equal to, less than or greater than `fr' purely based on user requirement. When the DSPI operates in TSB mode, the FMSZ field value plus 1 is equal to the data frame bit number, where control of the PCS assertion switches from the DSICR0 to the DSICR1 register. If TSB Mode is operated when DSI is used in 64 bit mode, then DSICR0[FMSZ4] also comes into picture while switching PCS assertion control.
5 CPOL	Clock Polarity Selects the inactive state of the Serial Communications Clock (SCK). This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock polarities. When the Continuous Selection Format is selected, switching between clock polarities without stopping the module can cause errors in the transfer due to the peripheral device interpreting the switch of clock polarity as a valid clock edge. 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.
6 CPHA	Clock Phase Selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode or TSB mode, the bit value is ignored and the transfers are done as if the CPHA bit is set to 1. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
7 LSBFE	LSB First Specifies whether the LSB or MSB of the frame is transferred first. When the DSPI operates in TSB configuration, this bit should be set to be compliant to the MSC specification. 0 Data is transferred MSB first. 1 Data is transferred LSB first.
8–9 PCSSCK	PCS to SCK Delay Prescaler Selects the prescaler value for the delay between assertion of PCS and the first edge of the SCK. See the CSSCK field description for information on how to compute the PCS to SCK Delay. Refer PCS to SCK Delay (t_{CSC}) for more details. In the TSB mode, the PCSSCK field has no effect. 00 PCS to SCK Prescaler value is 1. 01 PCS to SCK Prescaler value is 3. 10 PCS to SCK Prescaler value is 5. 11 PCS to SCK Prescaler value is 7.
10–11 PASC	After SCK Delay Prescaler

Table continues on the next page...

DSPI_CTAR*n* field descriptions (continued)

Field	Description
	Selects the prescaler value for the delay between the last edge of SCK and the negation of PCS. See the ASC field description for information on how to compute the After SCK Delay. In the TSB mode, the PASC field has no effect. Refer After SCK Delay (t_{ASC}) for more details. 00 Delay after Transfer Prescaler value is 1. 01 Delay after Transfer Prescaler value is 3. 10 Delay after Transfer Prescaler value is 5. 11 Delay after Transfer Prescaler value is 7.
12–13 PDT	Delay after Transfer Prescaler Selects the prescaler value for the delay between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. The PDT field is only used in master mode. In the TSB mode the PDT field defines two MSB bits of the Delay after Transfer. See the DT field description for details on how to compute the Delay after Transfer. Refer Delay after Transfer (t_{DT}) for more details. 00 Delay after Transfer Prescaler value is 1. 01 Delay after Transfer Prescaler value is 3. 10 Delay after Transfer Prescaler value is 5. 11 Delay after Transfer Prescaler value is 7.
14–15 PBR	Baud Rate Prescaler Selects the prescaler value for the baud rate. This field is used only in master mode. The baud rate is the frequency of the SCK. The protocol clock is divided by the prescaler value before the baud rate selection takes place. See the BR field description for details on how to compute the baud rate. 00 Baud Rate Prescaler value is 2. 01 Baud Rate Prescaler value is 3. 10 Baud Rate Prescaler value is 5. 11 Baud Rate Prescaler value is 7.
16–19 CSSCK	PCS to SCK Delay Scaler Selects the scaler value for the PCS to SCK delay. This field is used only in master mode. In the TSB mode the field has no effect. The PCS to SCK Delay is the delay between the assertion of PCS and the first edge of the SCK. The delay is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{CSC} = (1/f_P) \times PCSSCK \times CSSCK$. The following table lists the delay scaler values.

Table 48-11. Delay Scaler Encoding

Field Value	Delay Scaler Value
0000	2
0001	4
0010	8
0011	16
0100	32
0101	64
0110	128
0111	256

Table continues on the next page...

DSPI_CTAR*n* field descriptions (continued)

Field	Description
	Table 48-11. Delay Scaler Encoding (continued) Field Value / Delay Scaler Value: 1000 – 512 1001 – 1024 1010 – 2048 1011 – 4096 1100 – 8192 1101 – 16384 1110 – 32768 1111 – 65536 Refer PCS to SCK Delay (t_{CSC}) for more details.
20–23 ASC	After SCK Delay Scaler Selects the scaler value for the After SCK Delay. This field is used only in master mode. The After SCK Delay is the delay between the last edge of SCK and the negation of PCS. The delay is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{ASC} = (1/f_P) \times PASC \times ASC$ See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values. Refer After SCK Delay (t_{ASC}) for more details.
24–27 DT	Delay After Transfer Scaler Selects the Delay after Transfer Scaler. This field is used only in master mode. The Delay after Transfer is the time between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. In the Continuous Serial Communications Clock operation, the DT value is fixed to one SCK clock period, The Delay after Transfer is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{DT} = (1/f_P) \times PDT \times DT$ See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values. In the TSB mode, the Delay after Transfer is equal to a number formed by concatenation of the PDT and DT fields plus 1 of the SCK clock periods. Refer Delay after Transfer (t_{DT}) for more details.
28–31 BR	Baud Rate Scaler Selects the scaler value for the baud rate. This field is used only in master mode. The prescaled protocol clock is divided by the Baud Rate Scaler to generate the frequency of the SCK. The baud rate is computed according to the following equation: SCK baud rate = (f_P /PBR) x [(1+DBR)/BR] The following table lists the baud rate scaler values. **Table 48-10. Baud Rate Scaler** CTARn[BR] / Baud Rate Scaler Value: 0000 – 2

Table continues on the next page...

DSPI_CTAR*n* field descriptions (continued)

Field	Description

Table 48-10. Baud Rate Scaler (continued)

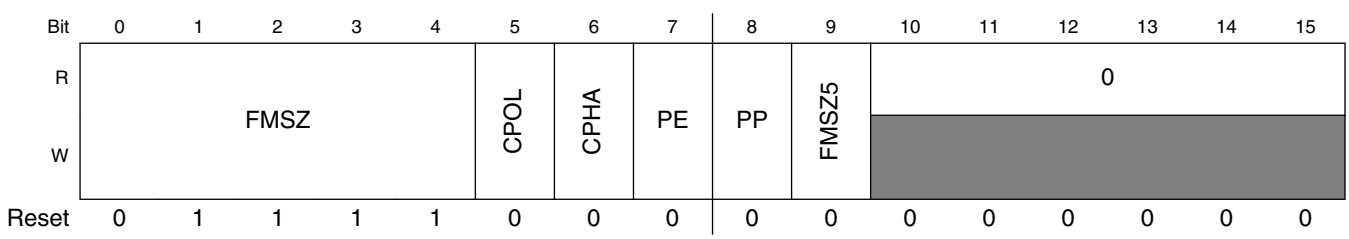
CTARn[BR]	Baud Rate Scaler Value
0001	4
0010	6
0011	8
0100	16
0101	32
0110	64
0111	128
1000	256
1001	512
1010	1024
1011	2048
1100	4096
1101	8192
1110	16384
1111	32768

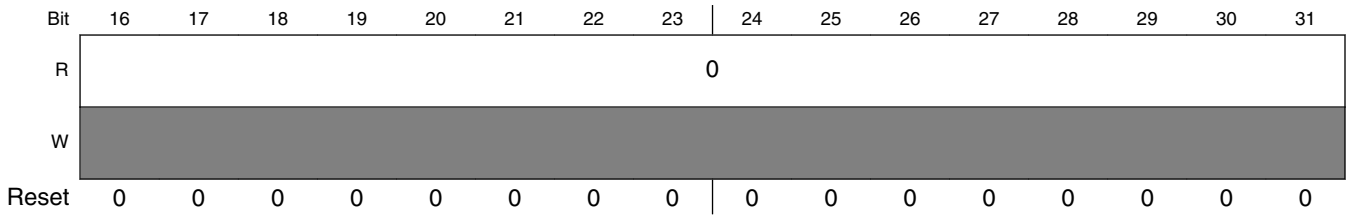
48.4.4 Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTAR*n*_SLAVE)

When the module is configured as an SPI bus slave, the CTAR0 register is used.

When the module is configured as a DSI master, the DSICTAS field in the DSI Configuration Register 0 (DSICR0), selects which of the CTAR register is used. When the module is configured as a DSI bus slave, the CTAR1 register is used.

Address: 0h base + Ch offset + (4d × i), where i=0d to 1d





DSPI_CTAR*n*_SLAVE field descriptions

Field	Description
0–4 FMSZ	Frame Size The number of bits transfered per frame is equal to the concatenated field {CTARn_SLAVE[FMSZ5], CTARn_SLAVE[FMSZ]} value plus 1. Note that the minimum valid value of frame size is 4.
5 CPOL	Clock Polarity Selects the inactive state of the Serial Communications Clock (SCK). 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.
6 CPHA	Clock Phase Selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode or TSB mode , the bit value is ignored and the transfers are done as the CPHA bit is set to 1. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
7 PE	Parity Enable Enables parity bit transmission and reception for the frame. 0 No parity bit included/checked. 1 Parity bit is transmitted instead of last data bit in frame, parity checked for received frame.
8 PP	Parity Polarity Controls polarity of the parity bit transmitted and checked. 0 Even Parity: the number of 1 bits in the transmitted frame is even. The SR[SPEF] bit is set if the number of 1 bits is odd in the received frame. 1 Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[SPEF] bit is set if the number of 1 bits is even in the received frame.
9 FMSZ5	MSB of Frame Size when DSI is used in 64-bit Mode The number of bits transferred per frame is equal to the concatenated field {CTARn_SLAVE[FMSZ[5]], CTARn_SLAVE[FMSZ]} value plus 1.
10–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.5 Status Register (DSPI_SR)

SR contains status and flag bits. The bits reflect the status of the module and indicate the occurrence of events that can generate interrupt or DMA requests. Software can clear flag bits in the SR by writing a 1 to them. Writing a 0 to a flag bit has no effect. This register may not be writable in Module Disable mode due to the use of power saving mechanisms.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF	TXRXS	SPITCF	EOQF	TFUF	DSITCF	TFFF	BSYF	CMDTCF	DPEF	SPEF	DDIF	RFOF	TFIWF	RFDF	CMDFFF
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXCTR				TXNXTPTR				RXCTR				POPNXTPTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_SR field descriptions

Field	Description
0 TCF	Transfer Complete Flag Indicates that all bits in a frame have been shifted out. TCF remains set until it is cleared by writing a 1 to it. 0 Transfer not complete. 1 Transfer complete.

Table continues on the next page...

DSPI_SR field descriptions (continued)

Field	Description
1 TXRXS	TX and RX Status Reflects the run status of the module. 0 Transmit and receive operations are disabled (The module is in Stopped state). 1 Transmit and receive operations are enabled (The module is in Running state).
2 SPITCF	SPI Frame Transfer Complete Flag. This bit functions similar to TCF except that this bit is asserted only on SPI frame transmission completion in CSI Mode. This includes the ITSB Mode and 'TSB in CSI' Mode. This flag will not be asserted when the module is used in SPI or DSI Modes. 0 SPI frame transfer is not complete. 1 SPI frame transfer is complete.
3 EOQF	End of Queue Flag Indicates that the last entry in a queue has been transmitted when the module is in Master mode. The EOQF bit is set when the TX FIFO entry has the EOQ bit set in the command halfword and the end of the transfer is reached. The EOQF bit remains set until cleared by writing a 1 to it. When the EOQF bit is set, the TXRXS bit is automatically cleared. 0 EOQ is not set in the executing command. 1 EOQ is set in the executing SPI command.
4 TFUF	Transmit FIFO Underflow Flag Indicates an underflow condition in the TX FIFO. The transmit underflow condition is detected only for DSPI blocks operating in Slave mode and SPI configuration. TFUF is set when the TX FIFO of the module operating in SPI Slave mode is empty and an external SPI master initiates a transfer. The TFUF bit remains set until cleared by writing 1 to it. 0 No TX FIFO underflow. 1 TX FIFO underflow has occurred.
5 DSITCF	DSI Frame Transfer Complete Flag. This bit functions similar to TCF except that this bit is asserted only on DSI frame transmission completion in CSI Mode. This includes the ITSB Mode and 'TSB in CSI' Mode. This flag will not be asserted when the module is used in SPI or DSI Modes. 0 DSI frame transfer is not complete. 1 DSI frame transfer is complete.
6 TFFF	Transmit FIFO Fill Flag Provides a method for the module to request more entries to be added to the TX FIFO. The TFFF bit is set while the TX FIFO is not full. The TFFF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller to the TX FIFO full request. 0 TX FIFO is full. 1 TX FIFO is not full.
7 BSYF	Busy Flag. This bit is valid only when DSPI_MCR[XSPI] is enabled. Indicates that the current Command Frame is being used for transmiting multiple data frames. This bit is not set for the last Data Frame of a Cyclic

Table continues on the next page...

DSPI_SR field descriptions (continued)

Field	Description
	command Transfer or when DSPI_CTARE[DTCP] = 1. Refer Command First In First Out (CMD FIFO) Buffering Mechanism for more details. 0 No Cyclic Command Transfer in Progress. 1 Cyclic Command Transfer is in progress. Current Data Frame is not the last data frame for on-going cyclic command transfer..
8 CMDTCF	Command Transfer Complete Flag. Indicates that the last Data frame for the current Cyclic Command has been transmitted. Hence this bit is set only for the last Data Frame of a Cyclic Command Transfer or when DSPI_CTARE[DTCP] = 1. The bit remains set until it is cleared by writing a '1' to it. 0 Data Transfer by current Command not complete. 1 Data Transfer by current Command is complete.
9 DPEF	DSI Parity Error Flag Indicates that a DSI frame with parity error had been received. The bit remains set until it is cleared by writing a 1 to it. 0 No parity error. 1 Parity error has occurred.
10 SPEF	SPI Parity Error Flag Indicates that a SPI frame with parity error had been received. The bit remains set until it is cleared by writing a 1 to it. 0 No parity error. 1 Parity error has occurred.
11 DDIF	DSI Data Received with Active Bits Indicates that a DSI frame had been received with bits selected by DIMR with active polarity, defined by the DPIR register. The bit remains set until it is cleared by writing a 1 to it. 0 No DSI data with active bits was received. 1 DSI data with active bits was received.
12 RFOF	Receive FIFO Overflow Flag Indicates an overflow condition in the RX FIFO. The field is set when the RX FIFO and shift register are full and a transfer is initiated. The bit remains set until it is cleared by writing a 1 to it. 0 No Rx FIFO overflow. 1 Rx FIFO overflow has occurred.
13 TFIWF	Tranmit FIFO Invalid Write Flag Indicates Data Write on TX FIFO while CMD FIFO is empty. Without a Command, the Data entries present in TXFIFO are invalid. This bit remains set until it is cleared by writing a '1' to it. 0 No Invalid Data present in TX FIFO. 1 Invalid Data present in TX FIFO since CMD FIFO is empty.
14 RFDF	Receive FIFO Drain Flag

Table continues on the next page...

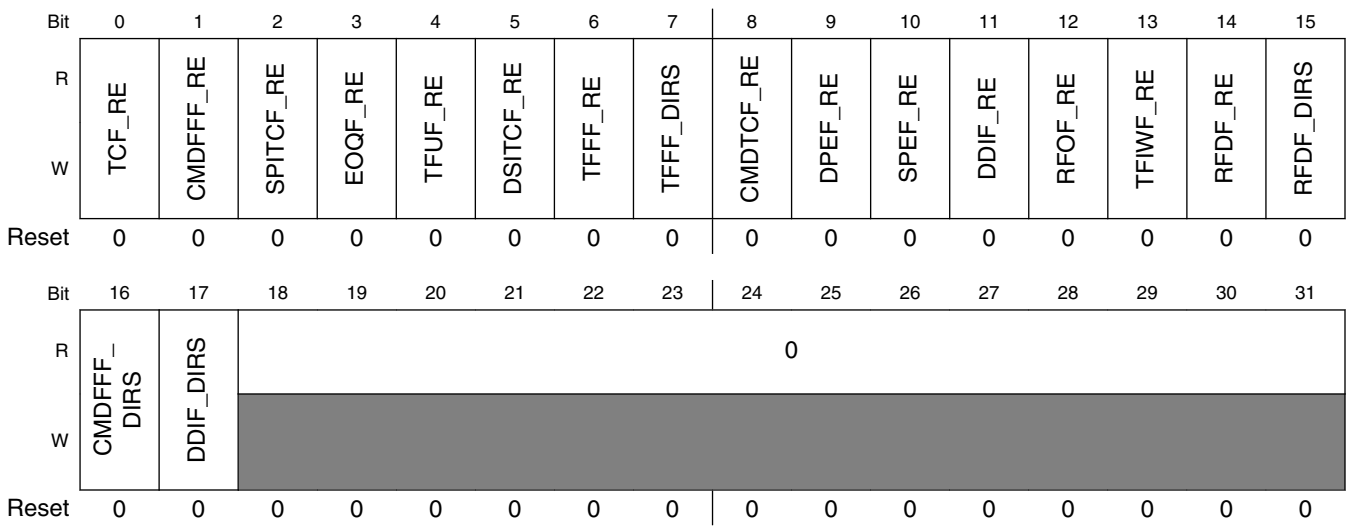
DSPI_SR field descriptions (continued)

Field	Description
	Provides a method for the module to request that entries be removed from the RX FIFO. The bit is set while the RX FIFO is not empty. The RFDF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller when the RX FIFO is empty. 0 RX FIFO is empty. 1 RX FIFO is not empty.
15 CMDFFF	Command FIFO Fill Flag Provides a method for the module to request more entries to be added to the CMD FIFO. The CMDFFF bit is set while the CMD FIFO is not full. The CMDFFF is cleared by writing a '1' to it or by acknowledgement from the DMA controller to the CMD FIFO full request. 0 CMD FIFO is full. 1 CMD FIFO is not full.
16–19 TXCTR	TX FIFO Counter Indicates the number of valid entries in the TX FIFO. The TXCTR is incremented every time the PUSHR is written. The TXCTR is decremented every time an SPI command is executed and the SPI data is transferred to the shift register.
20–23 TXNXTPTR	Transmit Next Pointer Indicates which TX FIFO entry is transmitted during the next transfer. The TXNXTPTR field is updated every time SPI data is transferred from the TX FIFO to the shift register.
24–27 RXCTR	RX FIFO Counter Indicates the number of entries in the RX FIFO. The RXCTR is decremented every time the POPR is read. The RXCTR is incremented every time data is transferred from the shift register to the RX FIFO.
28–31 POPNXTPTR	Pop Next Pointer Contains a pointer to the RX FIFO entry to be returned when the POPR is read. The POPNXTPTR is updated when the POPR is read.

48.4.6 DMA/Interrupt Request Select and Enable Register (DSPI_RSER)

RSER controls DMA and interrupt requests. Do not write to the RSER while the module is in the Running state.

Address: 0h base + 30h offset = 30h



DSPI_RSER field descriptions

Field	Description
0 TCF_RE	Transmission Complete Request Enable Enables TCF flag in the SR to generate an interrupt request. 0 TCF interrupt requests are disabled. 1 TCF interrupt requests are enabled.
1 CMDFFF_RE	Command FIFO Fill Flag Request Enable. Enables the CMDFFF flag in the SR to generate a request. The CMDFFF_DIRS bit selects between generating an interrupt request or a DMA request. 0 CMDFFF interrupts or DMA requests are disabled. 1 CMDFFF interrupts or DMA requests are enabled.
2 SPITCF_RE	SPI Frame Transmission Complete Request Enable. The SPITCF_RE bit enables SPITCF flag in the SR to generate an interrupt request. 0 SPITCF interrupt requests are disabled. 1 SPITCF interrupt requests are enabled.
3 EOQF_RE	Finished Request Enable

Table continues on the next page...

DSPI_RSER field descriptions (continued)

Field	Description
	Enables the EOQF flag in the SR to generate an interrupt request. 0 EOQF interrupt requests are disabled. 1 EOQF interrupt requests are enabled.
4 TFUF_RE	Transmit FIFO Underflow Request Enable Enables the TFUF flag in the SR to generate an interrupt request. 0 TFUF interrupt requests are disabled. 1 TFUF interrupt requests are enabled.
5 DSITCF_RE	DSI Frame Transmission Complete Request Enable. The DSITCF_RE bit enables DSITCF flag in the SR to generate an interrupt request. 0 DSITCF interrupt requests are disabled. 1 DSITCF interrupt requests are enabled.
6 TFFF_RE	Transmit FIFO Fill Request Enable Enables the TFFF flag in the SR to generate a request. The TFFF_DIRS bit selects between generating an interrupt request or a DMA request. 0 TFFF interrupts or DMA requests are disabled. 1 TFFF interrupts or DMA requests are enabled.
7 TFFF_DIRS	Transmit FIFO Fill DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. When SR[TFFF] and RSER[TFFF_RE] are set, this field selects between generating an interrupt request or a DMA request. 0 TFFF flag generates interrupt requests. 1 TFFF flag generates DMA requests.
8 CMDTCF_RE	Command Transmission Complete Request Enable. The CMDTCF_RE bit enables CMDTCF flag in the SR to generate an interrupt request. 0 CMDTCF interrupt requests are disabled. 1 CMDTCF interrupt requests are enabled.
9 DPEF_RE	DSI Parity Error Request Enable Enables the DPEF flag in the SR to generate an interrupt request. 0 DPEF interrupt requests are disabled. 1 DPEF interrupt requests are enabled.
10 SPEF_RE	SPI Parity Error Request Enable Enables the SPEF flag in the SR to generate an interrupt request. 0 SPEF interrupt requests are disabled. 1 SPEF interrupt requests are enabled.
11 DDIF_RE	DSI data received with active bits Request Enable Enables the DDIF flag in the SR to generate an interrupt or DMA request.

Table continues on the next page...

DSPI_RSER field descriptions (continued)

Field	Description
	0 DDIF interrupt and DMA requests are disabled. 1 DDIF interrupt and DMA requests are enabled.
12 RFOF_RE	Receive FIFO Overflow Request Enable Enables the RFOF flag in the SR to generate an interrupt request. 0 RFOF interrupt requests are disabled. 1 RFOF interrupt requests are enabled.
13 TFIWF_RE	Transmit FIFO Invalid Write Request Enable. Enables the TFIWF flag in the SR to generate an interrupt request. 0 TFIWF interrupt requests are disabled. 1 TFIWF interrupt requests are enabled.
14 RFDF_RE	Receive FIFO Drain Request Enable Enables the RFDF flag in the SR to generate a request. The RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 RFDF interrupt or DMA requests are disabled. 1 RFDF interrupt or DMA requests are enabled.
15 RFDF_DIRS	Receive FIFO Drain DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. When the RFDF flag bit in the SR is set, and the RFDF_RE bit in the RSER is set, the RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 Interrupt request. 1 DMA request.
16 CMDFFF_DIRS	Command FIFO Fill DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. When the CMDFFF flag bit in the SR is set, and the CMDFFF_RE bit in the RSER is set, the CMDFFF_DIRS bit selects between generating an interrupt request or a DMA request. 0 CMDFFF flag generates interrupt requests. 1 CMDFFF flag generates DMA requests.
17 DDIF_DIRS	DSI data received with active bits - DMA or Interrupt Request Select. Selects between generating a DMA request or an interrupt request. When the DDIF flag bit in the SR is set, and the DDIF_RE bit in the RSER is set, the DDIF_DIRS bit selects between generating an interrupt request or a DMA request. 0 Interrupt request. 1 DMA request.
18–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.7 PUSH TX FIFO Register In Master Mode (DSPI_PUSHR)

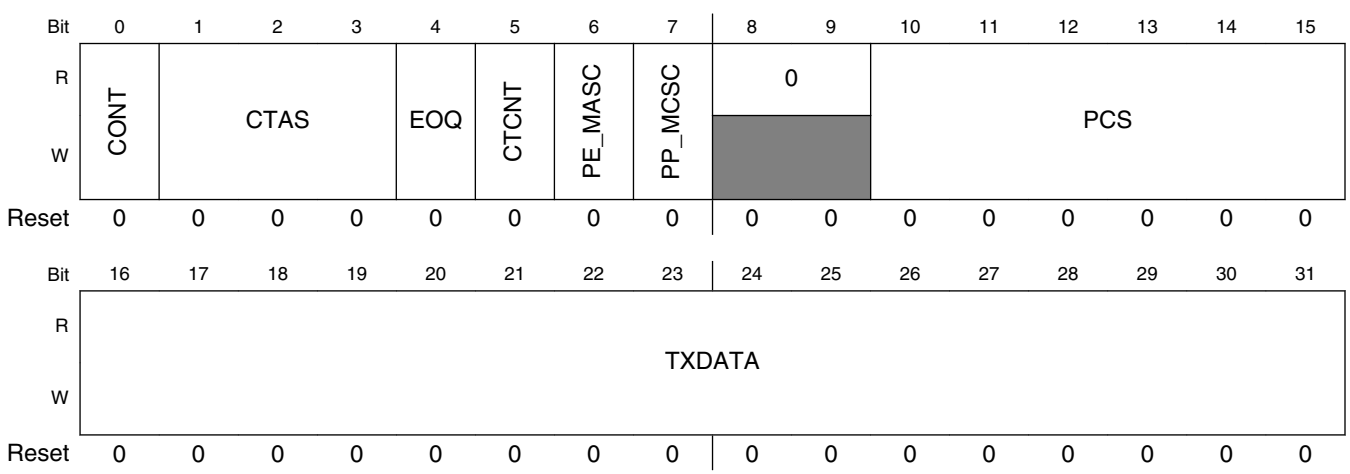
PUSHR provides the means to write to the TX FIFO and CMD FIFO . Data written to this register is transferred to the TX FIFO and CMD FIFO. 8- or 16-bit write accesses to the Data Field of PUSHR transfers the 16 bit Data field of PUSHR to the TX FIFO. Write accesses to the Command Field of PUSHR transfers the 16 bit Command Field of PUSHR to the CMD FIFO. The register structure is different in Master and Slave modes. In Master mode, the register provides 16-bit command to the CMD FIFO and 16-bit data to the TX FIFO. In Slave mode, CMD FIFO is unused and hence the 16 bit Command Field of PUSHR is reserved.

Unless Extended SPI Mode is being used, the TX FIFO and CMD FIFO must be filled simultaneously. In other words, write accesses must be given to both Data and Command fields of PUSHR for every PUSHR operation performed when MCR[XSPI] is 0. When Extended SPI Mode is disabled, since both TX FIFO and CMD FIFO are written to and read from simultaneously, they can be thought of as a single 32 bit FIFO. When Extended SPI Mode is enabled, TX FIFO and CMD FIFO can be written independently.

A PUSHR Read Operation returns the topmost TX FIFO and CMD FIFO entries concatenated.

When the module is disabled, any writes to this register will not update the FIFO. Hence any reads performed during Module disable mode will return the last PUSHR write performed when Module was enabled.

Address: 0h base + 34h offset = 34h



DSPI_PUSHR field descriptions

Field	Description
0 CONT	Continuous Peripheral Chip Select Enable

Table continues on the next page...

DSPI_PUSHR field descriptions (continued)

Field	Description
	Selects a continuous selection format. The bit is used in SPI Master mode. The bit enables the selected PCS signals to remain asserted between transfers. 0 Return PCSn signals to their inactive state between transfers. 1 Keep PCSn signals asserted between transfers.
1–3 CTAS	Clock and Transfer Attributes Select Selects which CTAR to use in master mode to specify the transfer attributes for the associated SPI frame. In SPI Slave mode, CTAR0 is used. See Number of CTARs to determine how many CTARs this device has. You should not program a value in this field for a register that is not present. 000 CTAR0 001 CTAR1 010 CTAR2 011 CTAR3 100 CTAR4 101 CTAR5 110 CTAR6 111 CTAR7
4 EOQ	End Of Queue Host software uses this bit to signal to the module that the current SPI transfer is the last in a queue. At the end of the transfer, the EOQF bit in the SR is set. 0 The SPI data is not the last data to transfer. 1 The SPI data is the last data to transfer.
5 CTCNT	Clear Transfer Counter Clears the TCNT field in the TCR register. The TCNT field is cleared before the module starts transmitting the current SPI frame. 0 Do not clear the TCR[TCNT] field. 1 Clear the TCR[TCNT] field.
6 PE_MASC	Parity Enable or Mask T_{ASC} delay in the current frame PE – This bit enables parity bit transmission and parity reception check for the SPI frame. MASC - The current frame has the “after SCK” delay masked if this bit is asserted. See Fast Continuous Selection Format for more details. **NOTE:** This bit is used as Mask T_{ASC} in the Fast Continuous PCS mode when MCR[FCPCS] is set. 0 PE - No parity bit included/checked. MASC - T_{ASC} delay is not masked and the current frame has the after SCK delay. 1 PE - Parity bit is transmitted instead of the last data bit in the frame; parity is checked for the received frame. MASC - T_{ASC} delay is masked in the current frame.
7 PP_MCSC	Parity Polarity or Mask T_{CSC} delay in the next frame PP - It controls the polarity of the parity bit transmitted and checked. MCSC - The next frame has the “PCS to SCK” delay masked if this bit is asserted. See Fast Continuous Selection Format for more details.

Table continues on the next page...

DSPI_PUSHR field descriptions (continued)

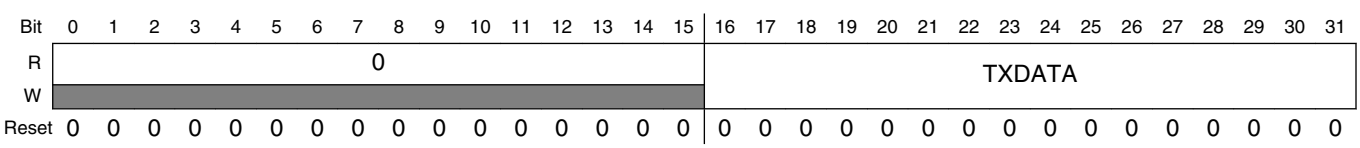
Field	Description
	NOTE: This bit is used as Mask T $_{CSC}$ in the Fast Continuous PCS mode when MCR[FCPCS] is set. 0 PP - Even Parity: the number of 1 bits in the transmitted frame is even. The SR[SPEF] bit is set if the number of 1 bits is odd in the received frame. MCSC - T $_{CSC}$ delay is not masked and the next frame has the PCS to SCK delay. 1 PP - Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[SPEF] bit is set if the number of 1 bits is even in the received frame. MCSC - T $_{CSC}$ delay is masked in the next frame.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 PCS	Select which PCS signals are to be asserted for the transfer. See Number of PCS signals for the number of PCS signals used in this MCU. 0 Negate the PCS[x] signal. 1 Assert the PCS[x] signal.
16–31 TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command.

48.4.8 PUSH TX FIFO Register In Slave Mode (DSPI_PUSHR_SLAVE)

PUSHR provides the means to write to the TX FIFO. Data written to this register is transferred to the TX FIFO. Eight- or sixteen-bit write accesses to the Data Field of PUSHR transfers the 16 bit Data Field of PUSHR to the TX FIFO. The register structure is different in master and slave modes. In master mode the register provides 16-bit command to the CMD FIFO and 16-bit data to the TX FIFO. In slave mode, CMD FIFO is unused and hence the 16 bit Command Field of PUSHR is reserved.

In addition, upto 32 bit SPI frames may be queued for transmission/reception by enabling MCR[XSPI] mode.

Address: 0h base + 34h offset = 34h



DSPI_PUSHR_SLAVE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command.

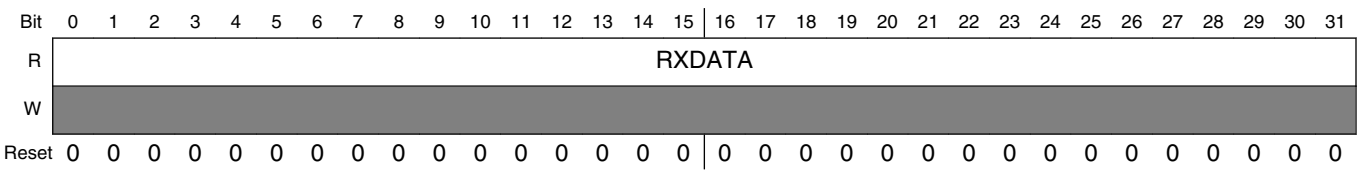
DSPI_PUSHR_SLAVE field descriptions (continued)

Field	Description

48.4.9 POP RX FIFO Register (DSPI_POPR)

POPR is used to read the RX FIFO. Eight- or sixteen-bit read accesses to the POPR have the same effect on the RX FIFO as 32-bit read accesses. A write to this register will generate a Transfer Error.

Address: 0h base + 38h offset = 38h



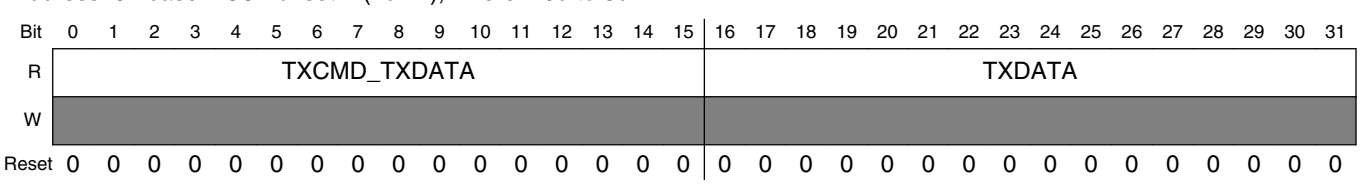
DSPI_POPR field descriptions

Field	Description
0–31 RXDATA	Received Data Contains the SPI data from the RX FIFO entry to which the Pop Next Data Pointer points.

48.4.10 Transmit FIFO Registers (DSPI_TXFR*n*)

TXFRn registers provide visibility into the TX FIFO for debugging purposes. Each register is an entry in the TX FIFO. The registers are read-only and cannot be modified. Reading the TXFRx registers does not alter the state of the TX FIFO. Reading TXFRn registers is invalid for a D SPI Master when being used in the Extended SPI (XSPI) Mode.

Address: 0h base + 3Ch offset + (4d × i), where i=0d to 3d



DSPI_TXFR*n* field descriptions

Field	Description
0–15 TXCMD_ TXDATA	Transmit Command or Transmit Data In Master mode the TXCMD field contains the command that sets the transfer attributes for the SPI data. In Slave mode, this field is reserved.
16–31 TXDATA	Transmit Data Contains the SPI data to be shifted out.

48.4.11 Receive FIFO Registers (DSPI_RXFR*n*)

RXFRn provide visibility into the RX FIFO for debugging purposes. Each register is an entry in the RX FIFO. The RXFR registers are read-only. Reading the RXFRx registers does not alter the state of the RX FIFO. The field MCR[MDIS] must be 0 when RXFR is read.

Address: 0h base + 7Ch offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXDATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_RXFR*n* field descriptions

Field	Description
0–31 RXDATA	Receive Data Contains the received SPI data.

48.4.12 DSI Configuration Register 0 (DSPI_DSICR0)

DSICR0 selects various attributes associated with DSI, TSB (and ITSB) configuration. Do not write to the DSICR0 while the module is in the Running state.

Address: 0h base + BCh offset = BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MTOE	FMSZ4	MTOCNT						FMSZ5	0	ITSB	TSBC	TXSS	0		CID
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DCONT	DSICTAS			DMS	PES	PE	PP	DPCSx							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_DSICR0 field descriptions

Field	Description
0 MTOE	Multiple Transfer Operation Enable Enables multiple DSPIs to be connected in a parallel or serial configuration. The MTOE and TSB bits should not be set simulteneously. 0 Multiple transfer operation disabled. 1 Multiple transfer operation enabled.
1 FMSZ4	MSB of the frame size in master mode. If the bit is set, 16 is added to the frame size, as defined by the CTARn[FMSZ] field. The CTARx register is selected by the DSICTAS field.
2–7 MTOCNT	Multiple Transfer Operation Count Selects the number of bits to be shifted out during a transfer in Multiple Transfer Operation. The field sets the number of SCK cycles that the bus master must generate to complete the transfer. The number of SCK cycles used is one more than the value in the MTOCNT field. The number of SCK cycles defined by MTOCNT must be equal to or greater than the frame size. When TSBC is set, MTOCNT has no effect.
8 FMSZ5	MSB of the frame size in master mode when DSI is used in 64-bit mode. If the bit is set, 32 is added to the frame size, as defined by the concatenation of {DSICR0[FMSZ[4]] , CTARn[FMSZ]} fields. The CTARn register is selected by the DSICTAS field. Hence the final concatenated frame size using this bit would be {DSICR0[FMSZ[5]], DSICR0[FMSZ[4]] , CTARn[FMSZ]}. This field is only valid when DSICR1[DSI64E] is enabled.

Table continues on the next page...

DSPI_DSICR0 field descriptions (continued)

Field	Description
9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 ITSB	Interleaved TSB mode. This bit enables the Interleaved TSB mode of operation. When enabled, there is no priority among frames from the SPI/DSI. DSI frames are always sent when either the previous transmission was a SPI frame or if the TX_FIFO is empty. Frames are transmitted on every trigger whose source is selected by the TRG bit setting. The ITSB mode requires the DSICR0[TSBC] bit to be set and should be written only when MCR[HALT] bit is asserted. If ITSB bit is set without setting DSICR0[TSBC], then the module operates in the Normal Mode depending on MCR[DCONF] bit. See Interleaved TSB (ITSB) Mode for more details. 0 Interleaved mode is disabled. 1 Interleaved mode is enabled.
11 TSBC	Timed Serial Bus Configuration. Enables the Timed Serial Bus Configuration, which allows 64-bit data to be used. It also allows tDT to be programmable. 0 Timed Serial Bus Configuration disabled. 1 Timed Serial Bus Configuration enabled.
12 TXSS	Transmit Data Source Select. Selects the source of data to be serialized. The source can be either data from host software written to the DSI Alternate Serialization Data Register (ASDR0/1) or Parallel Input pin states latched into the DSI Serialization Data Register (SDR0/1). 0 Source of serialized data is the SDR. 1 Source of serialized data is the ASDR.
13–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 CID	Change In Data Transfer Enable Enables a change in serialization data to initiate a DSI frames transfer in DSI and CSI configurations. When the CID bit is set, DSI frames are initiated when the current DSI data differs from the previous DSI data shifted out. **NOTE:** When TRIG[ENABLE] is set, the CID bit is given by TRIG[CID] which overrides DSICR0[CID].
16 DCONT	DSI Continuous Peripheral Chip Select Enable Enables the PCS signals to remain asserted between transfers. The DCONT bit affects the PCS signals only in DSI master mode. When the TSBC bit is set, DCONT has no effect. 0 Return Peripheral Chip Select signals to their inactive state after the transfer is complete. 1 Keep Peripheral Chip Select signals asserted after transfer is complete.
17–19 DSICTAS	DSI Clock and Transfer Attributes Select Selects the CTAR register to use to provide transfer attributes for DSI frames. The DSICTAS field is used in DSI master mode. In DSI slave mode, CTAR1 is always selected.
20 DMS	Data Match Stop If set, stops DSI frame transmissions if DDIF flag is set in the SR register.

Table continues on the next page...

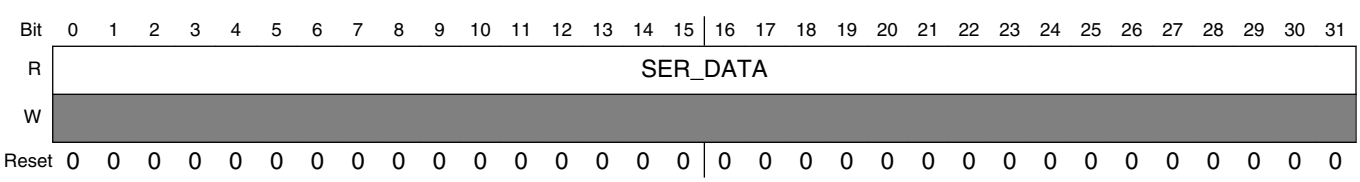
DSPI_DSICR0 field descriptions (continued)

Field	Description
	0 DDIF flag does not have an effect on DSI frame transmissions. 1 DDIF flag stops DSI frame transmissions.
21 PES	Parity Error Stop If set, stops DSI operation if the parity error happened in the received DSI frame. 0 Parity error does not stop DSI frame transmissions. 1 Parity error stops all DSI frame transmissions.
22 PE	Parity Enable Enables parity bit transmission and parity reception check for the DSI frames. 0 No parity bit included/checked. 1 Parity bit is transmitted instead of the last data bit in the frame; parity is checked for the received frame.
23 PP	Parity Polarity Controls the polarity of the parity bit transmitted and checked. 0 Even Parity: the number of 1 bits in the transmitted frame is even. The SR[DPEF] bit is set if in the received frame number of 1 bits is odd. 1 Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[DPEF] bit is set if in the received frame number of 1 bits is even.
24–31 DPCSx	DSI Peripheral Chip Select 0–7 Selects which of the PCS signals to assert during a DSI master mode transfer. 0 Negate PCS[x]. 1 Assert PCS[x].

48.4.13 DSI Serialization Data Register 0 (DSPI_SDR0)

SDR0 contains the states of the 32 LSB Parallel Input signals. The states of the Parallel Input signals are latched into the SDR0 on the rising edge of every bus clock. The SDR0 is read-only. When the TXSS bit in the DSICR0 is cleared, the data in the SDR0 is used as the source of the DSI frames. When DSICR1[DSI64E] is enabled, SDR1 forms the MSB half and SDR0 forms the LSB half of the 64-bit Serilization Data Register.

Address: 0h base + C0h offset = C0h



DSPI_SDR0 field descriptions

Field	Description
0–31 SER_DATA	Serialized Data Contains the signal states of the 32 LSB Parallel Input signals.

48.4.14 DSI Alternate Serialization Data Register 0 (DSPI_ASDR0)

ASDR0 is used by host software to write the 32 LSB of the data to be serialized. When the TXSS bit in the DSICR0 is set, the data in the ASDR0 is the source of the DSI frames. Writes to the ASDR0 take effect on the next frame boundary. When DSICR1[DSI64E] is enabled, ASDR1 forms the MSB half and ASDR0 forms the LSB half of the 64-bit Alternate Serilization Data Register.

Address: 0h base + C4h offset = C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ASER_DATA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_ASDR0 field descriptions

Field	Description
0–31 ASER_DATA	Alternate Serialized Data Holds the alternate 32 LSB of the data to be serialized.

48.4.15 DSI Transmit Comparison Register 0 (DSPI_COMPR0)

COMPR0 holds a copy of the 32 LSB of the last transmitted DSI data. COMPR0 is read-only. DSI data is transferred to this register as it is loaded into the TX Shift Register. When DSICR1[DSI64E] is enabled, COMPR1 forms the MSB half and COMPR0 forms the LSB half of the 64-bit Transmit Comparison Register.

Address: 0h base + C8h offset = C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COMP_DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

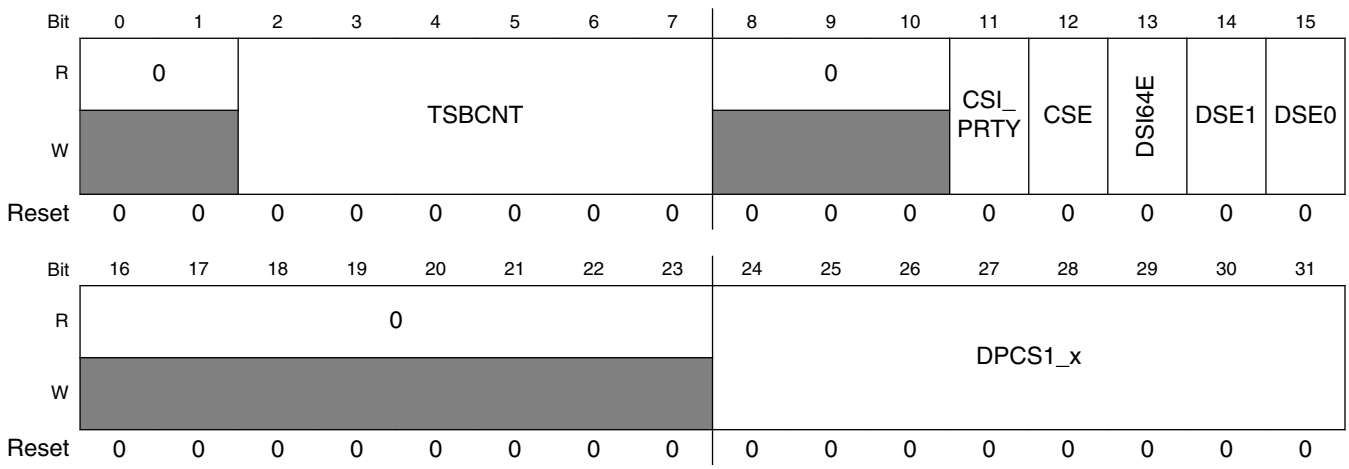
DSPI_COMPR0 field descriptions

Field	Description
0–31 COMP_DATA	Compare Data Holds the 32 LSB of the last serialized DSI data.

48.4.16 DSI Configuration Register 1 (DSPI_DSICR1)

DSICR1 selects various attributes associated with DSI, TSB (and ITSB) mode. The user must not write to the DSICR1 while the module is in the Running state.

Address: 0h base + D0h offset = D0h



DSPI_DSICR1 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 TSBCNT	Timed Serial Bus Operation Count When TSBC is set, TSBCNT defines the length of the data frame. The TSBCNT field valid value can vary from 3 to 63 when DSICR1[DSI64E] is set, otherwise this field can vary from 3 to 31. The TSBCNT field selects the number of data bits to be shifted out during a transfer in TSB mode. The number of data bits in the data frame is one more than the value in the TSBCNT field.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 CSI_PRTY	CSI Priority This field allows the user to configure the priority of SPI and DSI frames when module is operating in CSI Mode. This is not applicable when DSICR0[ITSB] is enabled. 0 SPI frames has priority over DSI frames. 1 DSI frames have priority over SPI frames.

Table continues on the next page...

DSPI_DSICR1 field descriptions (continued)

Field	Description
12 CSE	Command Select Enable When the DSICR0[ITSB] is set, the CSE bit controls insertion of command selection bit (Logic 1) in the beginning of the SPI (command) frame. 0 No selection bit is inserted in the beginning of the command frame. 1 Command selection bit is inserted at the beginning of the command frame. Total number of bits in the command frame is increased by 1.
13 DSI64E	DSI 64-bit Mode Enable When DSI64E is set, DSI Mode operates in 64-bit configuration allowing serialization and deserialization of Data frames up to 64 bits in size. The Extended Registers DSPI_SDR1, DSPI_ASDR1, DSPI_COMPR1, DSPI_DDR1, DSPI_SSR1, DSPI_DIMR1, DSPI_DPIR1 are only valid when this Mode is enabled. 0 DSI 64-bit Mode Disabled. DSI Mode operates in the default 32 bit configuration. 1 DSI 64-bit Mode Enabled. DSI Mode operates in 64-bit configuration.
14 DSE1	Data Select Enable 1 When the TBSC bit is set, the DSE1 bit controls insertion of the zero bit (Data Select) in the middle of the data frame. The insertion bit position is defined by the FMSZ field of CTARn register, selected by the DSICR0[DSICTAS] field. If DSICR1[DSI64] is enabled, then the insertion bit position is defined by concatenation of {DSICR0[FMSZ4], CTAR[FMSZ]} fields. The TSBCNT field value must be greater than the position of insertion for proper operation of the DSE1 bit. 0 No Zero bit is inserted in the middle of the data frame. 1 Zero bit is inserted at the middle of the data frame. Total number of bits in the data frame is increased by 1.
15 DSE0	Data Select Enable 0 When the TBSC bit is set, the DSE1 bit controls insertion of the zero bit (Data Select) in the beginning of the DSI frame. 0 No Zero bit is inserted in the beginning of the frame. 1 Zero bit is inserted at the beginning of the data frame. Total number of bits in the data frame is increased by 1.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 DPCS1_x	DSI Peripheral Chip Select 0–7 These bits define the PCSs to assert for the second part of the DSI frame when operating in TSB (and ITSB) configuration with dual receiver. The DPCS1 bits select which of the PCS signals to assert during the second part of the DSI frame. The DPCS1 bits control the assertions of the PCS signals only in TSB and ITSB mode. 0 Negate PCS[x]. 1 Assert PCS[x].

48.4.17 DSI Serialization Source Select Register 0 (DSPI_SSR0)

SSR0 is used to create a combined frame for transmission that contains bits from ASDR0 and from SDR0. Each bit in the SSR0 register selects a corresponding bit to be serialized. When DSICR0[TXSS] is set, the SSR0 register value has no effect. When DSICR1[DSI64E] is enabled, SSR1 forms the MSB half and SSR0 forms the LSB half of the 64-bit Serilization Source Select Register.

Address: 0h base + D4h offset = D4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_SSR0 field descriptions

Field	Description
0–31 SS	Source Select Select the serialization source for the 32 LSB of the DSI frame. Each SS bit selects data for a corresponded bit in the transmitted frame. 0 The bit in the transmitted frame is taken from the Parallel Input pin. 1 The bit in the transmitted frame is taken from the ASDR0 register

48.4.18 DSI Serialization Data Register 1 (DSPI_SDR1)

SDR1 contains the states of the 32 MSB parallel input signals. The states of these parallel input signals are latched into the SDR1 on the rising edge of every bus clock. The SDR1 is read-only. When the TXSS bit in the DSICR0 is cleared, the data in the SDR1 and SDR0 is used as the source of the DSI frames. The concatenation of {DSPIx_SDR1, DSPIx_SDR0} provides the 64-bit data to be serialized. This register is valid only when DSICR1[DSI64E] is enabled.

Address: 0h base + F0h offset = F0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SER_DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_SDR1 field descriptions

Field	Description
0–31 SER_DATA	Serialized Data Contains the 32 MSB signal states of the 64 parallel input signals. This field is valid only when DSICR1[DSI64E] is set. The concatenation of {DSPIx_SDR1, DSPIx_SDR0} provides the 64-bit data to be serialized.

48.4.19 DSI Alternate Serialization Data Register 1 (DSPI_ASDR1)

ASDR1 is used by host software to write the 32 MSB of the 64-bit data to be serialized. When the TXSS bit in the DSICR0 is set, the data in the ASDR1 and ASDR0 is the source of the DSI frames. Writes to the ASDR1 take effect on the next frame boundary. The concatenation of {DSPIx_ASDR1, DSPIx_ASDR0} provides the 64-bit alternate serialization data. This register is valid only when DSICR1[DSI64E] is enabled.

Address: 0h base + F4h offset = F4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ASER_DATA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

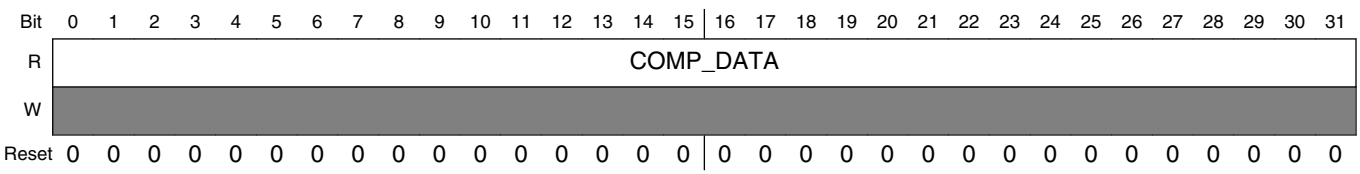
DSPI_ASDR1 field descriptions

Field	Description
0–31 ASER_DATA	Alternate Serialized Data Holds the 32 MSB of the 64-bit alternate data to be serialized. This field is valid only when DSICR1[DSI64E] is set. The concatenation of {DSPIx_ASDR1, DSPIx_ASDR0} provides the 64-bit alternate serialization data.

48.4.20 DSI Transmit Comparison Register 1 (DSPI_COMPR1)

COMPR1 holds a copy of the 32 MSB of the last transmitted 64-bit DSI data. COMPR1 is read-only. Upper 32 bits of the 64-bit DSI data is transferred to this register as it is loaded into the TX Shift Register. The concatenation of {DSPIx_COMPR1, DSPIx_COMPR0} provides the 64-bit transmit comparison data. This register is valid only when DSICR1[DSI64E] is enabled.

Address: 0h base + F8h offset = F8h



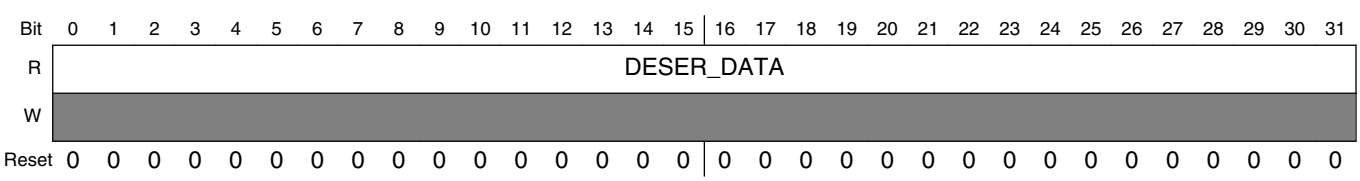
DSPI_COMPR1 field descriptions

Field	Description
0–31 COMP_DATA	Compare Data Holds the 32 MSB of the last serialized 64-bit DSI data. This field is valid only when DSICR1[DSI64E] is set. The concatenation of {DSPIx_COMPR1, DSPIx_COMPR0} provides the last serialized 64-bit DSI data.

48.4.21 DSI Deserialization Data Register 1 (DSPI_DDR1)

DDR1 holds the 32 MSB signal states for the 64 Parallel Output signals. The DDR1 is read-only and host software can read data from incoming DSI frames. The concatenation of {DSPIx_DDR1, DSPIx_DDR0} provides the 64-bit Deserialized data. This register is valid only when DSICR1[DSI64E] is enabled.

Address: 0h base + FCh offset = FCh



DSPI_DDR1 field descriptions

Field	Description
0–31 DESER_DATA	Deserialized Data Holds 32 MSB of the deserialized data that is presented as signal states to the 64 Parallel Output signals.This field is valid only when DSICR1[DSI64E] is set. The concatenation of {DSPIx_DDR1, DSPIx_DDR0} provides the 64-bit deserialized data.

48.4.22 DSI Serialization Source Select Register 1 (DSPI_SSR1)

SSR1 is used to create a combined frame for transmission that contains bits from ASDR1 and from SDR1. Each bit in the SSR1 register selects a corresponding bit to be serialized. When DSICR0[TXSS] is set, the SSR1 register value has no effect. The concatenation of {DSPIx_SSR1, DSPIx_SSR0} provides the 64-bit Serialization Source Select Value. This register is valid only when DSICR1[DSI64E] is enabled.

Address: 0h base + 100h offset = 100h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_SSR1 field descriptions

Field	Description
0–31 SS	Source Select Select the serialization source for the 32 MSB of the 64-bit DSI frame. Each SS bit selects data for a corresponded bit in the transmitted frame. This field is valid only when DSICR1[DSI64E] is set. The concatenation of {DSPIx_SSR1, DSPIx_SSR0} provides the 64-bit serialization source select value. 0 The bit in the transmitted frame is taken from the Parallel Input pin. 1 The bit in the transmitted frame is taken from the ASDR0/1 register

48.4.23 DSI Deserialized Data Interrupt Mask Register 1 (DSPI_DIMR1)

DIMR1 selects bits from the 32 MSB in the received 64-bit DSI frame to be checked to generate the DDI interrupt. The concatenation of {DSPIx_DIMR1, DSPIx_DIMR0} provides the 64-bit Deserialized Data Interrupt Mask Value. This register is valid only when DSICR1[DSI64E] is enabled.

Address: 0h base + 114h offset = 114h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MASK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_DIMR1 field descriptions

Field	Description
0–31 MASK	Mask Defines which of the 32 MSB in the received 64-bit deserialization data should be checked to produce the Deserialized Data Interrupt (DDI). 0 The bit in the received DSI frame does not produce a DDI interrupt. 1 The bit in the received DSI frame can produce a DDI interrupt if the data bit matches the configured polarity.

48.4.24 DSI Deserialized Data Polarity Interrupt Register 1 (DSPI_DPIR1)

DPIR1 defines which of the 32 MSB value in the received 64-bit DSI frame generates the DDI interrupt. The concatenation of {DSPIx_DPIR1, DSPIx_DPIR0} provides the 64-bit Deserialized Data Polarity Value. This register is valid only when DSICR1[DSI64E] is enabled.

Address: 0h base + 118h offset = 118h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_DPIR1 field descriptions

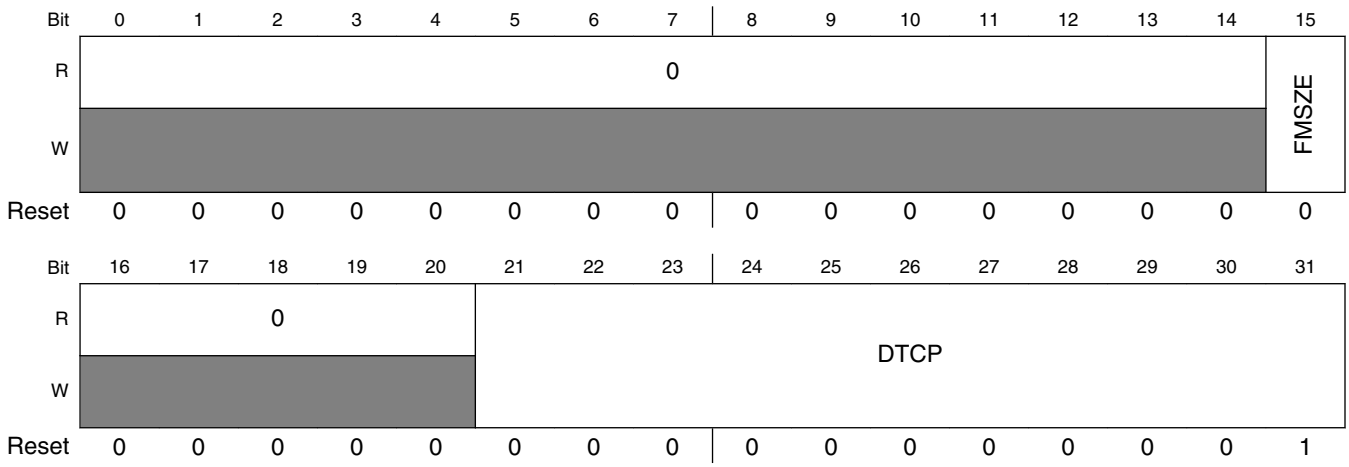
Field	Description
0–31 DP	Data Polarity Defines which of the 32 MSB of the received 64-bit deserialization data sets the SR[DDIF] bit. 0 If the received bit is 0, the SR[DDIF] bit is set. 1 If the received bit is 1, the SR[DDIF] bit is set.

48.4.25 Clock and Transfer Attributes Register Extended (DSPI_CTARE*n*)

CTARE registers are used to define the extended transfer attributes for an SPI frame. These registers are valid only when DSPIx_MCR[XSPI] is set.

When the module is configured as an SPI master, the CTAS field in CMD FIFO entry selects which of the CTARE registers is used. When the module is configured as an SPI bus slave, the CTARE0 register is used.

Address: 0h base + 11Ch offset + (4d × i), where i=0d to 7d



DSPI_CTARE*n* field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 FMSZE	Frame Size Extended This field is valid only when DSPIx_MCR[XSPI] is set. This field concatenated with DSPIx_CTAR[FMSZ] defines the Frame size of the SPI frames to be transmitted. Effective frame size would be the concatenation of {DSPIx_CTARE[FMSZE], DSPIx_CTAR[FMSZ]} plus 1.

Table continues on the next page...

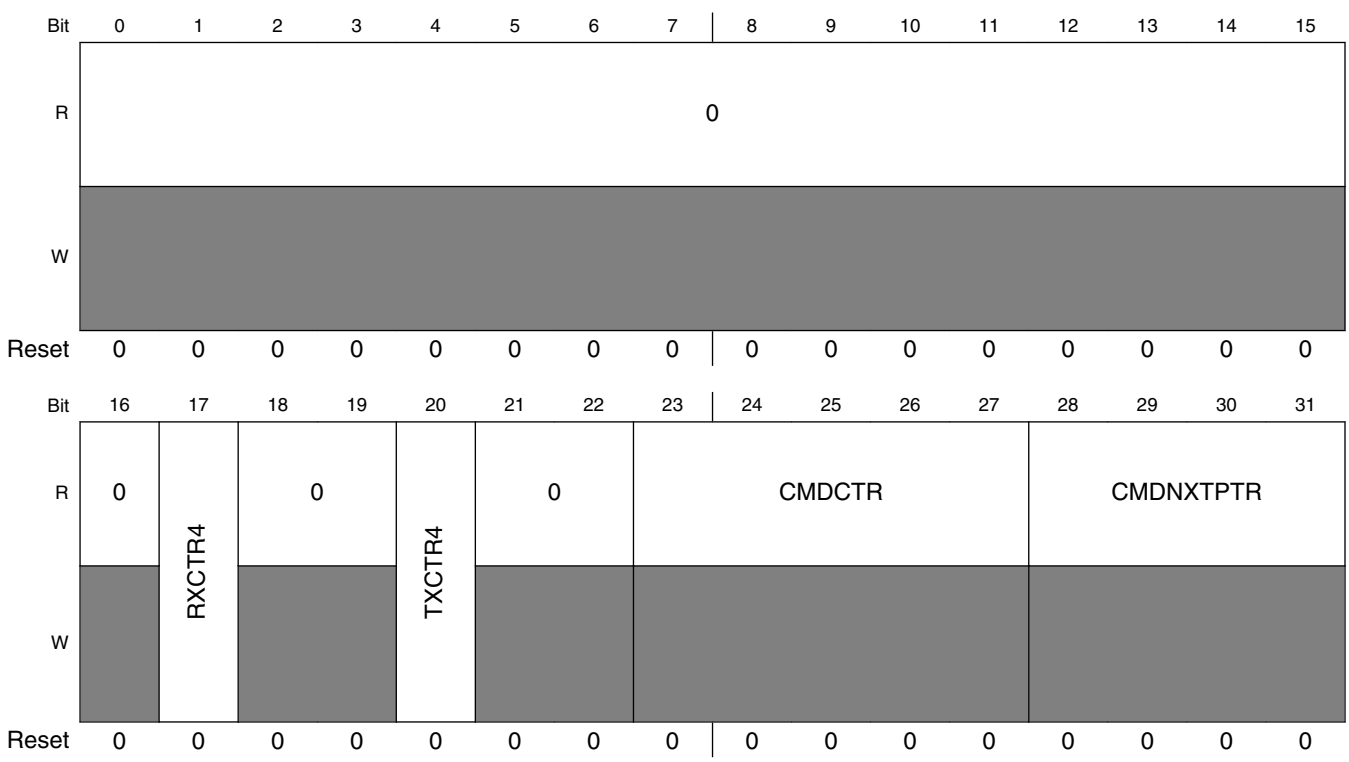
DSPI_CTARE*n* field descriptions (continued)

Field	Description
	0 Default Mode. Up to 16 bit SPI frames can be transfered. 1 Up to 32 bit SPI frames can be transfered. Each Frame transfer will be a result of 2 simulataneous TX FIFO Pop Operation.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–31 DTCP	Data Transfer Count Preload This field is valid only when DSPIx_MCR[XSPI] is set. This field defines the number of data frames (whose size is defined by CTARE[FMSZE] and CTAR[FMSZ]) to be transmitted using the Command frame which selected this DSPIx_CTARE register. The value 0 is reserved and should not be written in this field. The default value of this field is 1.

48.4.26 Status Register Extended (DSPI_SREX)

The register contains status fields. The fields reflect the status of the module and indicate the occurrence of events. This register is not writable.

Address: 0h base + 13Ch offset = 13Ch



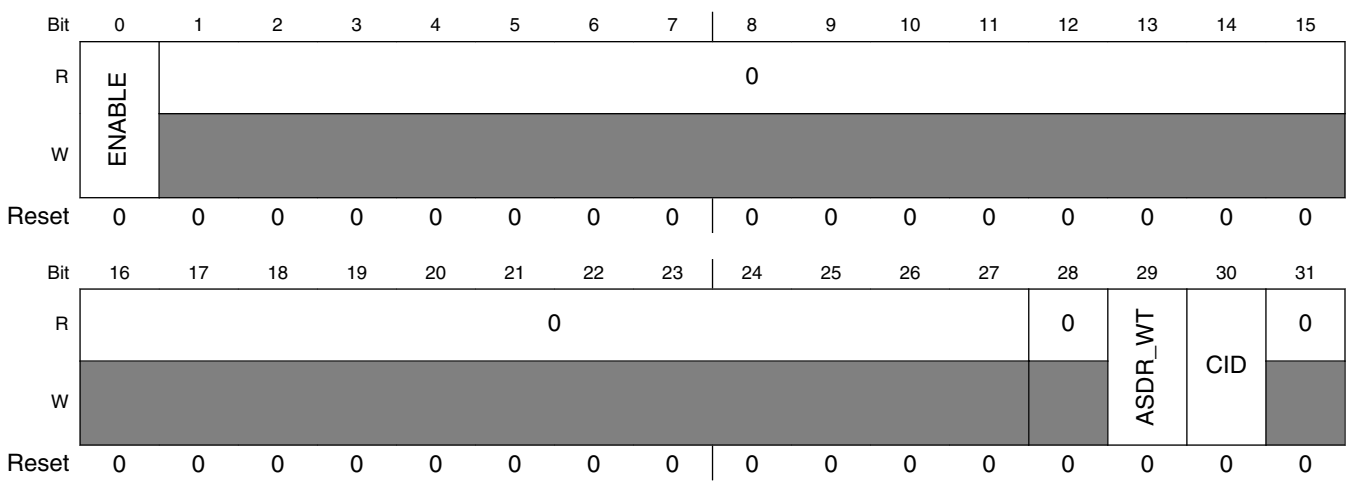
DSPI_SREX field descriptions

Field	Description
0–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 RXCTR4	RX FIFO Counter[4] This bit is an extension of SR[RXCTR]. The concatenated field {RXCTR4, RXCTR} indicates the number of entries in the RX FIFO. This field is decremented every time the POPR is read. And this field is incremented every time data is transferred from the shift register to the RX FIFO.
18–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 TXCTR4	TX FIFO Counter[4] This bit is an extension of SR[TXCTR]. The concatenated field {TXCTR4, TXCTR} indicates the number of valid entries in the TX FIFO. This field is incremented every time the PUSHR is written. And this field is decremented every time an SPI command is executed and the SPI data is transferred to the shift register.
21–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–27 CMDCTR	CMD FIFO Counter Indicates the number of entries in the CMD FIFO. The CMDCTR is incremented every time the command part of PUSHR is written. The CMDCTR is decremented every time a SPI command is executed (all data frames due to current command frame have been transmitted).
28–31 CMDNXTPTR	Command Next Pointer Indicates which CMD FIFO Entry is used during the next transfer. The CMDNXTPTR field is updated every time SPI data due to current command have been transmitted.

48.4.27 Trigger Register (DSPI_TRIG)

This register provides the user to configure trigger settings for various modes. The trigger settings in this register are activated only if the bit TRIG[ENABLE] is set.

Address: 0h base + 140h offset = 140h



DSPI_TRIG field descriptions

Field	Description
0 ENABLE	Trigger Register Enable This bit is used to enable the TRIG register. 0 TRIG register is disabled. Trigger settings configured in this register have no effect on the module. 1 TRIG register is enabled. Trigger settings configured in this register are valid.
1–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 ASDR_WT	ASDR Write Trigger Enable Enables the module to initiate a DSI frames transfer on a write operation performed on ASDR0 register. 0 ASDR0 write trigger disabled. 1 ASDR0 write trigger enabled.
30 CID	Change In Data Transfer Enable Enables a change in serialization data to initiate a DSI frames transfer in DSI and CSI configurations. When the CID bit is set, DSI frames are initiated when the current DSI data differs from the previous DSI data shifted out. **NOTE:** When TRIG[ENABLE] is set, the CID bit is given by TRIG[CID] which overrides DSICR0[CID].
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.28 Time Slot Length Register (DSPI_TSL)

The TSL register is only valid when DSICR0[ITSB] is enabled and internal triggering mechanism is used. It provides the length of every time slot in terms of baud cycles.

Address: 0h base + 150h offset = 150h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	TS3_LEN							0	TS2_LEN						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	TS1_LEN							0	TS0_LEN						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_TSL field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

DSPI_TSL field descriptions (continued)

Field	Description
1–7 TS3_LEN	Time Slot 3 Length This field provides the length of TS3 in terms of baud cycles. It must be ensured that the programmed value is greater than or equal to the length of the frame to be transmitted during TS3.Time Slot3 Length = TS3_LEN+1.This time slot shall be skipped if programmed to 0.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 TS2_LEN	Time Slot 2 Length This field provides the length of TS2 in terms of baud cycles. It must be ensured that the programmed value is greater than or equal to the length of the frame to be transmitted during TS2.Time Slot 2 Length = TS2_LEN+1.This time slot shall be skipped if programmed to 0.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–23 TS1_LEN	Time Slot 1 Length This field provides the length of TS1 in terms of baud cycles. Time Slot 1 Length = TS1_LEN+1. Ensure that the programmed value is greater than or equal to the length of the frame to be transmitted during TS1. This time slot shall be skipped if programmed to 0.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 TS0_LEN	Time Slot 0 Length This field provides the length of TS0 in terms of baud cycles. Time Slot 0 Length = TS0_LEN+1. Ensure that the programmed value is greater than or equal to the length of the frame to be transmitted during TS0. This time slot shall be skipped if programmed to 0.

48.4.29 Time Slot Configuration Register (DSPI_TS_CONF)

This register provides the basic configuration options available for the triggered implementation of MSC protocol. This register is only valid when DSICR0[ITSB] is enabled. Some configurations in this register check for the validity of command frame. The following conditions must be true for a command frame to be valid : a) The previous transmitted frame should not be a command frame. b) Data required to transmit the command frame should be available.

Address: 0h base + 154h offset = 154h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TS3				TS2				TS1				TS0			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_TS_CONF field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–19 TS3	Time Slot 3 This field is used to program the frame rule to be followed during Time Slot 3. This field is only valid when DSICR0[ITSB] mode is enabled. 0000 PF - Passive Frame 0001 SRDF - Single Receiver Data Frame 0010 DRF1 - First part of Dual Receiver Frame 0011 DRF2 - Second part of Dual Receiver Frame 0100 CMD_PF - Command Frame if valid, else Passive Frame 0101 CMD_SRDF - Command Frame if valid, else Single Receiver Data Frame 0110 CMD_DRF1 - Command Frame if valid, else first half of Dual Receiver Data Frame 0111 CMD_DRF2 - Command Frame if valid, else second half of Dual Receiver Data Frame 1xxx Reserved
20–23 TS2	Time Slot 2 This field is used to program the frame rule to be followed during Time Slot 2. This field is only valid when DSICR0[ITSB] mode is enabled. 0000 PF - Passive Frame 0001 SRDF - Single Receiver Data Frame 0010 DRF1 - First part of Dual Receiver Frame 0011 DRF2 - Second part of Dual Receiver Frame 0100 CMD_PF - Command Frame if valid, else Passive Frame 0101 CMD_SRDF - Command Frame if valid, else Single Receiver Data Frame 0110 CMD_DRF1 - Command Frame if valid, else first half of Dual Receiver Data Frame 0111 CMD_DRF2 - Command Frame if valid, else second half of Dual Receiver Data Frame 1xxx Reserved
24–27 TS1	Time Slot 1 This field is used to program the frame rule to be followed during Time Slot 1. This field is only valid when DSICR0[ITSB] mode is enabled. 0000 PF - Passive Frame 0001 SRDF - Single Receiver Data Frame 0010 DRF1 - First part of Dual Receiver Frame 0011 DRF2 - Second part of Dual Receiver Frame 0100 CMD_PF - Command Frame if valid, else Passive Frame 0101 CMD_SRDF - Command Frame if valid, else Single Receiver Data Frame 0110 CMD_DRF1 - Command Frame if valid, else first half of Dual Receiver Data Frame 0111 CMD_DRF2 - Command Frame if valid, else second half of Dual Receiver Data Frame 1xxx Reserved
28–31 TS0	Time Slot 0 This field is used to program the frame rule to be followed during Time Slot 0. This field is only valid when DSICR0[ITSB] mode is enabled. 0000 PF - Passive Frame

Table continues on the next page...

DSPI_TS_CONF field descriptions (continued)

Field	Description
	0001 SRDF - Single Receiver Data Frame 0010 DRF1 - First part of Dual Receiver Frame 0011 DRF2 - Second part of Dual Receiver Frame 0100 CMD_PF - Command Frame if valid, else Passive Frame 0101 CMD_SRDF - Command Frame if valid, else Single Receiver Data Frame 0110 CMD_DRF1 - Command Frame if valid, else first half of Dual Receiver Data Frame 0111 CMD_DRF2 - Command Frame if valid, else second half of Dual Receiver Data Frame 1xxx Reserved

48.5 Functional description

The Deserial Serial Peripheral Interface (DSPI) block supports full-duplex, synchronous serial communications between MCUs and peripheral devices. The DSPI can also be used to reduce the number of pins required for I/O by serializing and deserializing up to 64 Parallel Input/Output signals. All communications are done with SPI-like protocol.

The DSPI has the following configurations

- The SPI Configuration in which the module operates as a basic SPI or a queued SPI.
- DSI Configuration in which the DSPI serializes Parallel Input signals or bits from memory mapped register.
- CSI Configuration in which the DSPI combines the functionality of the SPI and DSI configurations.

The DCONF field in the DSPI Module Configuration Register (MCR) determines the module Configuration. See Module Configuration Register (DSPI_MCR) for the DSPI configuration values.

The CTARn registers hold clock and transfer attributes. The SPI configuration allows to select which CTAR to use on a frame by frame basis by setting a field in the SPI command. The Extended SPI Mode (DSPIx_MCR[XSPI]) further allows the usage CTAREn (CTARn Extended) registers which allows the user to send multiple data frames using a single command frame.

The DSI configuration statically selects which CTAR to use. In CSI, Configuration priority logic determines if SPI data or DSI data is transferred and dictates what CTAR register is used for the data transfer. See Clock and Transfer Attributes Register (In

Master Mode) (DSPI_CTAR*n*) for information on the fields of CTAR registers. See Clock and Transfer Attributes Register Extended (DSPI_CTARE*n*) for information on the fields of CTARE registers.

Typical master to slave connections are shown in the following figure. When a data transfer operation is performed, data is serially shifted a predetermined number of bit positions. Because the modules are linked, data is exchanged between the master and the slave. The data that was in the master shift register is now in the shift register of the slave, and vice versa. At the end of a transfer, the Transfer Control Flag(TCF) bit in the Shift Register(SR) is set to indicate a completed frame transfer.

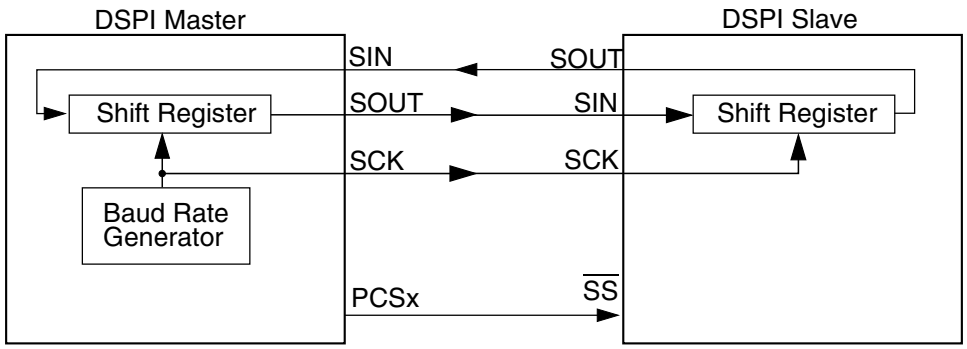


Figure 48-58. SPI and DSI serial protocol overview

Generally, more than one slave device can be connected to the module master. Six Peripheral Chip Select (PCS) signals of the module masters can be used to select which of the slaves to communicate with. Refer to Number of PCS signals for the number of PCS signals used in this MCU.

The three DSPI configurations share transfer protocol and timing properties which are described independently of the configuration in Transfer formats. The transfer rate and delay settings are described in Module baud rate and clock delay generation.

48.5.1 Start and Stop of module transfers

The module has two operating states: Stopped and Running. Both the states are independent of its configuration. The default state of the module is Stopped. In the Stopped state, no serial transfers are initiated in Master mode and no transfers are responded to in Slave mode. The Stopped state is also a safe state for writing the various configuration registers of the module without causing undetermined results. In the Running state serial transfers take place.

The TXRXS bit in the SR indicates the state of module. The bit is set if the module is in Running state.

The module starts or transitions to Running when all of the following conditions are true:

- SR[EOQF] bit is clear
- MCU is not in the Debug mode or the MCR[FRZ] bit is clear
- MCR[HALT] bit is clear

The module stops or transitions from Running to Stopped after the current frame when any one of the following conditions exist:

- SR[EOQF] bit is set
- MCU in the Debug mode and the MCR[FRZ] bit is set
- MCR[HALT] bit is set

State transitions from Running to Stopped occur on the next frame boundary if a transfer is in progress, or immediately if no transfers are in progress.

48.5.2 Serial Peripheral Interface (SPI) configuration

The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes. The module is in SPI configuration when the DCONF field in the MCR is 0b00. The SPI frames can be 32 bits long. The host CPU or a DMA controller transfers the SPI data from the external to DSPI RAM queues to a TX FIFO buffer. The received data is stored in entries in the RX FIFO buffer. The host CPU or the DMA controller transfers the received data from the RX FIFO to memory external to the module. The operation of FIFO buffers is described in Transmit First In First Out (TX FIFO) buffering mechanism, Transmit First In First Out (TX FIFO) buffering mechanism, Command First In First Out (CMD FIFO) Buffering Mechanism and Receive First In First Out (RX FIFO) buffering mechanism. The interrupt and DMA request conditions are described in Interrupts/DMA requests.

The SPI configuration supports two block-specific modes—Master mode and Slave mode. In Master mode the module initiates and controls the transfer according to the fields of the executing SPI Command. In Slave mode, the module responds only to transfers initiated by a bus master external to it and the SPI command field space is reserved.

48.5.2.1 Master mode

In SPI Master, mode the module initiates the serial transfers by controlling the SCK and the PCS signals. The executing SPI Command determines which CTARs will be used to set the transfer attributes and which PCS signals to assert. The command field also contains various bits that help with queue management and transfer protocol. See PUSH TX FIFO Register In Master Mode (DSPI_PUSHR) for details on the SPI command fields. The data in the executing TX FIFO entry is loaded into the shift register and shifted out on the Serial Out (SOUT) pin. In SPI Master mode, each SPI frame to be transmitted has a command associated with it, allowing for transfer attribute control on a frame by frame basis. In Extended SPI Master Mode, multiple SPI frames can have a single command associated with them allowing for efficient SPI frame transfers requiring common transfer attributes. In addition, the Extended SPI Mode allows for larger frame sizes of up to 32 bits.

48.5.2.2 Slave mode

In SPI Slave mode the module responds to transfers initiated by an SPI bus master. It does not initiate transfers. Certain transfer attributes such as clock polarity, clock phase, and frame size must be set for successful communication with an SPI master. The SPI Slave mode transfer attributes are set in the CTAR0 and CTARE0. The data is shifted out with MSB first. Shifting out of LSB is not supported in this mode.

48.5.2.3 FIFO disable operation

The FIFO disable mechanisms allow SPI transfers without using the TX FIFO, CMD FIFO or RX FIFO. The module operates as a double-buffered simplified SPI when the FIFOs are disabled. The Transmit and Receive side of the FIFOs are disabled separately; setting the MCR[DIS_TXF] bit disables the TX FIFO and CMD FIFO, and setting the MCR[DIS_RXF] bit disables the RX FIFO.

The FIFO disable mechanisms are transparent to the user and to host software. Transmit data and commands are written to the PUSHR and received data is read from the POPR.

When the TX FIFO is disabled (CMD FIFO is also disabled), the fields SR[TFFF], SR[TFUF], SR[CMDFFF], SREX[CMDCTR] and SR[TXCTR] behave as if there is a one-entry FIFO but the contents of TXFRs, SR[TXNXTPTR] and SREX[CMDNXTPTR] are undefined. Similarly, when the RX FIFO is disabled, the RFDF, RFOF, and RXCTR fields in the SR behave as if there is a one-entry FIFO, but the contents of the RXFR registers and POPNXTPTR are undefined.

48.5.2.4 Transmit First In First Out (TX FIFO) buffering mechanism

The TX FIFO functions as a buffer of SPI data for transmission. The TX FIFO holds four words, each consisting of SPI data. The number of entries in the TX FIFO is device-specific. SPI data is added to the TX FIFO by writing to the Data Field of module PUSH FIFO Register (PUSHR). TX FIFO entries can only be removed from the TX FIFO by being shifted out or by flushing the TX FIFO.

The TX FIFO Counter field (TXCTR) in the module Status Register (SR) indicates the number of valid entries in the TX FIFO. The TXCTR is updated every time a 8- or 16-bit write takes place to the Data Field of PUSHR or SPI data is transferred into the shift register from the TX FIFO.

The TXNXTPTR field indicates the TX FIFO Entry that will be transmitted during the next transfer. The TXFRn Registers are invalid in the Extended SPI Mode, since the TX FIFO and CMD FIFO can be used independently. The TXNXTPTR field is incremented every time SPI data is transferred from the TX FIFO to the shift register. The maximum value of the field is equal to the maximum implemented TXFR number and it rolls over after reaching the maximum.

48.5.2.4.1 Filling the TX FIFO

Host software or other intelligent blocks can add (push) entries to the TX FIFO and CMD FIFO by writing to the PUSHR. When the TX FIFO is not full, the TX FIFO Fill Flag (TFFF) in the SR is set. The TFFF bit is cleared when TX FIFO is full and the DMA controller indicates that a write to PUSHR is complete. Writing a '1' to the TFFF bit also clears it. The TFFF can generate a DMA request or an interrupt request. See Transmit FIFO Fill Interrupt or DMA Request for details.

The module ignores attempts to push data to a full TX FIFO, and the state of the TX FIFO does not change and no error condition is indicated.

48.5.2.4.2 Draining the TX FIFO

The TX FIFO entries are removed (drained) by shifting SPI data out through the shift register. Entries are transferred from the TX FIFO to the shift register and shifted out as long as there are valid entries in the TX FIFO. Every time an entry is transferred from the TX FIFO to the shift register, the TX FIFO Counter decrements by one. When Extended SPI Mode (DSPIx_MCR[XSPI]) is enabled, if the frame size of SPI Data to be transmitted is more than 16 bits, then it causes two Data entries to be popped from TX FIFO simultaneously which are transferred to the shift register. The first of the two popped entries forms the 16 least significant bits of the SPI frame to be transmitted. Such

an operation also causes TX FIFO Counter to decrement by two. At the end of a transfer, the TCF bit in the SR is set to indicate the completion of a transfer. The TX FIFO is flushed by writing a '1' to the CLR_TXF bit in MCR.

If an external bus master initiates a transfer with a module slave while the slave's TX FIFO is empty, the Transmit FIFO Underflow Flag (TFUF) in the slave's SR is set. See Transmit FIFO Underflow Interrupt Request for details.

48.5.2.5 Command First In First Out (CMD FIFO) Buffering Mechanism

The CMD FIFO functions as a buffer of SPI command used for SPI data transmission. When Extended SPI Mode (DSPIx_MCR[XSPI]) is disabled, the TX FIFO and CMD FIFO must be filled together, i.e. write enables should be given for both the Data and Command fields while performing a PUSHR operation. When Extended SPI Mode (DSPIx_MCR[XSPI]) is enabled, the TX FIFO and CMD FIFO can be filled independently.

The CMD FIFO holds 4 words, each representing SPI command fields. The number of entries in the CMD FIFO is device-specific. SPI Command is added to the CMD FIFO by writing to the command field of DSPI PUSH FIFO Register (PUSHR). CMD FIFO entries can only be removed from the CMD FIFO by being shifted out (to help transmit SPI data) or by flushing the CMD FIFO.

When Extended SPI Mode (DSPIx_MCR[XSPI]) is disabled, every CMDFIFO entry has a corresponding single TXFIFO entry attached to it since both these FIFO's are filled simultaneously.

When Extended SPI Mode (DSPIx_MCR[XSPI]) is enabled, every CMDFIFO entry can have multiple TXFIFO entries attached to it. Thus a single CMD FIFO entry can be used to transmit multiple TX FIFO entries. The CTARE[DTCP] field decides the number of SPI Data Frames having frame size as {FMSZE, FMSZ} to be transmitted using the current Command Entry. The CTAR/CTARE registers pointed by the CTAS field in the Command frame gives FMSZ and FMSZE fields respectively. The time for which a command entry is in use is known as a Command Cycle. The Busy Flag SR[BSYF] is asserted for the duration of the Command Cycle except for the last SPI frame in the Command Cycle.

The CMD FIFO Counter field (CMDCTR) in the DSPI Status Register (SR) indicates the number of valid entries in the CMD FIFO. The CMDCTR field is updated every time a 8- or 16-bit write takes place on the lower half of DSPI_PUSHR or SPI data is transferred into the shift register from the TX FIFO.

The TXFRn Registers are invalid in the Extended SPI Mode, since the TX FIFO and CMD FIFO can be used independently. The CMDNXTPTR field indicates which CMD FIFO Entry will be used during the next command cycle. The CMDNXTPTR field is incremented every time the last SPI data in the command cycle is transferred from the TX FIFO to the shift register and it rolls over after reaching the maximum.

48.5.2.6 Receive First In First Out (RX FIFO) buffering mechanism

The RX FIFO functions as a buffer for data received on the SIN pin. The RX FIFO holds four received SPI data frames. The number of entries in the RX FIFO is device-specific. SPI data is added to the RX FIFO at the completion of a transfer when the received data in the shift register is transferred into the RX FIFO. SPI data are removed (popped) from the RX FIFO by reading the module POP RX FIFO Register (POPR). RX FIFO entries can only be removed from the RX FIFO by reading the POPR or by flushing the RX FIFO.

The RX FIFO Counter field (RXCTR) in the module's Status Register (SR) indicates the number of valid entries in the RX FIFO. The RXCTR is updated every time the POPR is read or SPI data is copied from the shift register to the RX FIFO.

The POPNXTPTR field in the SR points to the RX FIFO entry that is returned when the POPR is read. The POPNXTPTR contains the positive offset from RXFR0 in a number of 32-bit registers. For example, POPNXTPTR equal to two means that the RXFR2 contains the received SPI data that will be returned when the POPR is read. The POPNXTPTR field is incremented every time the POPR is read. The maximum value of the field is equal to the maximum implemented RXFR number and it rolls over after reaching the maximum.

48.5.2.6.1 Filling the RX FIFO

The RX FIFO is filled with the received SPI data from the shift register. While the RX FIFO is not full, SPI frames from the shift register are transferred to the RX FIFO. Every time an SPI frame is transferred to the RX FIFO, the RX FIFO Counter is incremented by one.

If the RX FIFO and shift register are full and a transfer is initiated, the RFOF bit in the SR is set indicating an overflow condition. Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

48.5.2.6.2 Draining the RX FIFO

Host CPU or a DMA can remove (pop) entries from the RX FIFO by reading the module POP RX FIFO Register (POPR). A read of the POPR decrements the RX FIFO Counter by one. Attempts to pop data from an empty RX FIFO are ignored and the RX FIFO Counter remains unchanged. The data, read from the empty RX FIFO, is undetermined.

When the RX FIFO is not empty, the RX FIFO Drain Flag (RFDF) in the SR is set. The RFDF bit is cleared when the RX_FIFO is empty and the DMA controller indicates that a read from POPR is complete or by writing a 1 to it.

48.5.3 Deserial Serial Interface (DSI) Configuration

The DSI Configuration supports pin count reduction by serializing Parallel Input signals or register bits and shifting them out in a SPI-like protocol. The timing and transfer protocol is described in Transfer formats. The received serial frames are converted to a parallel form (deserialized) and placed on the Parallel Output signals or in the DSPI_DDR register. The various features of the DSI Configuration are set in DSPI DSI Configuration Registers (DSPI_DSICR0/1).

The DSI frames can be from 4 to 32 bits. This can go up to 64 bits when DSICR1[DSI64E] is enabled. With Multiple Transfer Operation (MTO) the DSPI supports serial chaining of DSPI blocks within the device to create DSI frames up to 64 bits, consisting of concatenated bits from multiple DSPIs. The DSPI also supports parallel chaining allowing several DSPIs and off-chip SPI devices to share the same Serial Communications Clock (SCK) and Peripheral Chip Select (PCS) signals. See Multiple Transfer Operation (MTO) for details on the serial and parallel chaining support.

Multiple Transfer Operation (MTO) and DSICR1[DSI64E] are not supported together and cannot be enabled simultaneously.

48.5.3.1 DSI Master Mode

In DSI master mode the DSPI initiates and controls the DSI transfers. The DSI master has eight different conditions that can initiate a transfer:

- Continuous
- Change in data
- Trigger signal

- ASDR write
- Trigger signal combined with a change in data
- Trigger signal combined with ASDR write
- Change in data combined with ASDR write
- Trigger signal combined with change in data combined with ASDR write

These transfer initiation conditions are described in DSI Transfer Initiation Control. Transfer attributes are set during initialization. The DSICTAS field in the DSPI_DSICR0 determines which of the DSPI_CTAR registers will control the transfer attributes.

48.5.3.2 Slave Mode

In DSI slave mode the DSPI responds to transfers initiated by a SPI or DSI bus master. In this mode the DSPI does not initiate DSI transfers. Certain transfer attributes such as clock polarity and phase must be set for successful communication with a DSI master. The DSI slave mode transfer attributes are set in the DSPI_CTAR1. The data is shifted out with MSB first.

If the CID bit in the DSPI_DSICR0 is set and the data in DSPI_COMPR0/1 differs from the selected source of the serialized data, the slave DSPI will assert the $\overline{\text{MTRIG}}$ signal. Also, if TRIG[ASDR_WT] is set and an ASDR0 write is detected, the slave DSPI will assert the $\overline{\text{MTRIG}}$ signal. These features are included to support chaining of several DSPI. Details about the $\overline{\text{MTRIG}}$ signal is found in Multiple Transfer Operation (MTO).

48.5.3.3 DSI Serialization

In the DSI Configuration, upto 64 bits can be serialized using two different sources. The TXSS bit in the DSPI_DSICR0 selects between the DSPI DSI Serialization Data Register (DSPI_SDR0/1) and the DSPI DSI Alternate Serialization Data Register (DSPI_ASDR0/1) as the source of the serialized data. The DSPI_SDR0 holds the latest 32 Parallel Input signal values which is sampled at every rising edge of the bus clock. When DSICR1[DSI64E] is enabled, 64-bit DSI Frames are supported and hence DSPI_SDR1 holds upto 32 most significant parallel input signal values which is sampled at every rising edge of the bus clock. The DSPI_ASDR0/1 register is written by host software and used as an alternate source of serialized data. (DSPI_ASDR1 is only usable when DSICR1[DSI64E] is enabled).

The DSPI has to sample the inputs (from SDR0/1 or ASDR0/1) before it transmits a DSI frame. Due to the asynchronous clocking structure within DSPI, the sampled data (from SDR0/1 or ASDR0/1) must be synchronized into the Protocol Domain before being loaded into the Shift Register for transmission.

To prevent Clock Domain Crossing issues, inputs from SDR0/1 or ASDR0/1 should not change while they are being sampled into the Protocol Domain. This is ensured in the following manner for the ASDR0 inputs. The initial write performed on ASDR0 register will cause the 32-bit ASDR0 input to be latched into a temporary register which is then locked (further latching is prevented) until this data gets synchronized into the Protocol Domain. Once synchronized, the lock is removed and the next ASDR0 write will again trigger this process. If there was a write performed on ASDR0 while the lock was asserted, this will be would be taken into consideration as soon as the previous synchronization process completes and lock gets de-asserted. In case multiple writes were performed on ASDR0 when lock was asserted, only the latest value will be considered for synchronization once the lock gets de-asserted. This process is true for ASDR1 register as well.

The entire process explained above is also true for the SDR0 and SDR1 inputs, the only difference being that SDR0/1 inputs are delayed internally by a single Bus clock (register interface clock) to detect a change in the parallel input pins.

Once the synchronization process for a particular De-serial input begins, it takes up to 5 Bus clock (register interface clock) periods and 5 Protocol clock periods before the synchronization process for the next De-serial input can begin. Hence the writes performed on ASDR0/1 and SDR0/1 are constrained with this timing.

Note

The Parallel Input pin state, to be transmitted, should be selected by TXSS or DSPI_SSR0/1 register and the frame size should be higher than the bit position in the preselected frame.

DSPI_SSR0/1 register provides additional way to create the frame for transmission. Each bit from this register is OR'd with the TXSS bit and controls individual transmitted bit source. This way, the transmitted frame can have any combination of the DSPI_SDR0/1 and DSPI_ASDR0/1 bits. This feature allows control SPI based devices, requiring control and data fields in the frame. Control field may come from DSPI_ASDR0/1 register, set by the device's CPU, while data field can be generated by device peripheral modules, like PWM timers.

A copy of the last DSI frame shifted out of the Shift Register is stored in the DSPI DSI Transmit Comparison Registers (DSPI_COMPR0/1). This register provides added visibility for debugging and it serves as a reference for transfer initiation control. The following figure shows the DSI Serialization logic.

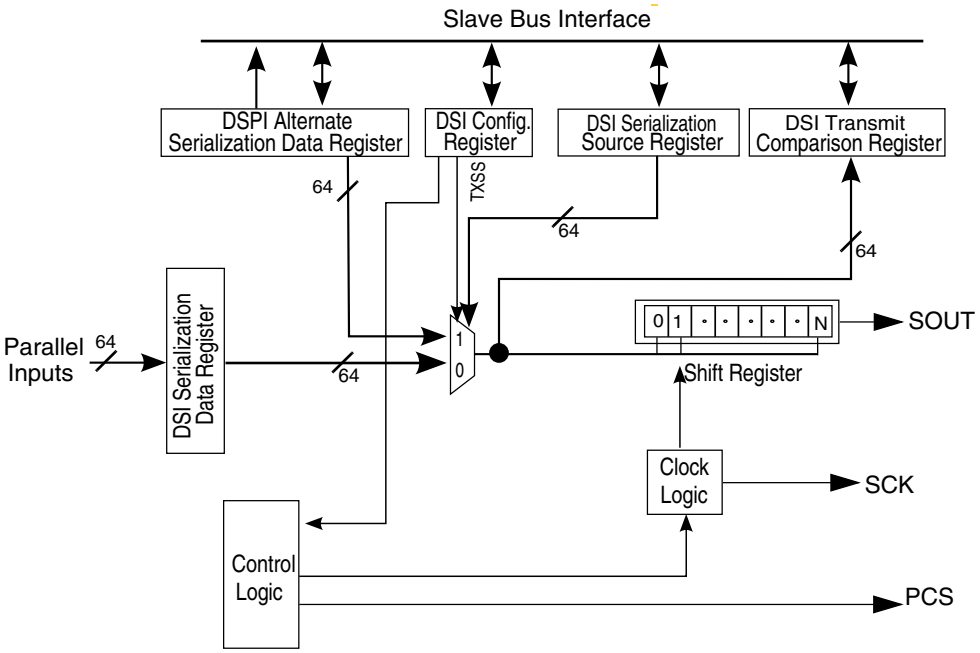


Figure 48-59. DSI Serialization Diagram

48.5.3.4 DSI Transfer Initiation Control

Data transfers for a DSPI in DSI configuration are initiated by a condition. The transfer initiation conditions are selected by the TRRE and CID bits in the DSPI_DSICR0.

NOTE

By default, the TRRE and CID bits are programmed using DSICR0[TRRE] and DSICR0[CID]. When TRIG[ENABLE] is set, the TRRE and CID bits are given by TRIG[TRRE] and TRIG[CID] respectively and the bits DSICR0[TRRE], DSICR0[CID] have no effect.

The following table lists the transfer initiation conditions.

Table 48-89. DSI Data Transfer Initiation Control

TRRE	CID	ASDR_WT	Transfer Initiation Control
0	0	0	Continuous
0	0	1	ASDR Write
0	1	0	Change in Data
0	1	1	Change in Data or ASDR Write
1	0	0	Triggered
1	0	1	Triggered or ASDR Write
1	1	0	Triggered or Change in Data

Table continues on the next page...

Table 48-89. DSI Data Transfer Initiation Control (continued)

TRRE	CID	ASDR_WT	Transfer Initiation Control
1	1	1	Triggered or Change in Data or ASDR Write

48.5.3.4.1 Continuous Control

For Continuous Control a new DSI frame shifts out when the previous transfer cycle has completed and the Delay after Transfer (t_{DT}) has elapsed. This has no effect when DSPI is in Slave Mode.

48.5.3.4.2 ASDR Write Control

For ASDR Write Control, a DSI frame shifts out whenever a write operation to ASDR0 is detected. For ASDR_WT to have any affect, TRIG[ASDR_WT] bit must be set. Note that ASDR0 write operations will not generate ASDR write triggers when the module is in STOPPED state, including when MCR[HALT] is asserted.

NOTE

If a DSI frame > 32 bits needs to be transmitted, the user must update ASDR1 before updating ASDR0.

48.5.3.4.3 Change In Data Control

For Change in Data Control a transfer is initiated when the data to be serialized has changed since the transfer of the last DSI frame. A copy of the previously transferred DSI data is stored in DSPI_COMPR0/1. When the data, selected for the transfer from the DSPI_SDR0/1 and DSPI_ASDR0/1 registers is different from the data in the DSPI_COMPR0/1 a new DSI frame is transmitted. The $\overline{\text{MTRIG}}$ output signal is asserted every time a change in data is detected.

NOTE

It is not advisable to configure a DSPI Slave with the combination DSICR0[DCONT] = 1, DSICR0[CID] = 1 ; since the following statement then holds true.

Once slave select is asserted, there is a very small window (of up to 3 module clocks) after every SOF (Start of Frame) during which if a write on ASDR/SDR registers occur, the next frame to be transmitted would be loaded into the shift register.

However, it is not feasible that such a write is initiated and completed within this short span of time; hence the data transmitted by DSPI slave would lag by one data frame.

48.5.3.4.4 Change In Data or ASDR Write Control

For Change in Data or ASDR Write Control, initiation of a transfer is controlled by the detection of a change in data to be serialized or whenever a write operation to ASDR0 is detected.

NOTE

For any DSI transfer initiation combination where TRIG[ASDR_WT] and DSICR0[HT] are both enabled, it is possible that the ASDR0 write operation causes a Change of DSI Data. This special condition results in transmission of a single DSI frame even though 2 separate triggers were generated.

48.5.4 Combined Serial Interface (CSI) Configuration

The CSI Configuration of the DSPI is used to support SPI and DSI functions on a frame by frame basis. CSI Configuration allows interleaving of DSI data frames from the parallel input signals with SPI data frames from the TX FIFO. The data returned from the bus slave is either used to drive the Parallel Output signals or it is stored in the RX FIFO. The CSI Configuration allows serialized data and configuration or diagnostic data to be transferred to a slave device using only one serial link. The DSPI is in CSI Configuration when the DCONF field in the DSPI_MCR is 0b10. The following figure shows an example of how a DSPI can be used with a deserializing peripheral that supports SPI control for control and diagnostic frames.

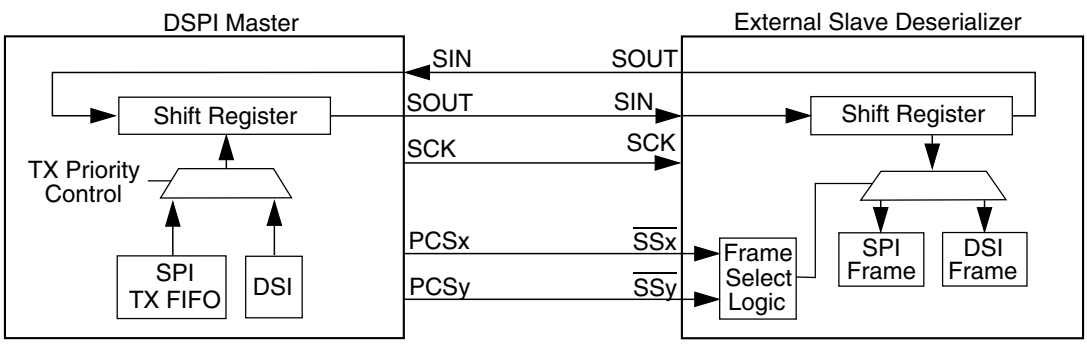


Figure 48-60. Example of System using DSPI in CSI Configuration

In CSI Configuration the DSPI transfers DSI data based on DSI Transfer Initiation Control.

When DSICR1[PRTY] is set to zero and there are SPI commands in the TX FIFO, the SPI frames have priority over the DSI frames. When the TX FIFO or CMD FIFO is empty, DSI transfer resumes.

When DSICR1[PRTY] bit is set, the DSI frames have priority over SPI frames. Thus, only when the DSI frame initiation requests have been catered to, will the SPI frames be transmitted, if available.

NOTE

The following combination is not a legal scenario and should be avoided - DSICR1[PRTY] = 1, DSICR1[TRRE] = 0 and DSICR1[CID] = 0, since such a configuration prevents SPI frames from ever getting transmitted.

Two peripheral chip select signals indicate whether DSI data or SPI data is transmitted. The user must configure the DSPI so that the two CTAR registers associated with DSI data and SPI data assert different peripheral chip select signals denoted in the figure as PCSx and PCSy. The CSI Configuration is only supported in master mode.

Data returned from the external slave while a DSI frame is transferred is placed on the Parallel Output signals. Data returned from the external slave while a SPI frame is transferred is moved to the RX FIFO. The TX FIFO, CMD FIFO and RX FIFO are fully functional in CSI mode.

48.5.4.1 CSI Serialization

Serialization in the CSI configuration is similar to serialization in DSI Configuration. The transfer attributes for SPI frames are determined by the DSPI_CTAR register selected by the CTAS field in the SPI command halfword. The transfer attributes for the DSI frames are determined by the DSPI_CTAR register selected by the DSICTAS field in the DSPI_DSICR0.

The Parallel Inputs signal states are latched into the DSPI DSI Serialization Data Register (DSPI_SDR0/1) and serialized based on the transfer initiation control settings in the DSPI_DSICR0. When SPI frames are written to the TX FIFO (and when CMD FIFO is not empty) they have priority over DSI data from the DSPI_SDR and are transferred at the next frame boundary. A copy of the most recently transferred DSI frame is stored in the DSPI_COMPR0/1. The Transfer Priority Logic selects the source of the serialized data and asserts the appropriate PCS signal.

Separate frame completion interrupts are available to indicate frame transmission completion from SPI and DSI frames.

48.5.4.2 CSI Deserialization

The deserialized frames in CSI Configuration goes into the DSPI_DDR0/1 or the RX FIFO based on the transfer priority logic. When DSI frames are transferred the returned frames are deserialized and latched into the DSPI_DDR0/1. When SPI frames are transferred the returned frames are deserialized and written to the RX FIFO.

48.5.5 Module baud rate and clock delay generation

The SCK frequency and the delay values for serial transfer are generated by dividing the protocol clock frequency by a prescaler and a scaler with the option for doubling the baud rate. The following figure shows conceptually how the SCK signal is generated.

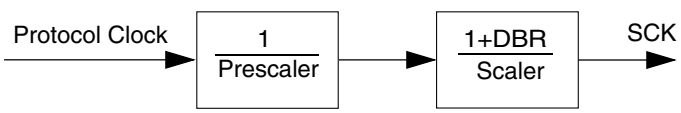


Figure 48-61. Communications clock prescalers and scalers

48.5.5.1 Baud rate generator

The baud rate is the frequency of the SCK. The protocol clock is divided by a prescaler (PBR) and scaler (BR) to produce SCK with the possibility of halving the scaler division. The DBR, PBR, and BR fields in the CTARs select the frequency of SCK by the formula in the BR field description. The following table shows an example of how to compute the baud rate.

Table 48-90. Baud rate computation example

f_P	PBR	Prescaler	BR	Scaler	DBR	Baud rate
100 MHz	0b00	2	0b0000	2	0	25 Mb/s
20 MHz	0b00	2	0b0000	2	1	10 Mb/s

NOTE

The clock frequencies mentioned in the preceding table are given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

48.5.5.2 PCS to SCK Delay (t_{CSC})

The PCS to SCK delay is the length of time from assertion of the PCS signal to the first SCK edge. See Figure 48-63 for an illustration of the PCS to SCK delay. The PCSSCK and CSSCK fields in the CTAR*x* registers select the PCS to SCK delay by the formula in the CSSCK field description. The following table shows an example of how to compute the PCS to SCK delay.

Table 48-91. PCS to SCK delay computation example

f_P	PCSSCK	Prescaler	CSSCK	Scaler	PCS to SCK Delay
100 MHz	0b01	3	0b0100	32	0.96 µs

PCSCSK and CSSCK fields have no effect in TSB configuration.

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

48.5.5.3 After SCK Delay (t_{ASC})

The After SCK Delay is the length of time between the last edge of SCK and the negation of PCS. See Figure 48-63 and Figure 48-64 for illustrations of the After SCK delay. The PASC and ASC fields in the CTAR*x* registers select the After SCK Delay by the formula in the ASC field description. The following table shows an example of how to compute the After SCK delay.

Table 48-92. After SCK Delay computation example

f_P	PASC	Prescaler	ASC	Scaler	After SCK Delay
100 MHz	0b01	3	0b0100	32	0.96 µs

PCASC and ASC fields have no effect in TSB configuration.

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

48.5.5.4 Delay after Transfer (t_{DT})

The Delay after Transfer is the minimum time between negation of the PCS signal for a frame and the assertion of the PCS signal for the next frame. See Figure 48-63 for an illustration of the Delay after Transfer. The PDT and DT fields in the CTAR*x* registers select the Delay after Transfer by the formula in the DT field description. The following table shows an example of how to compute the Delay after Transfer.

Table 48-93. Delay after Transfer computation example

f_P	PDT	Prescaler	DT	Scaler	Delay after Transfer
100 MHz	0b01	3	0b1110	32768	0.98 ms

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

When in Non-Continuous Clock mode the t_{DT} delay is configured according to the equation specified in the CTAR[DT] field description. When in Continuous Clock mode and TSB is not enabled, the delay is fixed at 1 SCK period.

In TSB mode the Delay after Transfer is equal to a number formed by concatenation of PDT and DT fields plus 1 of the SCK clock periods. See detailed information on Timed Serial Bus (TSB)".

48.5.5.5 Peripheral Chip Select Strobe Enable ($\overline{\text{PCSS}}$)

The $\overline{\text{PCSS}}$ signal provides a delay to allow the PCS signals to settle after a transition occurs thereby avoiding glitches. When the Module is in Master mode and the PCSSE bit is set in the MCR, $\overline{\text{PCSS}}$ provides a signal for an external demultiplexer to decode the PCS[0] – PCS[4] signals into as many as 128 glitch-free PCS signals. The following figure shows the timing of the $\overline{\text{PCSS}}$ signal relative to PCS signals.

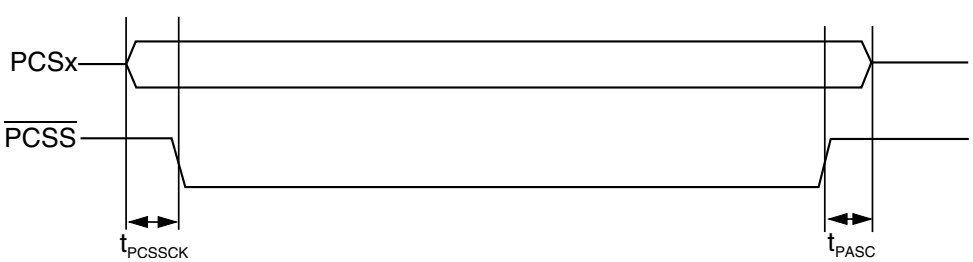


Figure 48-62. Peripheral Chip Select Strobe timing

The delay between the assertion of the PCS signals and the assertion of $\overline{\text{PCSS}}$ is selected by the PCSSCK field in the CTAR based on the following formula:

$$t_{\text{PCSSCK}} = \frac{1}{f_{\text{P}}} \times \text{PCSSCK}$$

At the end of the transfer, the delay between $\overline{\text{PCSS}}$ negation and PCS negation is selected by the PASC field in the CTAR based on the following formula:

$$t_{\text{PASC}} = \frac{1}{f_{\text{P}}} \times \text{PASC}$$

The following table shows an example of how to compute the t_{pcssck} delay.

Table 48-94. Peripheral Chip Select Strobe Assert computation example

f_P	PCSSCK	Prescaler	Delay before Transfer
100 MHz	0b11	7	70.0 ns

The following table shows an example of how to compute the t_{pasc} delay.

Table 48-95. Peripheral Chip Select Strobe Negate computation example

f_P	PASC	Prescaler	Delay after Transfer
100 MHz	0b11	7	70.0 ns

The $\overline{\text{PCSS}}$ signal is not supported when Continuous Serial Communication SCK or TSB mode is enabled.

NOTE

The clock frequency mentioned in the preceding tables is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

48.5.6 Transfer formats

The SPI serial communication is controlled by the Serial Communications Clock (SCK) signal and the PCS signals. The SCK signal provided by the master device synchronizes shifting and sampling of the data on the SIN and SOUT pins. The PCS signals serve as enable signals for the slave devices.

In Master mode, the CPOL and CPHA bits in the Clock and Transfer Attributes Registers (CTARn) select the polarity and phase of the serial clock, SCK.

- CPOL - Selects the idle state polarity of the SCK
- CPHA - Selects if the data on SOUT is valid before or on the first SCK edge

Even though the bus slave does not control the SCK signal, in Slave mode the values of CPOL and CPHA must be identical to the master device settings to ensure proper transmission. In SPI Slave mode, only CTAR0 is used. In DSI slave mode, only CTAR1 is used.

The module supports four different transfer formats:

- Classic SPI with CPHA=0
- Classic SPI with CPHA=1
- Modified Transfer Format with CPHA = 0
- Modified Transfer Format with CPHA = 1

A modified transfer format is supported to allow for high-speed communication with peripherals that require longer setup times. The module can sample the incoming data later than halfway through the cycle to give the peripheral more setup time. The MTFE bit in the MCR selects between Classic SPI Format and Modified Transfer Format.

In the SPI and DSI configurations, the module provides the option of keeping the PCS signals asserted between frames. See Continuous Selection Format for details.

48.5.6.1 Classic SPI Transfer Format (CPHA = 0)

The transfer format shown in following figure is used to communicate with peripheral SPI slave devices where the first data bit is available on the first clock edge. In this format, the master and slave sample their SIN pins on the odd-numbered SCK edges and change the data on their SOUT pins on the even-numbered SCK edges.

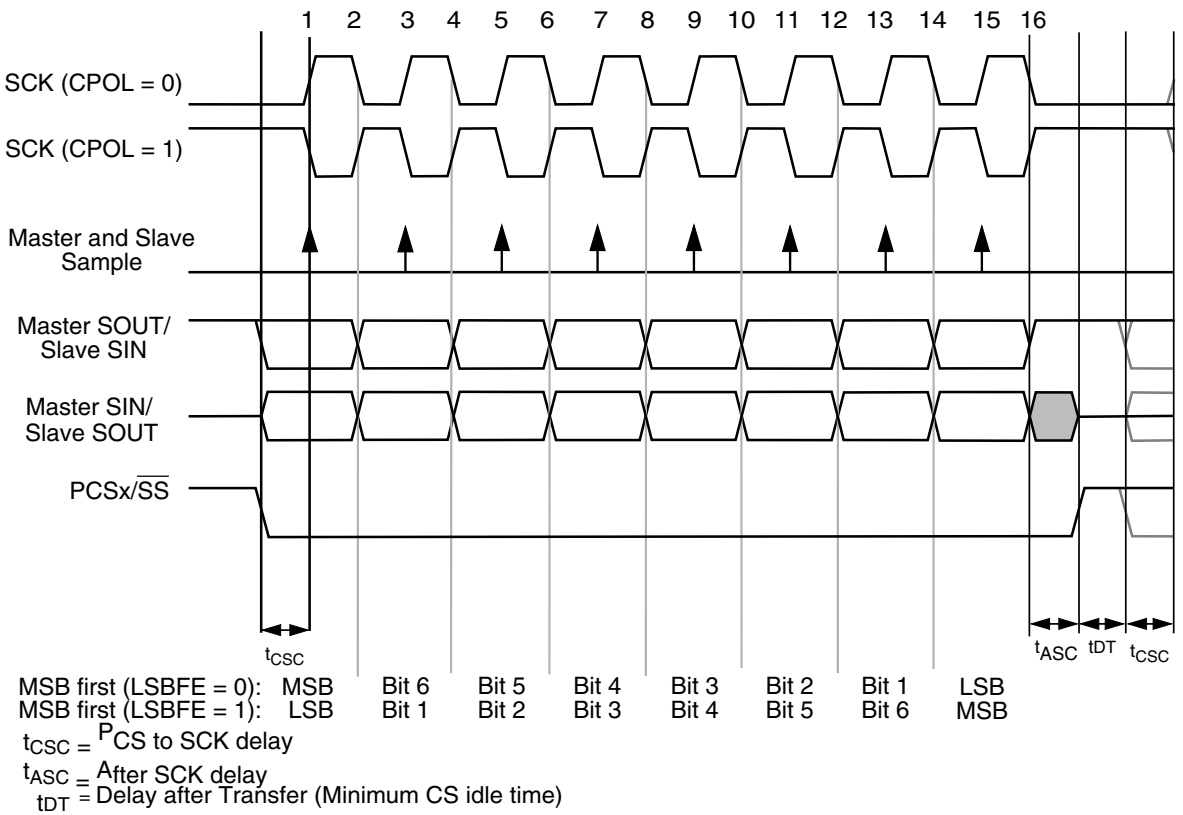


Figure 48-63. Module transfer timing diagram (MTFE=0, CPHA=0, FMSZ=8)

The master initiates the transfer by placing its first data bit on the SOUT pin and asserting the appropriate peripheral chip select signals to the slave device. The slave responds by placing its first data bit on its SOUT pin. After the t_{ASC} delay elapses, the master outputs the first edge of SCK. The master and slave devices use this edge to sample the first input data bit on their serial data input signals. At the second edge of the SCK, the master and slave devices place their second data bit on their serial data output signals. For the rest of the frame the master and the slave sample their SIN pins on the odd-numbered clock edges and changes the data on their SOUT pins on the even-numbered clock edges. After the last clock edge occurs, a delay of t_{ASC} is inserted before the master negates the PCS signals. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

48.5.6.2 Classic SPI Transfer Format (CPHA = 1)

This transfer format shown in the following figure is used to communicate with peripheral SPI slave devices that require the first SCK edge before the first data bit becomes available on the slave SOUT pin. In this format, the master and slave devices change the data on their SOUT pins on the odd-numbered SCK edges and sample the data on their SIN pins on the even-numbered SCK edges.

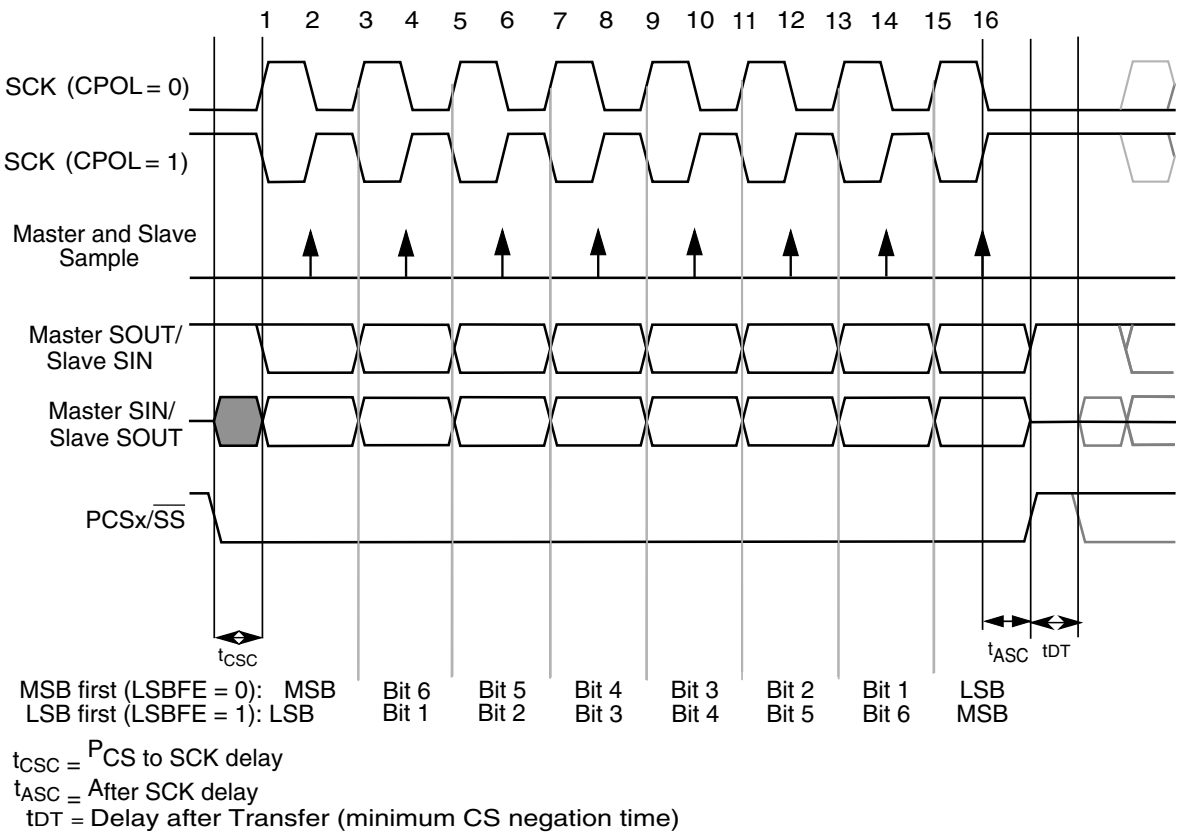


Figure 48-64. Module transfer timing diagram (MTFE=0, CPHA=1, FMSZ=8)

The master initiates the transfer by asserting the PCS signal to the slave. After the t_{CSC} delay has elapsed, the master generates the first SCK edge and at the same time places valid data on the master SOUT pin. The slave responds to the first SCK edge by placing its first data bit on its slave SOUT pin.

At the second edge of the SCK the master and slave sample their SIN pins. For the rest of the frame the master and the slave change the data on their SOUT pins on the odd-numbered clock edges and sample their SIN pins on the even-numbered clock edges. After the last clock edge occurs, a delay of t_{ASC} is inserted before the master negates the PCS signal. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

48.5.6.3 Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 0)

In this Modified Transfer Format both the master and the slave sample later in the SCK period than in Classic SPI mode to allow the logic to tolerate more delays in device pads and board traces. These delays become a more significant fraction of the SCK period as the SCK period decreases with increasing baud rates.

The master and the slave place data on the SOUT pins at the assertion of the PCS signal. After the PCS to SCK delay has elapsed the first SCK edge is generated. The slave samples the master SOUT signal on every odd numbered SCK edge. The DSPI in the slave mode when the MTFE bit is set also places new data on the slave SOUT on every odd numbered clock edge. Regular external slave, configured with CPHA=0 format drives its SOUT output at every even numbered SCK clock edge.

The DSPI master places its second data bit on the SOUT line one protocol clock after odd numbered SCK edge if the protocol clock frequency to SCK frequency ratio is higher than three. If this ratio is below four the master changes SOUT at odd numbered SCK edge. The point where the master samples the SIN is selected by the DSPI_MCR[SMPL_PT] field. The master sample point can be delayed by one or two protocol clock cycles. The SMPL_PT field should be set to 0 if the protocol to SCK frequency ratio is less than 4. However if this ratio is less than 4, the actual sample point is delayed by one protocol clock cycle automatically by the design.

The following timing diagrams illustrate the DSPI operation with MTFE=1. Timing delays shown are:

- T_{csc} - PCS to SCK assertion delay
- T_{acs} - After SCK PCS negation delay
- T_{su_ms} - master SIN setup time
- T_{hd_ms} - master SIN hold time
- T_{vd_sl} - slave data output valid time, time between slave data output SCK driving edge and data becomes valid.
- T_{su_sl} - data setup time on slave data input
- T_{hd_sl} - data hold time on slave data input
- T_{sys} - protocol clock period.

The following figure shows the modified transfer format for CPHA = 0 and Fsys/Fsck = 4. Only the condition where CPOL = 0 is illustrated. Solid triangles show the data sampling clock edges. The two possible slave behavior are shown.

- Signal, marked "SOUT of Ext Slave", presents regular SPI slave serial output.
- Signal, marked "SOUT of DSPI Slave", presents DSPI in the slave mode with MTFE bit set.

Other MTFE = 1 diagrams show DSPI SIN input as being driven by a regular external SPI slave, configured according DSPI master CPHA programming.

Note

In the following diagrams, f_{sys} represents the protocol clock frequency from which the Baud frequency f_{sck} is derived.

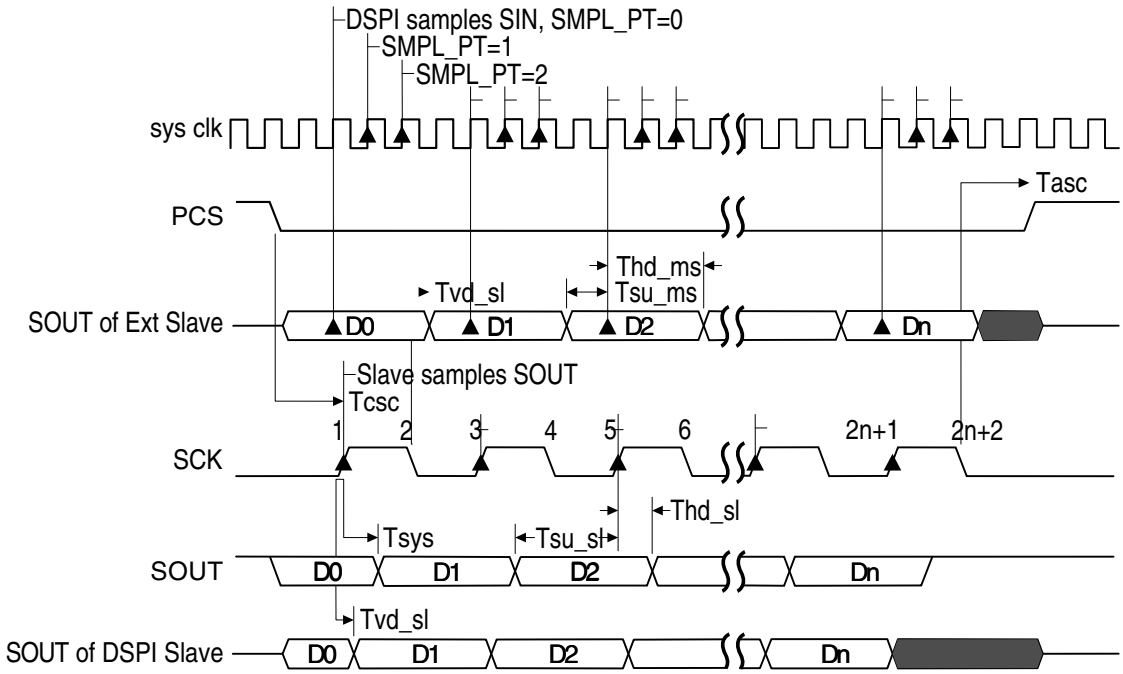


Figure 48-65. DSPI Modified Transfer Format (MTFE=1, CPHA=0, $f_{sck} = f_{sys}/4$)

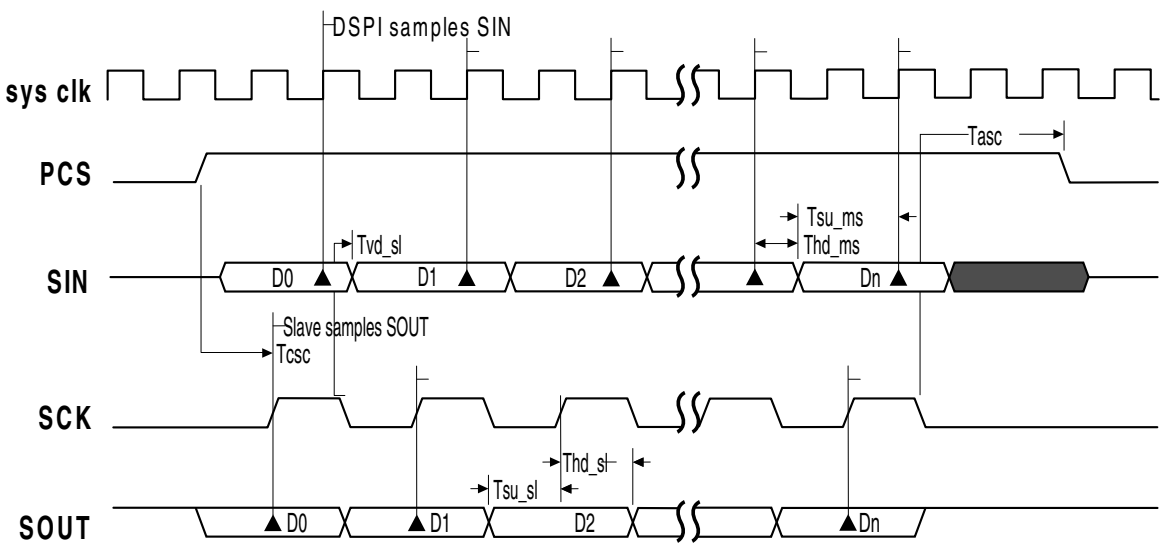


Figure 48-66. DSPI Modified Transfer Format (MTFE=1, CPHA=0, $f_{sck} = f_{sys}/2$)

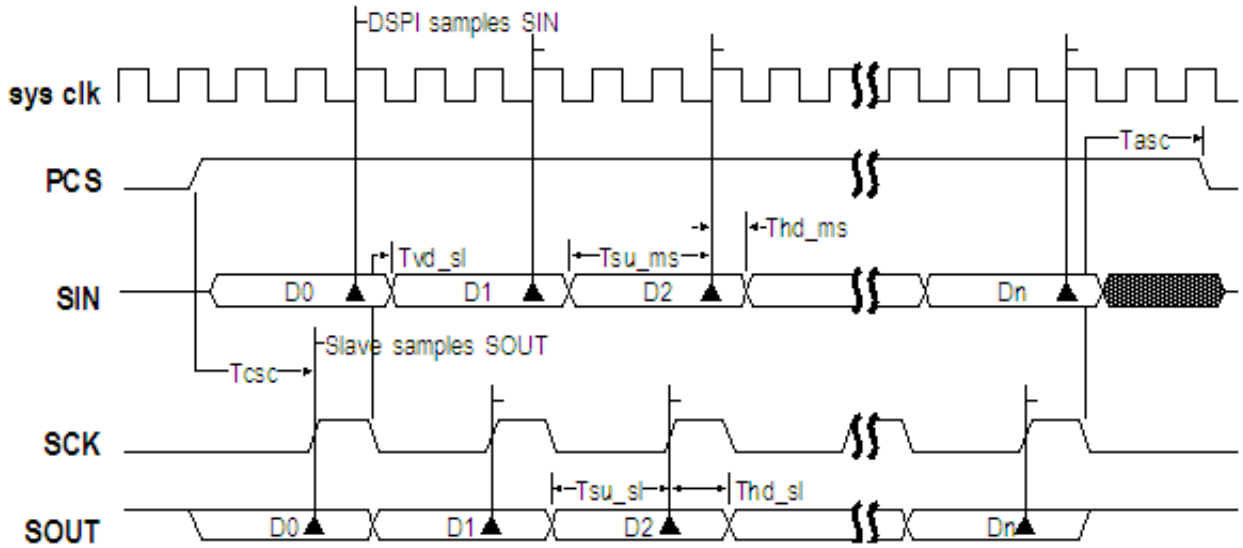


Figure 48-67. DSPI Modified Transfer Format (MTFE=1, CPHA=0, $f_{sck} = f_{sys}/3$)

48.5.6.4 Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 1)

The following figures show the Modified Transfer Format for CPHA = 1. Only the condition, where CPOL = 0 is shown. At the start of a transfer the DSPI asserts the PCS signal to the slave device. After the PCS to SCK delay has elapsed the master and the slave put data on their SOUT pins at the first edge of SCK. The slave samples the master SOUT signal on the even numbered edges of SCK. The master samples the slave SOUT signal on the odd numbered SCK edges starting with the third SCK edge. The slave samples the last bit on the last edge of the SCK. The master samples the last slave SOUT bit one half SCK cycle after the last edge of SCK. No clock edge will be visible on the master SCK pin during the sampling of the last bit. **The SCK to PCS delay and the After SCK delay must be greater or equal to half of the SCK period.**

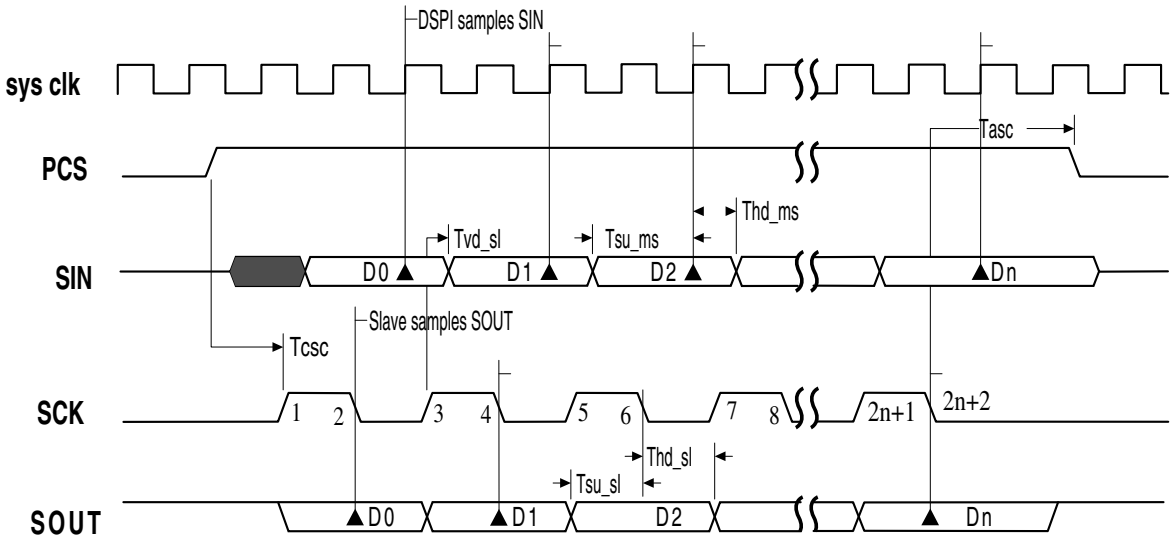


Figure 48-68. DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{sck} = f_{sys}/2$)

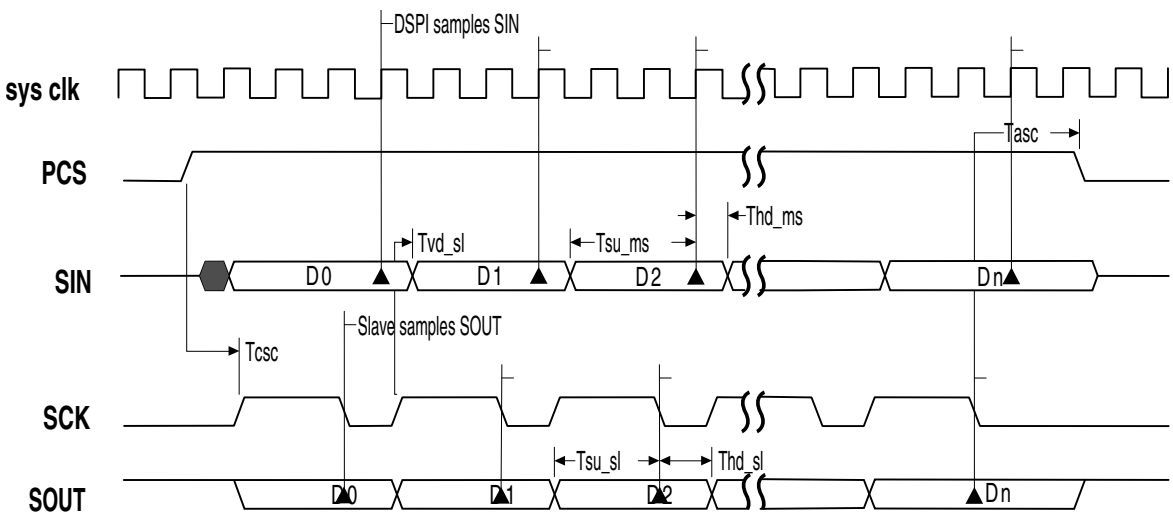


Figure 48-69. DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{sck} = f_{sys}/3$)

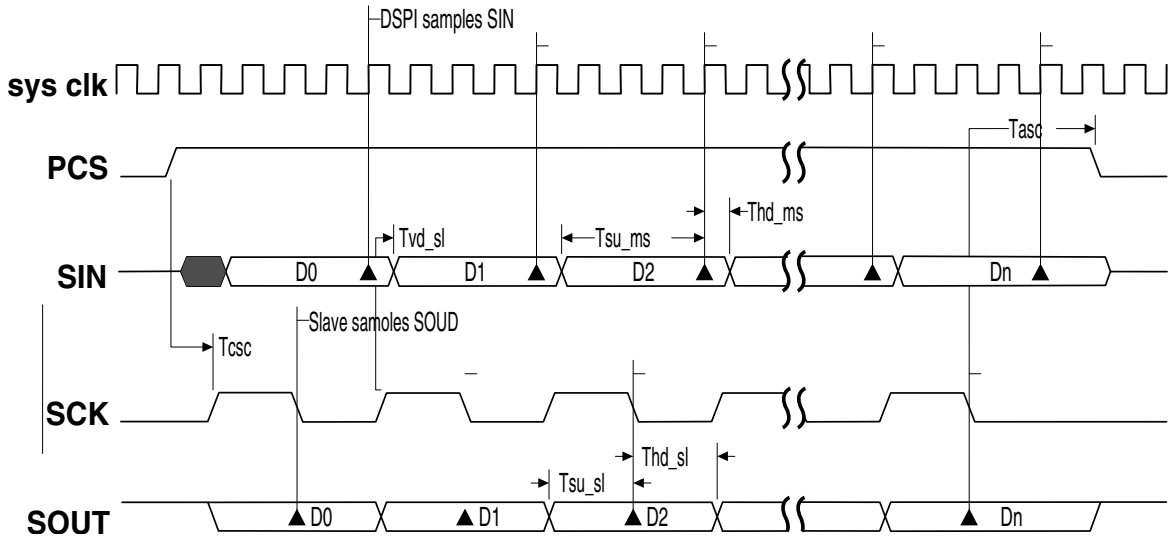


Figure 48-70. DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{sck} = f_{sys}/4$)

48.5.6.5 Continuous Selection Format

Some peripherals must be deselected between every transfer. Other peripherals must remain selected between several sequential serial transfers. The Continuous Selection Format provides the flexibility to handle both the following cases. The Continuous Selection Format is enabled for the SPI configuration by setting the CONT bit in the SPI command. Continuous Selection is enabled for the DSI Configuration by setting the DCONT bit in the DSICR0. The behavior of the PCS signals in the two configurations is identical so only SPI configuration will be described.

When the CONT bit = 0, the module drives the asserted Chip Select signals to their idle states in between frames. The idle states of the Chip Select signals are selected by the PCSISn bits in the MCR. The following timing diagram is for two four-bit transfers with CPHA = 1 and CONT = 0.

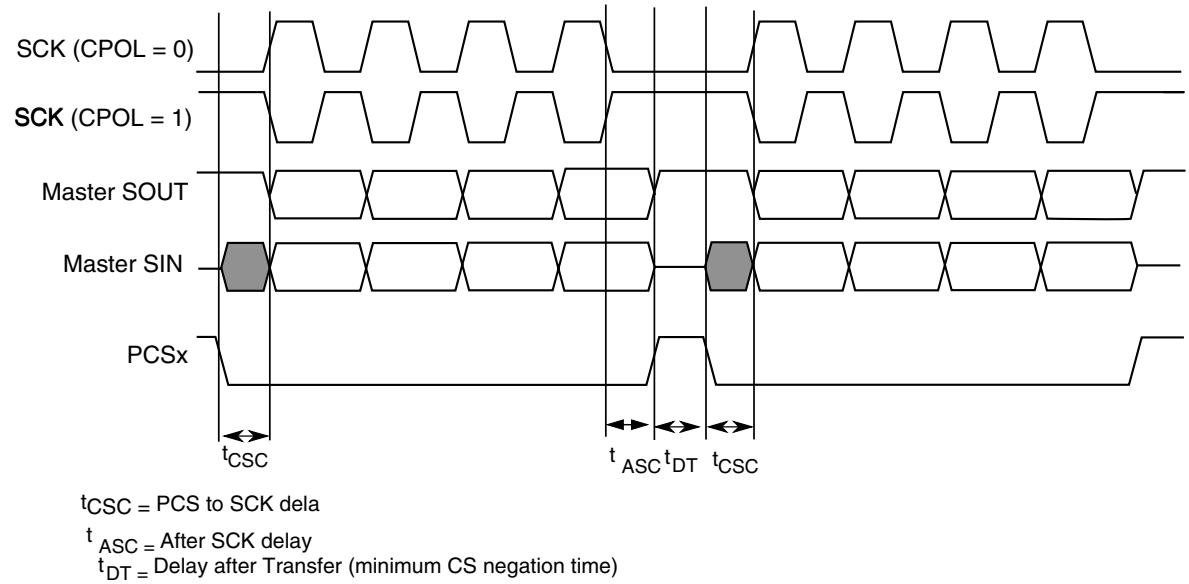


Figure 48-71. Example of non-continuous format (CPHA=1, CONT=0)

When the CONT bit = 1, the PCS signal remains asserted for the duration of the two transfers. The Delay between Transfers (t_{DT}) is not inserted between the transfers. The following figure shows the timing diagram for two four-bit transfers with CPHA = 1 and CONT = 1.

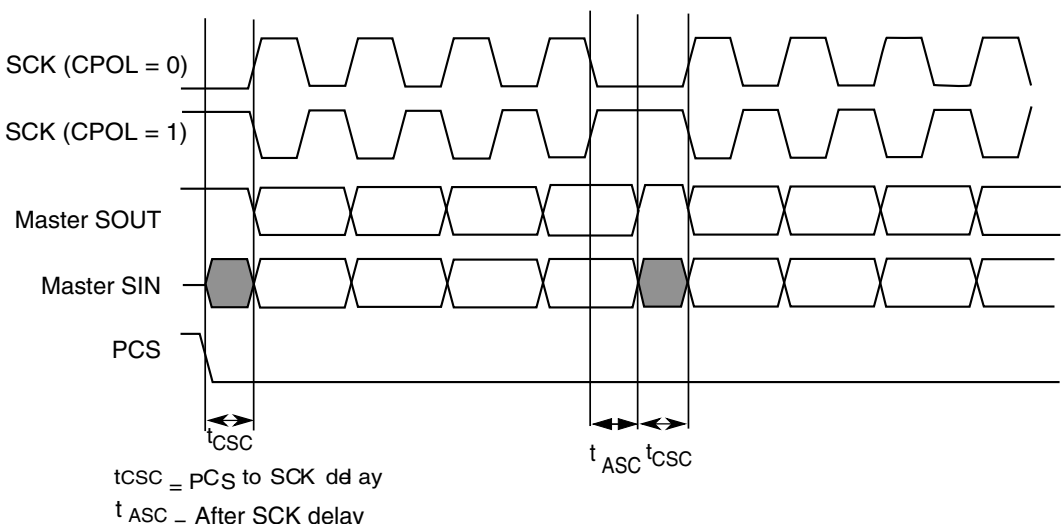


Figure 48-72. Example of continuous transfer (CPHA=1, CONT=1)

When using the module with continuous selection follow these rules:

- All transmit commands must have the same PCSn bits programming.
- The CTARs, selected by transmit commands, must be programmed with the same transfer attributes. Only FMSZ field can be programmed differently in these CTARs.

- When transmitting multiple frames in this mode, the user software must ensure that the last frame has the PUSHR[CONT] bit deasserted in Master mode and the user software must provide sufficient frames in the TX_FIFO to be sent out in Slave mode and the master deasserts the PCSn at end of transmission of the last frame.
- The PUSHR[CONT] / DSICR0[DCONT] bits must be deasserted before asserting MCR[HALT] bit in master mode. This will make sure that the PCSn signals are deasserted. Asserting MCR[HALT] bit during continuous transfer will cause the PCSn signals to remain asserted and hence Slave Device cannot transition from Running to Stopped state.

NOTE

User must fill the TX FIFO with the number of entries that will be concatenated together under one PCS assertion for both master and slave before the TX FIFO becomes empty.

When operating in Slave mode, ensure that when the last entry in the TX FIFO is completely transmitted, that is, the corresponding TCF flag is asserted and TXFIFO is empty, the slave is deselected for any further serial communication; otherwise, an underflow error occurs.

48.5.6.6 Fast Continuous Selection Format

The Fast Continuous Selection Format functions similar to Continuous Selection Format except that the inter command delays, t_{ASC} and t_{CSC}, can be masked out and are not inserted by the hardware.

NOTE

The Fast Continuous Selection Format is available in the SPI configuration only and when Continuous Serial Communication Clock mode is disabled. Masking of delays is not allowed in DSI and CSI configurations or if the transfer is non-continuous.

The Fast Continuous Selection Format is enabled by writing ‘1’ into FCPCS bit of the MCR register. When this bit is asserted, MASC and MCSC bits of the PUSHR register perform the function of mask bits for the transmit frame. These bits individually mask the t_{ASC} and t_{CSC} delays as programmed by the user software. A normal Continuous Selection Format has these two delays for each frame that is transmitted with the CONT

bit asserted. In order to avoid these delays and to speed up the transfer process, the software can simply mask these delays while programming the command in the PUSHR register.

While masking the delays, the software must follow the following masking rules, else correct operation is not guaranteed.

- MASC bit masks the “After SCK” delay for the current frame.
- MCSC bit masks the “PCS to SCK” delay for the next frame.
- “After SCK” (t_{ASC}) delay must not be masked when the current frame is the last frame in the continuous selection format.
- The “PCS to SCK” delay for the first frame in the continuous selection format cannot be masked.
- Masking of only t_{ASC} is not allowed. If t_{ASC} is masked then t_{CSC} must be masked too.
- Masking of both t_{ASC} and t_{CSC} delays is allowed. In this case, the delay between two frames is equal to half the baud rate set by the user software.
- Masking of only t_{CSC} is allowed. In this case, the delay between two frames is equal to the t_{ASC} time and thus the user software must ensure that the t_{ASC} time is greater than the baud rate.
- The user software must not mask these delays if the continuous selection format is not used and MCR[FCPCS] is asserted.
- Rules applicable to the Continuous Selection Format are applicable here too.

The following figure shows the timing for a Fast Continuous Selection Format transfer. Here seven frames are transferred with both t_{ASC} and t_{CSC} delays masked except for the last frame that terminated the transfer. The last frame has t_{ASC} delay at its end.

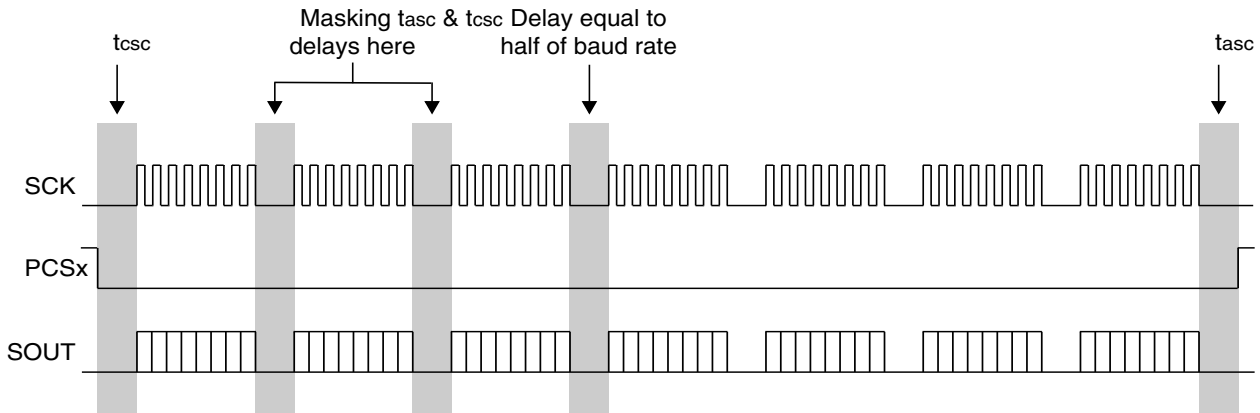


Figure 48-73. Example of Fast Continuous Selection Format

In case any chip select is to be changed, then the fast continuous selection format should be terminated and then the chips selects should change and appropriate delays must be introduced.

48.5.7 Continuous Serial Communications Clock

The module provides the option of generating a Continuous SCK signal for slave peripherals that require a continuous clock.

Continuous SCK is enabled by setting the CONT_SCKE bit in the MCR. Enabling this bit generates the Continuous SCK regardless of the MCR[HALT] bit status. Continuous SCK is valid in all configurations.

Continuous SCK is only supported for CPHA=1. Clearing CPHA is ignored if the CONT_SCKE bit is set. Continuous SCK is supported for Modified Transfer Format.

Clock and transfer attributes for the Continuous SCK mode are set according to the following rules:

- When the module is in SPI configuration, CTAR0 is used initially. At the start of each SPI frame transfer, the CTAR specified by the CTAS for the frame is used.
- When the module is in DSI configuration, the CTAR specified by the DSICTAS field is used at all times.
- When the module is in CSI configuration, the CTAR selected by the DSICTAS field is used initially.

- At the start of a SPI frame transfer, the CTAR specified by the CTAS value for the frame is used.
- At the start of a DSI frame transfer, the CTAR specified by the DSICTAS field is used.

- In all configurations, the currently selected CTAR remains in use until the start of a frame with a different CTAR specified, or the Continuous SCK mode is terminated.

It is recommended to keep the baud rate the same while using the Continuous SCK. Switching clock polarity between frames while using Continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the module is put into the External Stop mode or Module Disable mode.

Enabling Continuous SCK disables the PCS to SCK delay and the Delay after Transfer (t_{DT}) is fixed to one SCK cycle. When TSB configuration is enabled the t_{DT} is programmable from 1 to 64 SCK cycles. The following figure is the timing diagram for Continuous SCK format with Continuous Selection disabled.

NOTE

In Continuous SCK mode, for the SPI transfer CTAR0 should always be used, and the TX FIFO must be cleared using the MCR[CLR_TXF] field before initiating transfer.

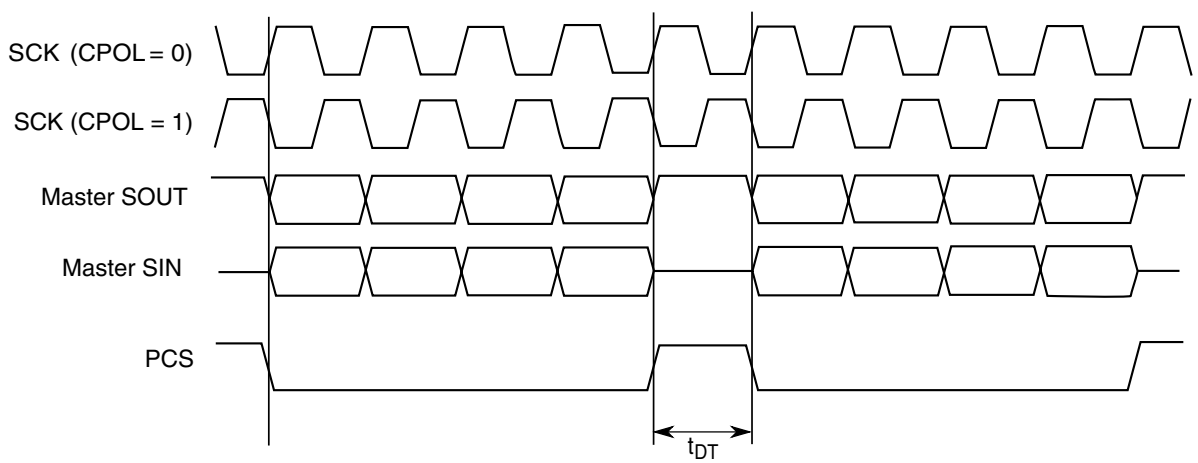


Figure 48-74. Continuous SCK Timing Diagram (CONT=0)

If the CONT bit in the TX FIFO entry is set or the DCONT in the DSICR0 is set, PCS remains asserted between the transfers. Under certain conditions, SCK can continue with PCS asserted, but with no data being shifted out of SOUT, that is, SOUT pulled high. This can cause the slave to receive incorrect data. Those conditions include:

- Continuous SCK with CONT bit set, but no data in the TX FIFO.

- Continuous SCK with CONT bit set and entering Stopped state (refer to Start and Stop of module transfers).
- Continuous SCK with CONT bit set and entering Stop mode or Module Disable mode.

The following figure shows timing diagram for Continuous SCK format with Continuous Selection enabled.

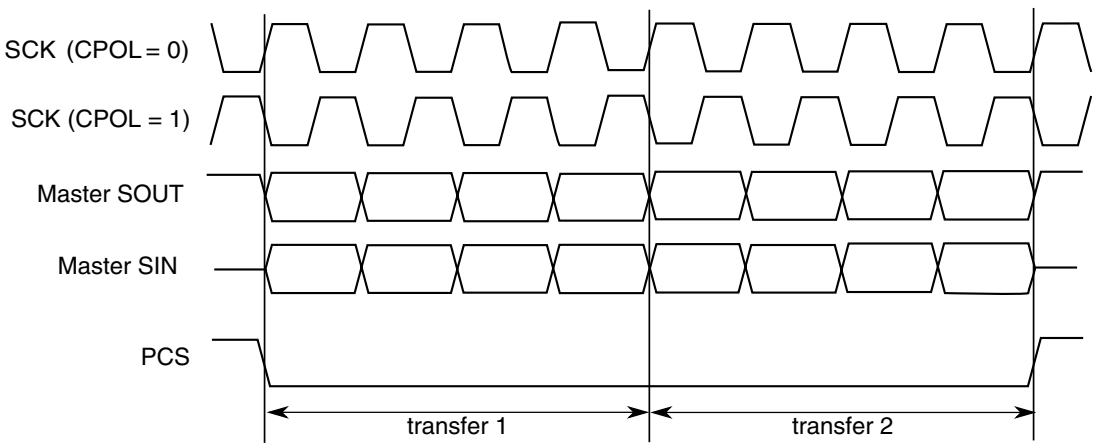


Figure 48-75. Continuous SCK timing diagram (CONT=1)

48.5.8 Slave Mode Operation Constraints

Slave mode logic shift register is buffered. This allows data streaming operation, when the DSPI is permanently selected and data is shifted in with a constant rate.

The transmit data is transferred at second SCK clock edge of the each frame to the shift register if the $\overline{SS}$ signal is asserted and any time when transmit data is ready and $\overline{SS}$ signal is negated.

Received data is transferred to the receive buffer at last SCK edge of each frame, defined by frame size programmed to the CTAR0/1 register. Then the data from the buffer is transferred to the RXFIFO or DDR register.

If the $\overline{SS}$ negates before that last SCK edge, the data from shift register is lost.

48.5.9 Timed Serial Bus (TSB)

The DSPI can be programmed in Timed Serial Bus configuration by setting the TSBC bit in the DSPI_DSICR0 register. See DSPI DSI Configuration Register 0 (DSPI_DSICR0) for details.

TSB configuration provides the Micro Second Channel (MSC) downstream channel support.

The MSC upstream channel is not supported by the DSPI, but can be supported by any available Serial Communication Controller (SCI or UART) in the chip.

The DSI Continuous Frame Selection Format is not supported by the TSB Mode (DSICR0[DCONT] bit cannot not be asserted).

To work in TSB mode the DSPI must be in master mode and in DSI (DCONF = 0b01) or CSI (DCONF = 0b10) configuration. Both Continuous and Non Continuous Serial Communication Clock (controlled by the DSPI_MCR[CONT_SCKE] bit) are supported in the TSB mode.

The following figure shows the signals used in the TSB interface.

In the TSB configuration the DSPI is able to send from 4 to 66 bits MSC data frames (4 to 64 serialized data bits and up to 2 Data Selection zero bits). The serialized data bits source can be either:

- the DSPI DSI Alternate Serialization Data Register (DSPI_ASDR0/1), written by the host software,
- Parallel Input pin states latched into the DSPI DSI Serialization Data Register (DSPI_SDR0/1).

DSPI_DSICR0 TXSS bit or DSPI_SSR0/1 register bits define the source of the data.

The Least Significant Bits of the DSPI_ASDR0 or DSPI_SDR0 registers are selected to be serialized if the data frame is set to less than 32 bits.

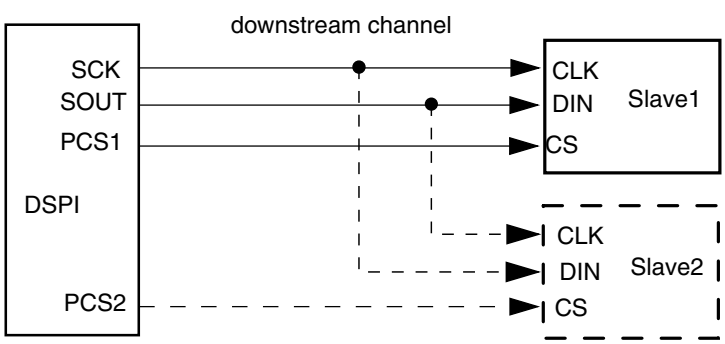


Figure 48-76. DSPI usage in the TSB Configuration

The PCS signals are driven together with SOUT. The t_{CSC} and t_{ASC} delays are not available. Delay after Transfer (DT) is set in SCK clock periods as a binary number formed by concatenation of the DSPI_CTARn PDT and DT fields plus one, allowing to set DT from 1 to 64 serial clock periods. DT field provides least significant bits and PDT field provides most significant bits of the Delay after Transfer.

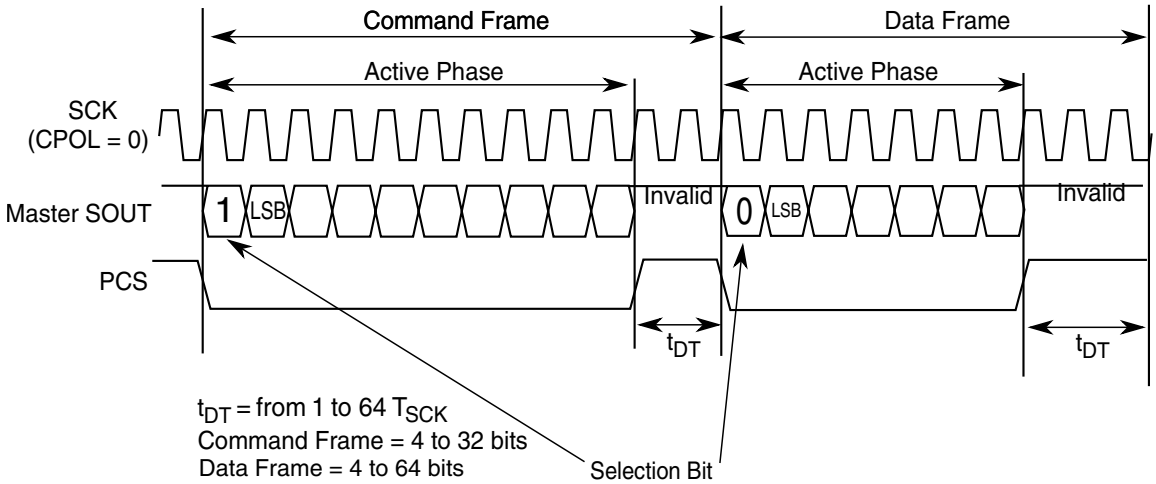


Figure 48-77. TSB Downstream Frames

Above figure shows the two types of MSC downstream frames - command frame, and data frame.

The first transmitted bit, called the selection bit, determines the frame type:

- The selection bit "0" indicates a data frame
- The selection bit "1" indicates a command frame

Data frame may contain up to 2 selection bits to support two external slave devices, (so called dual receiver configuration) or no selection bits at all.

The command frame can be written by software, through SPI TX FIFO, using one or two FIFO entries with help of the CONT bit or MCR[XSPI]. The data frame consists of up to 64 bits from the SDR0/1 or ASDR0/1 registers and up to two selection bits (0). Number of data bits in the data frame is defined by the DSCICR1[TSBCNT] field.

The selection bit of the MSC command frames (1) can be implemented by software.

The selection bits in the data frames are enabled by DSPI_DSICR1 DSE0 and DSE1 bits. Each DSEn bit set increases the data frame size by one bit.

To comply with MSC specification, set DSPI_CTARn[LSBFE] to transmit least significant bit first.

Regardless the LSBFE bit setting, the Data Frame Selection Bits, if enabled, are always transmitted first, before corresponded data sub frames.

48.5.9.1 MSC Dual Receiver Support with PCS Switch Over

When in TSB mode it is possible to switch the set of PCS signals that are driven during the first part of the frame to a different set of PCS signals during the second part of the frame. The bit, at which this switch over occurs, is defined by FMSZ field of the DSPI_CTARn register, selected by DSICTAS field of the DSICR0 register. If DSICR1[DSI64E] is enabled, then the bit at which the PCS switch occurs, is defined by the concatenation of {DSICR0[FMSZ4], CTAR[FMSZ]} fields.

Number of the bits, not including the Data Selection Bit, in the first part of the frame is equal to value of the FMSZ field plus one. (This value changes to {FMSZ4,FMSZ} plus one, when DSICR1[DSI64E] is enabled). During this part of the frame the PCS signal levels are controlled by DSPI_DSICR0 DPCSn bits, after that by DSPI_DSICR1 DPCS1_n bits.

The PCS switch over occurs at driving edge of the SCK clock output.

The second Data Selection Bit is inserted after the PCS switch over if enabled.

Data Frame with PCS switch over is shown in the following figure.

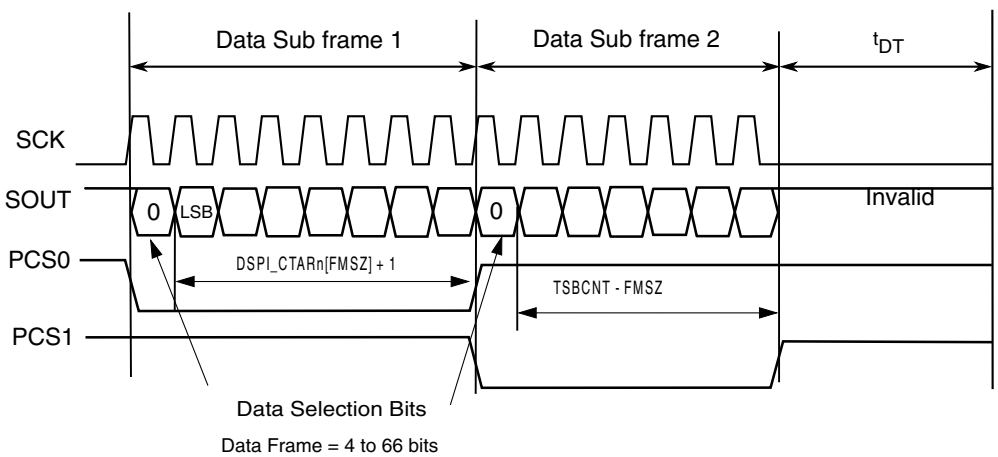


Figure 48-78. TSB Data Frame Format for MSC Dual Receiver Operation

48.5.10 Interleaved TSB (ITSB) Mode

This mode is similar to the TSB mode in some aspects except for the differences mentioned in the table below. The major difference is that the ITSB mode is highly generic in its approach to be compliant with the Downstream operation of the Microsecond Channel Protocol. It includes a triggered time-slot based system with user configurable options for the frames to be transmitted in each time slot. (The time period between any two triggers is known as a time-slot).

The DSPI can be programmed to the ITSB Mode by asserting DSICR0[TSBC], DSICR0[ITSB]. To work in ITSB Mode, the DSPI must be in Master Mode and MCR[DCONF] must be set to CSI mode. (DCONF = 0b01). Both continuous and non-continuous clock serial communication clock (MCR[CONT_SCKE] are supported in this mode. MSC dual receiver support with PCS switchover is also supported in ITSB mode. See MSC Dual Receiver Support with PCS Switch Over for more details.

There are three types of frames which can be transmitted in a time slot i.e. Data Frame, Command Frame and Passive (Idle) frame. Similar to the TSB Mode, SPI frames are used for MSC Command frames, while DSI frames are used for MSC Data frames. A Passive Frame is an Idle frame in which no chip selects are asserted and no data is transmitted. The frame transmission is broadly governed by the following rules:

- Transmission of frames is done on trigger pulses that can be generated internal to the DSPI. The internal trigger is periodic with a period of 4. Thus there are 4 time-slots which appear periodically in the ITSB mode. The frame rules to be followed during each of the 4 time-slots can be programmed via TS_CONF[TSn] fields. The length of each time slot can be programmed via TSL[TSx_LEN] fields. The minimum length of a time slot is governed by the following formulae :
  - Minimum Length for Time slot X = Frame size of the frame to be transmitted in Time slot X + no. of selection bits, if any + 2 (Inherent time(Delay) changed by baud rate(Fp/n)).
  - The above constraint is only true for Data and Command frames. Passive frames do not have a minimum trigger length constraint. The user software shall configure the trigger lengths and trigger source.
- If the previously transmitted frame was a Command (SPI) frame or the TX_FIFO (or CMD FIFO) is empty, the non-passive frame transmitted on a subsequent trigger will not be a command frame.
- If the previously transmitted frame was not a Command (SPI) frame and the TX_FIFO (and CMD FIFO) have sufficient data to allow a SPI frame transmission, the frame transmitted on subsequent trigger can be a Command (SPI) frame.
- A Dual Receiver Frame (DRF) is a single Data (DSI) Frame which is split and transmitted to two different receivers (via occurrence of PCS switchover at the split) - the former part being called DRF1 and the latter part as DRF2 (Dual Receiver Frames 1 & 2). Using TS_CONF[TSn] fields, a time-slot can be programmed to transmit a sole DRF1 or a DRF2 frame. A sole DRF1 frame followed by any other frame is a valid scenario. A DRF2 frame programmed for a time slot is only valid if the immediate previous time-slot was DRF1 - an invalid DRF2 frame, if programmed results in the transmission of a Passive frame (PF).

This mode does not change the way DSI and SPI frame transmissions are done except that ITSB mode interleaves the source (SPI or DSI) with a configurable set of rules based on triggered time-slot system. All features of SPI and DSI mode can be used except for the Continuous Frame Selection format (PUSHR[CONT] and DSICR0[DCONT] bits should not be asserted).

MCR[MTFE] should be set to 0 during ITSB Mode of operation, since MicroSecond Channel upstream operation is not supported.

The following table lists the differences between the TSB and ITSB modes:

Table 48-96. Differences between TSB and ITSB modes

TSB Mode	ITSB Mode
Priority available to frames from SPI. Back to back SPI frames are sent out.	No priority. Messages are interleaved such that there are no back-to-back SPI frames
Selection Bit inserted for DSI frames only. Selection bit for SPI frames is inserted by software.	Selection bit for both DSI and SPI frames inserted automatically. Encoding of selection bit same as TSB mode (0 = DSI; 1 = SPI).
No triggered transmission.	Triggered time-slot based transmission (that is frames are transmitted continuously on every trigger).
Gap between frames is configurable. Time between frames is t_{DT} which is configurable from 1 to 64 baud rate clock cycles.	Gap between frames is configurable and can be introduced in any of the two ways - By programming a Passive Frame or by increasing the length of a time-slot. Time t_{DT} is not used in this mode as it can be made part of the trigger period (as passive time).

In both TSB & ITSB mode, separate frame completion interrupts will be available to indicate frame transmission completion from SPI & DSI frames.

The ITSB mode is suitable to implement the Downstream Channel for the MSC Bus because of the above features.

48.5.10.1 Configuring DSPI for ITSB mode

The following steps give the recommended way to initialize the DSPI for the ITSB mode of operation:

- The DSPI should be put in HALT mode (by asserting the MCR[HALT] bit) before programming the registers.
- Set the DSICR0[TSBC] and DSICR0[ITSB] bits. The DSICR0[TRRE] and DSICR0[CID] bits should be cleared. DSICR0[DCONT] and PUSHR[CONT] should not be used in ITSB mode. The bit MCR[MTFE] should be cleared as

Upstream operation is not supported in ITSB Mode. If DSICR0[ITSB] bit is set without setting DSICR0[TSBC], then the DSPI operates in the Normal Mode depending on MCR[DCONF] bit.

- Configure the Time Slot Length's via TSL register and the frame rules for each time slot via TS_CONF register.
- Based on trigger source requirements, set the values in the DSICR0[TRG] bit.
- Enable frame completion interrupts for either any frame completion (TCF) or separate frame complete (DSITCF and SPITCF) as required by application.
- When remainder configuration of DSPI is complete, clear the MCR[HALT] bit to allow DSPI to start operations.

Once configured and DSPI is brought out of HALT, the ITSB mode shall start sequencing the frames from either DSI or SPI as per the rules mentioned in Interleaved TSB (ITSB) Mode.

48.5.10.2 Variable Length Time Slots

In the Interleaved TSB Mode, the TSL[TSx_LEN] registers are used to configure the time slot length's as required by the application. There are up to 4 time slots named TS0, TS1, TS2 and TS3 respectively, which repeat periodically. To achieve a periodicity less than 4, any time slot length can be programmed as 0 (which eliminates this time slot). The following figures illustrate this concept.

In the figure below, the fields TSL[TSx_LEN] are programmed to be non-zero for x = 0,1,2,3. Hence, the time slots repeat after every 4 trigger pulses as shown.

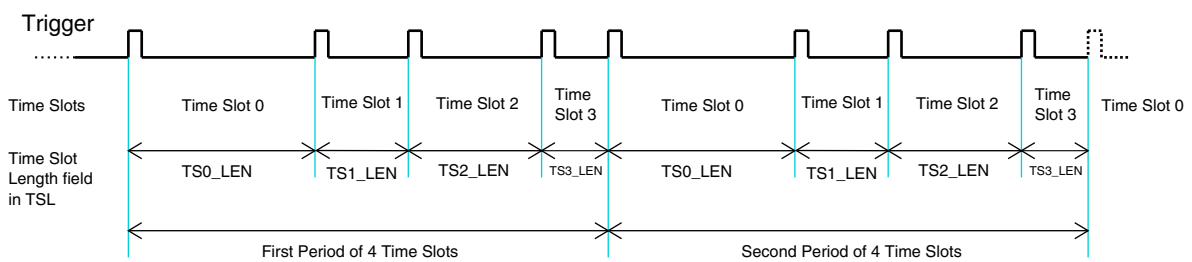


Figure 48-79. Periodic Time Slots with period = 4

In the figure below, the fields TSL[TSx_LEN] are programmed to be non-zero for x = 0,1,2, while TSL[TS3_LEN] = 0. Hence, the time slots repeat after every 3 trigger pulses as shown.

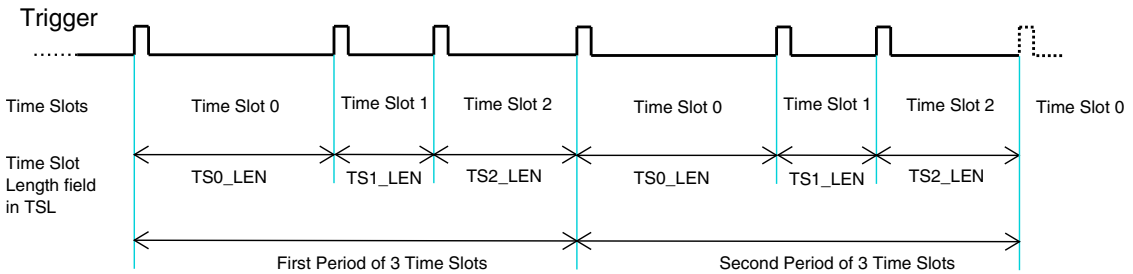


Figure 48-80. Periodic Time Slots with period = 3

The following figure shows time slots with periodicity of 1. Here the fields TSL[TSx_LEN] are programmed to zero for x = 1,2 and 3 while TSL[TS0_LEN] is non-zero. Hence, the time slots repeat after every trigger as shown.

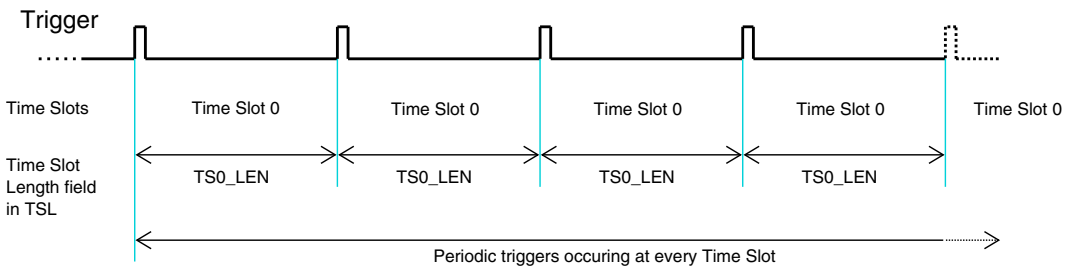


Figure 48-81. Periodic Time Slots with period = 1

Similarly, any TSL[TSx_LEN] can be made 0 to vary periodicity of time slots, but it is an illegal scenario to have all TSL[TSx_LEN] fields as 0.

NOTE

If external triggering mechanism is enabled (DSICR[TRG] is set), then it must be ensured that the minimum distance between two trigger pulses is at least 5P, where P is the protocol clock period.

When internal triggering mechanism is enabled (DSICR[TRG] is zero), the minimum value which can be programmed in the field TSL[TSx_LEN] is given by L = F + 1 + m; where F is the frame size of the frame programmed to be transmitted for the time slot TSx. The variable 'm' equals 1 when maximum baud rate is programmed (i.e. SCK = protocol clock frequency/ 2) and 0 otherwise.

48.5.10.3 Using ITSB mode for MSC Downstream Channel

MSC Downstream Channel uses three types of frames: command frames, data frames and passive frames. The Downstream operation of these frames require:

- Data frames can be sent back to back.
- Back to back command frames are not a legal scenario.
- Passive frames are used when no activity is desired for any time slot. In a passive frame, chip selects are de-asserted, and no SCK or data is transmitted. For this reason Command Frames cannot be separated by Passive frames as it would still be an illegal scenario.
- In dual MSC ASIC, some bits of the data frame goes to first ASIC and remainder bits to second ASIC. The first part is referred to as DRF1 (Dual Receiver Frame 1) and the second part as DRF2 (Dual Receiver Frame 2).
- A DRF2 programmed for a time slot is only valid if the immediately preceding time slot was DRF1. If DRF2 is valid, it would be transmitted seamlessly after DRF1 without waiting for the trigger pulse for the time slot of DRF2. For that reason, the TSL[TSx_LEN] for DRF1 must be programmed to the minimum possible value. If DRF2 is not valid, a Passive Frame is transmitted instead.

The ITSB mode can help implement the MSC Downstream Channel by allowing the command frames to be sent via the SPI interface and data frames to be sent via the DSI interface. Once the DSPI has been configured to work in ITSB mode (See Configuring DSPI for ITSB mode), the user software can program the DSI and SPI sides to send data and command frames respectively. The programming of frames for transmission via DSI or SPI is same as explained in Combined Serial Interface (CSI) Configuration.

The following figure shows a sample MSC Downstream transmission for the ITSB mode with the following configuration:

- All TSL[LEN] are programmed to be non-zero. Note that the length between triggers are not drawn to scale in the figure. The lengths of all 4 time slots shown are unequal.
- Data and Command selection bits are enabled, i.e. DSICR1[DSE0] and DSICR1[CSE] are set.
- TS_CONF[TS0] = CMD_SRDF
- TS_CONF[TS1] = CMD_SRDF
- TS_CONF[TS2] = CMD_PF
- TS_CONF[TS3] = CMD_SRDF
- PCS0 is used for Data Frames while PCS1 is used for Command Frames. Both chip select's are active low.

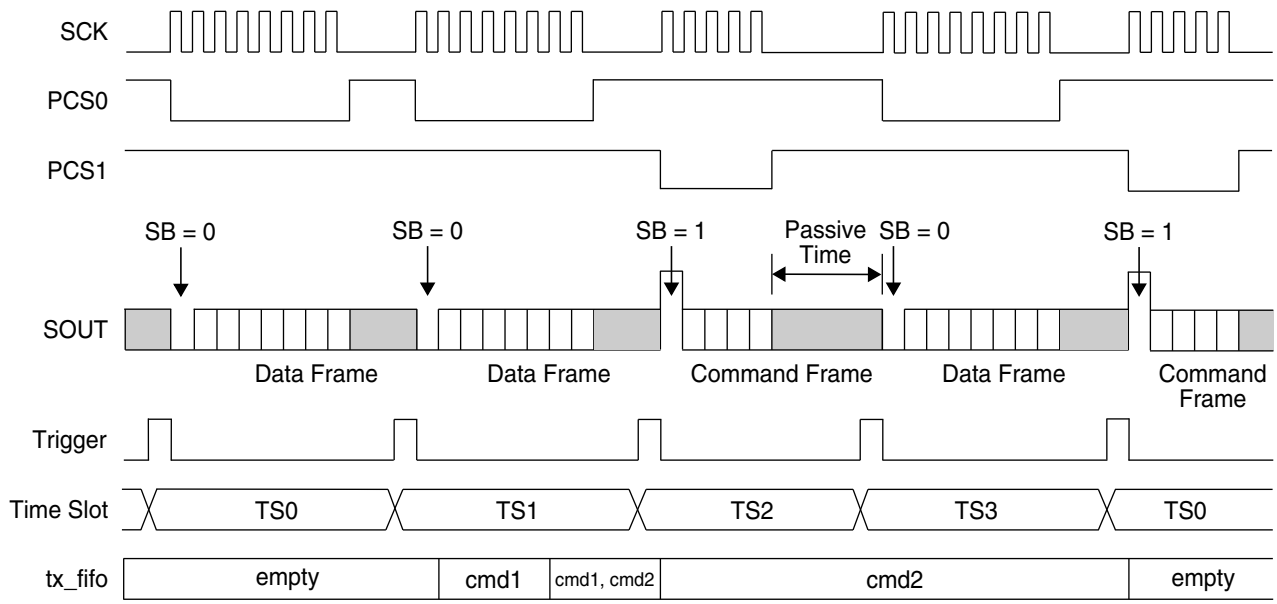


Figure 48-82. Sample MSC Downstream transmission using ITSB mode

Brief description of operation includes:

- At the first trigger, TS0 is initiated. The frame rule for TS0 is configured as CMD_SRDF (Command if valid, else Single Receiver Data Frame). Since there is no data present in the TXFIFO, a valid command frame is not available to be transmitted. Hence a single receiver data frame is transmitted instead.
- The second trigger initiates TS1. Again a single receiver data frame is transmitted since the frame rule for TS1 is programmed to CMD_SRDF and a command frame is not valid as there is no data present in the TXFIFO when TS1 is initiated.
- During TS1, two frames are pushed into the TX_FIFO.
- At third trigger, TS2 is initiated , the frame rule for which is programmed to CMD_PF (Command if valid. else Passive Frame). A Command frame is transmitted as the TX_FIFO is not empty.
    - Since the frame size of the command frame is less than the trigger length for TS2, no transmission happens till next trigger. This is marked as the “Passive Time”. Such passive time exists in all frames after frame transmission completes and until next trigger pulse is detected.
- For TS3, the frame rule programmed is CMD_SRDF. Since the previous transmission was a command frame, a single receiver data frame is transmitted during TS3.

- The time slot TS0 repeats, and this time a command frame is transmitted during TS0 since the Command Frame is eligible to be transmitted.
- At sixth trigger (not shown), TS1 occurs and the process continues.

The figure below shows another sample MSC Downstream transmission for the ITSB mode with the following configuration:

- All TSL[LEN] are programmed to be non-zero. Note that the length between triggers are not drawn to scale in the figure. The lengths of all 4 time slots shown are unequal.
- Data and Command selection bits are enabled, i.e. DSICR1[DSE0] and DSICR1[CSE] are set.
- TS_CONF[TS0] = CMD_DRF1
- TS_CONF[TS1] = DRF2
- TS_CONF[TS2] = CMD_DRF1
- TS_CONF[TS3] = CMD_DRF2
- PCS0 is used for DRF1 while PCS1 is used for DRF2 or Command Frames. Both chip select's are active low.

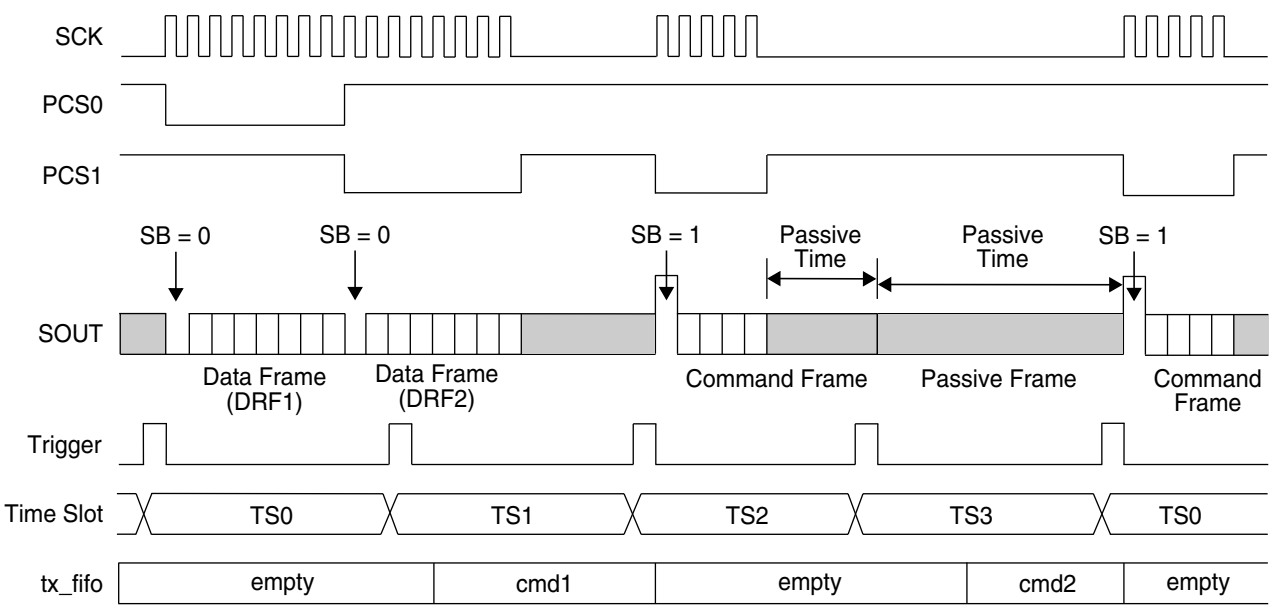


Figure 48-83. Sample MSC Downstream transmission using ITSB mode

Brief description of operation includes:

- At the first trigger, TS0 is initiated. The frame rule for TS0 is configured as CMD_DRF1 (Command if valid, else Dual Receiver Frame 1). Since there is no data present in the TXFIFO, a valid command frame is not available to be transmitted. Hence DRF1 is selected for transmission. Now it is checked if the next trigger is DRF2 which is true and hence DRF1 and DRF2 are transmitted back-to-back seamlessly with the PCS switch-over occurring in between. Note that this happens independent of the next trigger. If the next trigger were configured to CMD_DRF2 instead, then at the current trigger, the validity of CMD is checked to confirm the presence of DRF2 for next trigger. If presence of DRF2 is confirmed for next trigger, it is transmitted along with DRF1 seamlessly as shown here, otherwise only DRF1 would be transmitted in the current trigger and the CMD frame would be transmitted in the next trigger.
- During TS1, one frame is pushed into the TX_FIFO.
- Since DRF2 is transmitted immediately after DRF1, the occurrence second trigger does not cause any operation. Hence it must be ensured by the user that Trigger Length of TS0 must be programmed to minimum possible value - during which DRF1 can be successfully transmitted. This ensures that trigger for TS1 occurs sometime after the switch-over and not after DRF2 has been transmitted.
- The third trigger initiates TS2, the frame rule for which is configured as CMD_DRF1. A Command frame is transmitted as the data required to transmit a command frame is available (TX_FIFO and CMD_FIFO is not empty) and the previous transmitted frame was not a command frame.
- The fourth trigger initiates TS3, the frame rule for which is configured as CMD_DRF1. Since a CMD frame isn't available and DRF2 is not valid (since DRF1 wasn't transmitted in the previous trigger), a passive frame is transmitted instead.
- The time slot TS0 repeats, and this time a command frame is transmitted during TS0 since the Command Frame is eligible to be transmitted.
- At sixth trigger (not shown), TS1 occurs and the process continues.

NOTE

For a command/data frame to be transmitted on a trigger, they should be available some time before the occurrence of the trigger. A DSI frame must be updated/present "X" time units before the occurrence of the trigger for the time slot in which this data is to be transmitted. This is true for all DSI frames and also implies that when DRF1 and DRF2 are positioned to be

transmitted seamlessly, both DRF1 and DRF2 frames must be available 'X' time units before occurrence of trigger pulse for the time slot of DRF1 itself.

Similarly, a command frame must be available "Y" time units before the occurrence of the trigger for the time slot in which this data is to be transmitted.

Here X = 2R + 1P and Y = 1R + 3P; where R = register clock period, P = protocol clock period.

48.5.11 Parity Generation and Check

The module can generate and check parity in the serial frame. The parity bit replaces the last transmitted bit in the frame. The parity is calculated for all transmitted data bits in frame, not including the last data bit that would be transmitted. The parity generation/ control is done on frame basis. The registers field setting frame size defines the total number of bits in the frame, including the parity bit. Thus, to transmit/receive the same number of data bits with parity check, increase the frame size by one versus the same data size frame without the parity check.

Parity can be selected as odd or even. Parity Errors in the received frame set Parity Error flags in the Status register. The Parity Error Interrupt Requests are generated if enabled. The module can be programmed to stop SPI or/and DSI frame transmission in case of a frame reception with parity error.

48.5.11.1 Parity for SPI Frames

When the module is in the master mode the parity generation is controlled by PE and PP bits of the CMD FIFO entries (PUSHR). Setting the PE bit enables parity generation for transmitted SPI frames and parity check for received frames. PP bit defines polarity of the parity bit.

When continuous PCS selection is used to transmit SPI data, two parity generation scenarios are available:

- Generate/check parity for the whole frame
- Generate/check parity for each sub-frame separately.

To generate/check parity for the whole frame set PE bit only in the last command/TX FIFO entry, forming this frame (with the PUSHR register).

To generate/check parity for each sub-frame set PE bit in each command/TX FIFO entry, forming this frame.

If the parity error occurs for received SPI frame, the SR[SPEF] bit is set. If MCR[PES] bit is set, the module stops SPI frames transmission. To resume SPI operation clear the SR[SPEF] or the MCR[PES] bits.

In slave mode the parity is controlled by the PE and PP bits of the CTAR0 register similar to the master mode parity generation without continuous PCS selection.

48.5.11.2 Parity for DSI Frames

When DSPI is in Master Mode, parity generation is controlled by PE and PP bits of the DSPI_DSICR0 register similar to the SPI frames. The parity is calculated and checked for each DSI frame. DSPI_DSICR0[DCONT] bit has no effect on parity generation. In slave mode the parity is controlled by the PE and PP bits of the CTAR1 register.

If the parity error occurs for received DSI frame, the DSPI_SR[DPEF] bit is set. If DSPI_DSICR0[PES] bit is set, the DSPI stops DSI frames transmission. To resume DSI operation clear the DSPI_SR[DPEF] or the DSPI_DSICR0[PES] bits.

48.5.12 Interrupts/DMA requests

The module has several conditions that can generate only interrupt requests and two conditions that can generate interrupt or DMA requests. The following table lists these conditions.

Table 48-97. Interrupt and DMA request conditions

Condition	Flag	Interrupt	DMA
End of Queue (EOQ)	EOQF	Yes	-
TX FIFO Fill	TFFF	Yes	Yes
CMD FIFO Fill	CMDFFF	Yes	Yes
TX FIFO Invalid Write	TFIWF	Yes	-
Transfer Complete	TCF	Yes	-
CMD Transfer Complete	CMDTCF	Yes	-
SPI Transfer Complete	SPITCF	Yes	-
DSI Transfer Complete	DSITCF	Yes	-
TX FIFO Underflow	TFUF	Yes	-
RX FIFO Drain	RFDF	Yes	Yes
RX FIFO Overflow	RFOF	Yes	-
SPI Parity Error	SPEF	Yes	-

Table continues on the next page...

Table 48-97. Interrupt and DMA request conditions (continued)

Condition	Flag	Interrupt	DMA
DSI Parity Error	DPEF	Yes	-
DSI Deserialized Data Match	DDIF	Yes	Yes

Each condition has a flag bit in the module Status Register (SR) and a Request Enable bit in the DMA/Interrupt Request Select and Enable Register (RSER). Certain flags (as shown in above table) generate interrupt requests or DMA requests depending on configuration of RSER register.

The module also provides a global interrupt request line, which is asserted when any of individual interrupt requests lines is asserted.

48.5.12.1 End of Queue Interrupt Request

The End of Queue Request indicates that the end of a transmit queue is reached. The End of Queue Request is generated when the EOQ bit in the executing SPI command is set and the EOQF_RE bit in the RSER is set.

When MCR[XSPI] is enabled and the EOQ bit in the executing SPI command is set, the End of Queue Request will only be generated once the last bit of the last Data frame in the Command Cycle has been transmitted.

NOTE

When MCR[XSPI] is disabled, this interrupt request is generated when the last bit of the SPI frame with EOQ bit set is transmitted.

48.5.12.2 Transmit FIFO Fill Interrupt or DMA Request

The Transmit FIFO Fill Request indicates that the TX FIFO is not full. The Transmit FIFO Fill Request is generated when the number of entries in the TX FIFO is less than the maximum number of possible entries, and the TFFF_RE bit in the RSER is set. The TFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

NOTE

TFFF flag clears automatically when DMA is used to fill TX FIFO.

To clear TFFF when not using DMA, follow these steps for every PUSH performed using CPU to fill TX FIFO:

1. Wait until TFFF = 1.
2. Write data to PUSHR using CPU.
3. Clear TFFF by writing a 1 to its location. If TX FIFO is not full, this flag will not clear.

48.5.12.3 Command FIFO Fill Interrupt or DMA Request

The Command FIFO Fill Request indicates that the CMD FIFO is not full. The Command FIFO Fill Request is generated when the number of entries in the CMD FIFO is less than the maximum number of possible entries, and the CMDFFF_RE bit in the RSER is set. The CMDFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

This Request is useful when MCR[XSPI] is enabled, since TX FIFO and CMD FIFO can be filled independently. If MCR[XSPI] is disabled, then 'TX FIFO Fill Interrupt or DMA Request' will suffice to fill both FIFO's since both FIFO's must be filled simultaneously.

NOTE

CMDFFF flag clears automatically when DMA is used to fill CMDFIFO.

To clear CMDFFF when not using DMA, follow these steps for every PUSH performed using CPU to fill CMDFIFO:

1. Wait until CMDFFF = 1
2. Write data to PUSHR using CPU.
3. Clear CMDFFF by writing a 1 to its location. If CMD FIFO is not full, this flag will not clear.

48.5.12.4 Transmit FIFO Invalid Write Interrupt Request

The Transmit FIFO Invalid Write Request is valid only when MCR[XSPI] is enabled. This Request indicates that Data exists in the TX FIFO while the CMD FIFO is empty. Since no Command Fields are associated with the Data present in TX FIFO, this data is considered invalid until a Command Entry becomes available. The Transmit FIFO Invalid Write Request is generated for the above condition when TFIWF_RE bit is set in the RSER.

48.5.12.5 Transfer Complete Interrupt Request

The Transfer Complete Request indicates the end of the transfer of a serial frame. The Transfer Complete Request is generated at the end of each frame transfer when the TCF_RE bit is set in the RSER.

48.5.12.6 Command Transfer Complete Interrupt Request

The Command Transfer Complete Request indicates the end of transfer of the last SPI frame in a Command Cycle. The Transfer Complete Request is generated for the above condition when the CMDTCF_RE bit is set in the RSER.

48.5.12.7 SPI Transfer Complete Interrupt Request

The SPI Transfer Complete Request indicates the end of transfer of an SPI frame in any Transmission Mode which uses DSPI in CSI Mode. The SPI Transfer Complete Request is generated when SR[SPITCF] flag asserts and SPITCF_RE bit is set in the RSER.

48.5.12.8 DSI Transfer Complete Interrupt Request

The DSI Transfer Complete Request indicates the end of transfer of a DSI frame in any Transmission Mode which uses DSPI in CSI Mode. The DSI Transfer Complete Request is generated when SR[DSITCF] flag asserts and DSITCF_RE bit is set in the RSER.

48.5.12.9 Transmit FIFO Underflow Interrupt Request

The Transmit FIFO Underflow Request indicates that an underflow condition in the TX FIFO has occurred. The transmit underflow condition is detected only for the module operating in Slave mode and SPI configuration. The TFUF bit is set when the TX FIFO of a DSPI is empty, and a transfer is initiated from an external SPI master. If the TFUF bit is set while the TFUF_RE bit in the RSER is set, an interrupt request is generated.

48.5.12.10 Receive FIFO Drain Interrupt or DMA Request

The Receive FIFO Drain Request indicates that the RX FIFO is not empty. The Receive FIFO Drain Request is generated when the number of entries in the RX FIFO is not zero, and the RFDF_RE bit in the RSER is set. The RFDF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

48.5.12.11 Receive FIFO Overflow Interrupt Request

The Receive FIFO Overflow Request indicates that an overflow condition in the RX FIFO has occurred. A Receive FIFO Overflow request is generated when RX FIFO and shift register are full and a transfer is initiated. The RFOF_RE bit in the RSER must be set for the interrupt request to be generated.

Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

48.5.12.12 SPI Frame Parity Error Interrupt Request

The SPI Frame Parity Error Flag indicates that a SPI frame with parity error had been received. The SPEF_RE bit in the RSER must be set for the interrupt request to be generated.

48.5.12.13 DSI Frame Parity Error Interrupt Request

The DSI Frame Parity Error Flag indicates that a DSI frame with parity error has been received. The DPEF_RE bit in the DSPI_RSER must be set for the interrupt request to be generated.

48.5.12.14 Deserialized Data Match Interrupt or DMA Request

The Deserialized Data Match Flag (DDIF) indicates that a DSI frame matches DSPI_DPIR0/1 data, masked with DSPI_DIMR0/1, had been received. This Request is generated if DDIF_RE bit in the DSPI_RSER is set. The DDIF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

48.5.13 Power saving features

The module supports following power-saving strategies:

- External Stop mode
- Module Disable mode – Clock gating of non-memory mapped logic

48.5.13.1 Stop mode (External Stop mode)

The DSPI supports the Stop mode protocol. When a request is made to enter External Stop mode, the DSPI block acknowledges the request. If a serial transfer is in progress, the DSPI waits until it reaches the frame boundary before it is ready to have its clocks shut off. While the clocks are shut off, the DSPI memory-mapped logic is not accessible. This also puts the DSPI in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state. The states of the interrupt and DMA request signals cannot be changed while in External Stop mode.

48.5.13.2 Module Disable mode

Module Disable mode is a block-specific mode that the module can enter to save power. Host CPU can initiate the Module Disable mode by setting the MDIS bit in the MCR. The Module Disable mode can also be initiated by hardware.

When the MDIS bit is set, the module negates Clock Enable signal at the next frame boundary. Once the Clock Enable signal is negated, it is said to have entered Module Disable Mode. This also puts the module in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state.If implemented, the Clock Enable signal can stop the clock to the non-memory mapped logic. When Clock Enable is negated, the module is in a dormant state, but the memory mapped registers are still accessible. Certain read or write operations have a different effect when the module is in the Module Disable mode. Reading the RX FIFO Pop Register does not change the state of the RX FIFO. Similarly, writing to the PUSHR Register does not change the state of the TX FIFO or CMD FIFO. Clearing either of the FIFOs has no effect in the Module Disable mode. Changes to the DIS_TXF and DIS_RXF fields of the MCR have no effect in the Module Disable mode. In the Module Disable mode, all status bits and register flags in the module return the correct values when read, but writing to them has no effect. Writing to the TCR during Module Disable mode has no effect. Interrupt and DMA request signals cannot be cleared while in the Module Disable mode.

48.6 Initialization/application information

This section describes how to initialize the module.

48.6.1 How to manage queues

The queues are not part of the module, but it includes features in support of queue management. Queues are primarily supported in SPI configuration.

1. When module executes last command word from a queue, the EOQ bit in the command word is set to indicate it that this is the last entry in the queue.
2. At the end of the transfer, corresponding to the command word with EOQ set is sampled, the EOQ flag (EOQF) in the SR is set.
3. The setting of the EOQF flag disables serial transmission and reception of data, putting the module in the Stopped state. The TXRXS bit is cleared to indicate the Stopped state.
4. The DMA can continue to fill TX FIFO until it is full or step 5 occurs.
5. Disable DMA transfers by disabling the DMA enable request for the DMA channel assigned to TX FIFO (and CMD FIFO) and RX FIFO. This is done by clearing the corresponding DMA enable request bits in the DMA Controller.
6. Ensure all received data in RX FIFO has been transferred to memory receive queue by reading the RXCNT in SR or by checking RFDF in the SR after each read operation of the POPR.
7. Modify DMA descriptor of TX and RX channels for new queues
8. Flush TX FIFO (and CMD FIFO) by writing a 1 to the CLR_TXF bit in the MCR. Flush RX FIFO by writing a '1' to the CLR_RXF bit in the MCR.
9. Clear transfer count either by setting CTCNT bit in the command word of the first entry in the new queue or via CPU writing directly to SPI_TCNT field in the TCR.
10. Enable DMA channel by enabling the DMA enable request for the DMA channel assigned to the module TX FIFO,(and CMD FIFO) and RX FIFO by setting the corresponding DMA set enable request bit.
11. Enable serial transmission and serial reception of data by clearing the EOQF bit.

48.6.2 Switching Master and Slave mode

When changing modes in the module, follow the steps below to guarantee proper operation.

1. Halt it by setting MCR[HALT].
2. Clear the transmit and receive FIFOs by writing a 1 to the CLR_TXF and CLR_RXF bits in MCR.
3. Set the appropriate mode in MCR[MSTR] and enable it by clearing MCR[HALT].

48.6.3 Initializing Module in Master/Slave Modes

Once the appropriate mode in MCR[MSTR] is configured, the module is enabled by clearing MCR[HALT]. It should be ensured that module Slave is enabled before enabling it's Master. This ensures the Slave is ready to be communicated with, before Master initializes communication.

48.6.4 Baud rate settings

The following table shows the baud rate that is generated based on the combination of the baud rate prescaler PBR and the baud rate scaler BR in the CTARs. The values calculated assume a 100 MHz protocol frequency and the double baud rate DBR bit is cleared.

Table 48-98. Baud rate values (bps)

		Baud rate divider prescaler values			
		2	3	5	7
Baud Rate Scaler Values	2	25.0M	16.7M	10.0M	7.14M
	4	12.5M	8.33M	5.00M	3.57M
	6	8.33M	5.56M	3.33M	2.38M
	8	6.25M	4.17M	2.50M	1.79M
	16	3.12M	2.08M	1.25M	893k
	32	1.56M	1.04M	625k	446k
	64	781k	521k	312k	223k
	128	391k	260k	156k	112k
	256	195k	130k	78.1k	55.8k
	512	97.7k	65.1k	39.1k	27.9k
	1024	48.8k	32.6k	19.5k	14.0k
	2048	24.4k	16.3k	9.77k	6.98k
	4096	12.2k	8.14k	4.88k	3.49k
	8192	6.10k	4.07k	2.44k	1.74k
	16384	3.05k	2.04k	1.22k	872
	32768	1.53k	1.02k	610	436

48.6.5 Delay settings

The following table shows the values for the Delay after Transfer (t_{DT}) and CS to SCK Delay (T_{CSC}) that can be generated based on the prescaler values and the scaler values set in the CTARs. The values calculated assume a 100 MHz protocol frequency.

NOTE

The clock frequency mentioned above is given as an example in this chapter. See the clocking chapter for the frequency used to drive this module in the device.

This table does not apply for TSB Continuous mode.

Table 48-99. Delay values

		Delay prescaler values			
		1	3	5	7
Delay scaler values	2	20.0 ns	60.0 ns	100.0 ns	140.0 ns
	4	40.0 ns	120.0 ns	200.0 ns	280.0 ns
	8	80.0 ns	240.0 ns	400.0 ns	560.0 ns
	16	160.0 ns	480.0 ns	800.0 ns	1.1 µs
	32	320.0 ns	960.0 ns	1.6 µs	2.2 µs
	64	640.0 ns	1.9 µs	3.2 µs	4.5 µs
	128	1.3 µs	3.8 µs	6.4 µs	9.0 µs
	256	2.6 µs	7.7 µs	12.8 µs	17.9 µs
	512	5.1 µs	15.4 µs	25.6 µs	35.8 µs
	1024	10.2 µs	30.7 µs	51.2 µs	71.7 µs
	2048	20.5 µs	61.4 µs	102.4 µs	143.4 µs
	4096	41.0 µs	122.9 µs	204.8 µs	286.7 µs
	8192	81.9 µs	245.8 µs	409.6 µs	573.4 µs
	16384	163.8 µs	491.5 µs	819.2 µs	1.1 ms
	32768	327.7 µs	983.0 µs	1.6 ms	2.3 ms
	65536	655.4 µs	2.0 ms	3.3 ms	4.6 ms

48.6.6 Calculation of FIFO pointer addresses

Complete visibility of the TX, CMD and RX FIFO contents is available through the FIFO registers, and valid entries can be identified through a memory-mapped pointer and counter for each FIFO. The pointer to the first-in entry in each FIFO is memory mapped. For the TX FIFO the first-in pointer is the Transmit Next Pointer (TXNXTPTR). For the CMD FIFO the first-in pointer is the Command Next Pointer (CMDNXTPTR). For the RX FIFO the first-in pointer is the Pop Next Pointer (POPNXTPTR). The following figure illustrates the concept of first-in and last-in FIFO entries along with the FIFO Counter. The TX FIFO is chosen for the illustration, but the concepts carry over to the CMD FIFO and RX FIFO. See Transmit First In First Out (TX FIFO) buffering mechanism, Command First In First Out (CMD FIFO) Buffering Mechanism and Receive First In First Out (RX FIFO) buffering mechanism for details on the FIFO operation.

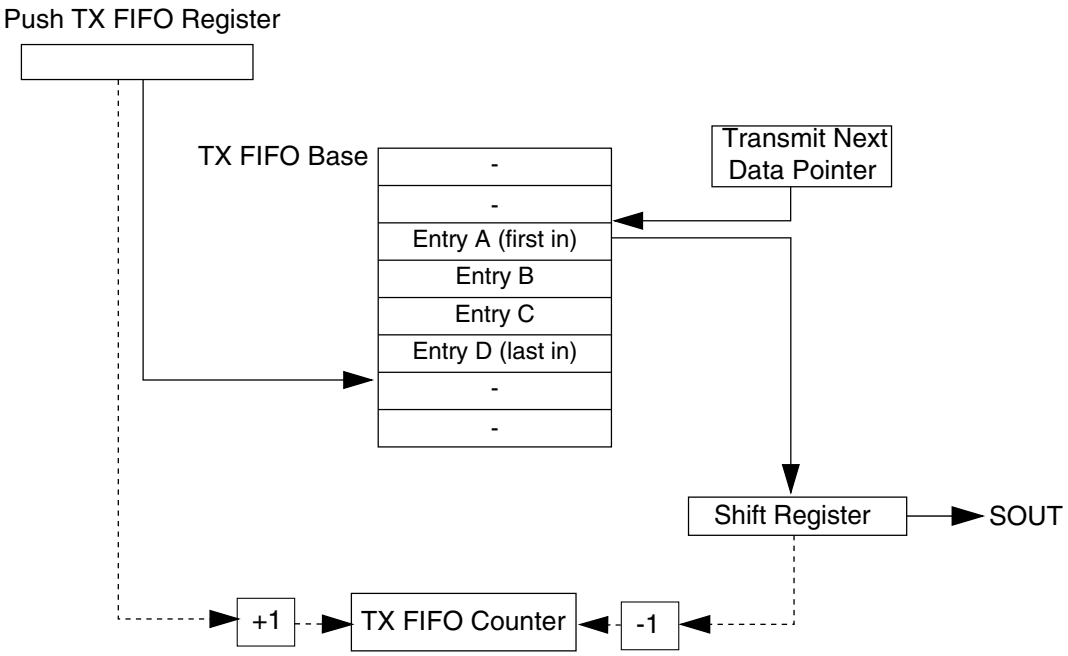


Figure 48-84. TX FIFO pointers and counter

48.6.6.1 Address Calculation for the First-in Entry and Last-in Entry in the TX FIFO

The memory address of the first-in entry in the TX FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{TXFIFOBase} + \left(4 \times \text{TXNXTPTR}\right)$$

The memory address of the last-in entry in the TX FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{TXFIFOBase} + 4 \times \left(\text{TXCTR} + \text{TXNXTPTR} - 1\right) \text{mod} \left(\text{TXFIFOdepth}\right)$$

TX FIFO Base - Base address of TX FIFO
TXCTR - TX FIFO Counter
TXNXTPTR - Transmit Next Pointer
TX FIFO Depth - Transmit FIFO depth, implementation specific

48.6.6.2 Address Calculation for the First-in Entry and Last-in Entry in the CMD FIFO

The memory address of the first-in entry in the CMD FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{TXFIFOBase} + (4 \times \text{TXNXTPTR})$$

The memory address of the last-in entry in the CMD FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{TXFIFOBase} + 4 \times (\text{TXCTR} + \text{TXNXTPTR} - 1) \bmod (\text{TXFIFOdepth})$$

CMD FIFO Base - Base address of CMD FIFO
CMDCTR - CMD FIFO Counter
CMDNXTPTR - Command Next Pointer
CMD FIFO Depth - Command FIFO depth, implementation specific

48.6.6.3 Address Calculation for the First-in Entry and Last-in Entry in the RX FIFO

The memory address of the first-in entry in the RX FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{RX FIFOBase} + (4 \times \text{POPNXTPTR})$$

The memory address of the last-in entry in the RX FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{RX FIFO Base} + 4 \times (\text{RXCTR} + \text{POPNXTPTR} - 1) \bmod (\text{RXFIFOdepth})$$

RX FIFO Base - Base address of RX FIFO
RXCTR - RX FIFO counter
POPNXTPTR - Pop Next Pointer
RX FIFO Depth - Receive FIFO depth, implementation specific

Chapter 49
Zipwire

49.1 Introduction

Zipwire is a group of modules that allow one MCU to have a fast, low pin count, serial communication link directly into the memory mapped peripherals and/or memories of another MCU or a smart ASIC. Zipwire is implemented in hardware so there is no CPU load for the intitiator or target nodes.

Zipwire supports 8-bit, 16-bit or 32-bit reads and writes to any 32-bit address at the target node.

Zipwire architecture is fully pipelined and support multiple outstanding commands to allow maximum use of the serial link bandwidth.

The serial link runs at 320 Mbaud using LVDS physical layer. One LVDS pair for Tx and another pair for Rx, with a separate 20 MHz reference clock. Five pins in total.

Zipwire also supports a streaming mode for the transfer of large blocks of data.

In streaming mode Zipwire has a transfer rate of approximately 32 MB/second. For single random access 32-bit read from any 32-bit address location, the latency is approximately 1 µs.

49.2 Block Diagram

Figure 49-1 shows two MCUs with Zipwire connected using a five wire interface. Both MCUs support Target and Initiator modes of Zipwire. The SIPI Bus Master Interface is used for all Target mode transactions. The SIPI Register Interface is used for all Initiator Transactions and initial setup. The LFAST Register Interface is used for Initial Setup.

The diagram shows the LFAST reference clock coming from MCU B clock system. The Ref_Clock can come from either MCU. It is user configurable, but it must be the same clock that drives both LFAST modules.

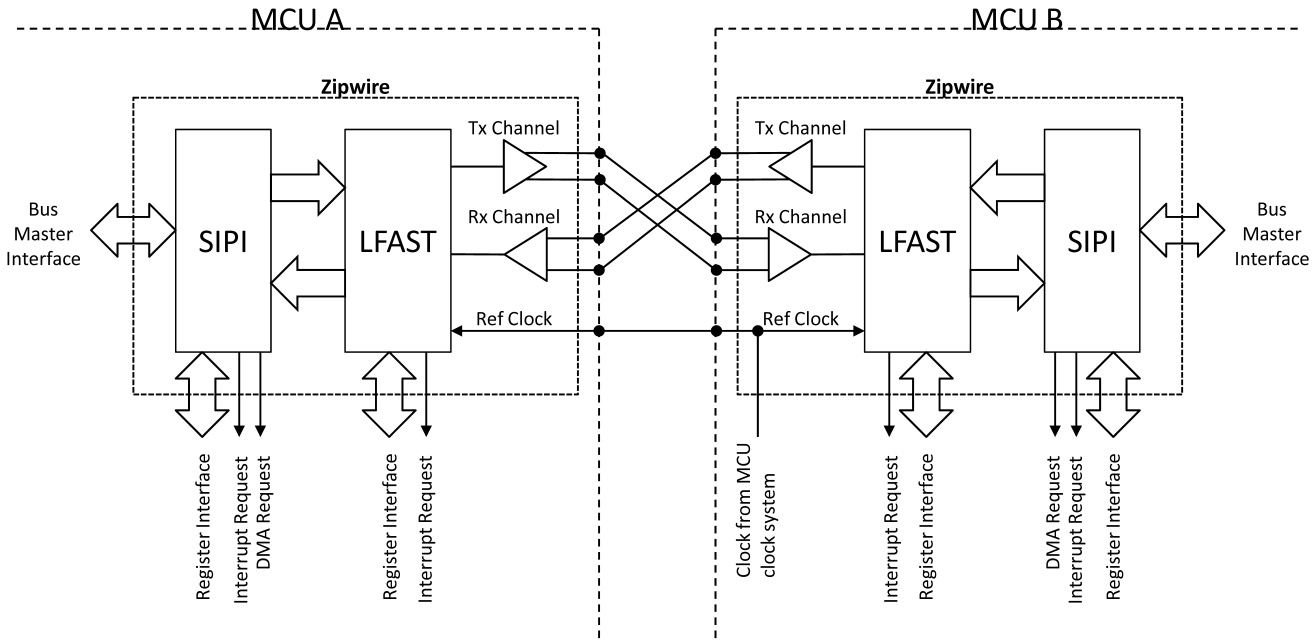


Figure 49-1. Zipwire block diagram

49.3 Architecture

Zipwire is composed of several modules and system resources.

The physical layer is LVDS with 1.2 V common mode voltage and 200mV swing. The LVDS pads are shared with other IO and GPIO.

The transport layer is a protocol called LFAST. LFAST is an asynchronous prototcol, using a non-return to zero encoding. The protocol starts with a fixed 16-bit sync frame to allow the receiver to detect the optimal point to sample the inconing data. Followed by an 8-bit header that defines the channel number and the size of the payload. Finally, the payload, which can be between 8 and 288 bits.

The application layer protocol is called Serial Inter Processor Interface (SIPI) and runs on top of LFAST and is fully encapsulated within the LFAST payload. SIPI uses four fixed sizes of LFAST payload:

- 32-bit
- 64-bit

- 96-bit
- 288-bit

The SIPI protocol implements a suite of commands initiated my one MCU, for reading and writing any 32-bit address location in another connected MCU. The SIPI protocol allows either or both MCUs to initiate commands. The protocol also supports interrupt requests from one MCU to the other.

The SIPI protocol is implemented fully in hardware in a module called SIPI. The module implements both the initiator part of the protocol and the recipient part of the protocol. The module implements four SIPI Initiator channels that can run independently of each other and can run simultaneously. The SIPI module is a bus master on the low speed crossbar. As the target MCU for a command from the inititating MCU, SIPI can perform read and write accesses to any address location within the target MCU. The software running on the local MCU, should configure the system MPU to allow/deny memory accesses to MCU memory and resources as required.

The initiator part of the module is connected to the DMA controller and is able to generate DMA requests as a command is completed. This allows a series of read or write commands to be queued in the Inititator MCU and executed by the DMA and SIPI module without CPU intervention at either Initiator or Target MCU.

49.4 LFAST Clocking

The LFAST implementation includes a dedicated PLL for multiplying a low speed clock source up to 320 MHz for the transmission and reception of data. The high speed asynchronous nature of LFAST, requires that both the local node and remote node, operate from the same, stable, low jitter clock sources. This common low speed clock is referred to a the Reference Clock.

The MCU includes clock multiplexing to allow the source of the LFAST reference clock to come from one of three sources via a programmable clock divider:

- Crystal oscillator
- Output of PLL0
- External pin

The most expedient source for the reference clock is the 20 MHz crystal oscillator on one MCU. This 20 MHz clock can be used as the LFAST Reference clock for the local LFAST. This same 20 MHz clock must be exported from the local MCU to the remote MCU, to be used as the LFAST reference at the remote MCU.

The 20 MHz clock may be used in the following ways:

- Shared between the MCUs using a dedicated reference clock pin on each MCU
- Divided by 2 and shared as a 10 MHz reference
- Shared as the fundamental clock for the MCU system clock and the LFAST reference clock

The LFAST PLL for generating the 320 MHz clock, is shared between LFAST for SIPI.

49.5 Zipwire interconnections

LFAST has the following connections:

- Tx and Rx connections to the pads via the SIUL and the MSCR registers.
- Peripheral Bridge interface (PBRIDGE) — Allows software to read and write configuration registers.
- Tx Data Port/Rx Data Port — Directly connected to the SIPI module. Allows received data to be efficiently transfered to SIPI and transmit data to be transfered from SIPI.
- Interrupt Request connections — Allows the module to flag to the CPU when it requires servicing.

SIPI has the following connections:

- Peripheral Bridge interface (PBRIDGE) — Allows software to read and write configuration registers and SIPI Interface Registers.
- DMA connections — Allows SIPI command sequences to be queues and initiated without CPU intervention.
- Crossbar master port — Allows SIPI to execute requested commands to read and write MCU address space.
- Tx Data Port/Rx Data Port — Directly connected to the LFAST module. Allows received data to be efficiently transfered from LFAST and transmit data to be transfered to LFAST.
- Interrupt Request connections — Allows the module to flag to the CPU when it requires servicing.

49.6 Zipwire performance

Two aspects of performance are considered:

- Bandwidth — The rate at which data can be read or written between two nodes. It assumes that read or write commands from the Inititator node, can be sent continuously at the highest speed the Target node can consume those commands.
- Latency — The time from the Initiator node sending a read or write command to the Initiator node receiving back the read data or write acknowledge.

49.6.1 Read performance

A Zipwire read operation consists of three stages:

- Inititator sends SIPI Read command to Target.
- Target parses the received command and runs a master bus cycle to read the data.
- Target sends the SIPI Read response back to the Initiator.

A SIPI Read command consists of:

- 16-bit header
- 32-bit read address
- 16-bit CRC
- 64 bits total

A SIPI Read response consists of:

- 16-bit header
- 32-bit read data
- 16-bit CRC
- 64 bits total

The SIPI message is encapsulated in an LFAST frame where the LFAST payload is the size of the SIPI message.

The LFAST frame consists of:

- 16-bit synchronisation header
- 8-bit header
- 64-bit payload
- 1-bit stop bit
- 89 bits total

LFAST Baud rate is 320 Mbaud which yields a bit time of 3.13 nS

Therefore, an 89-bit LFAST transmission of a SIPI 64-bit Read command or 64-bit Read response takes 278 nS

The SIPI module takes thirteen system clock cycles to parse a command and create a Read response, plus the time to run the master bus cycle. The time to run the master bus cycle will vary depending on the memory being read.

Assumptions:

- Average of 6 × slow crossbar cycles to read different memories.
- Slow crossbar clocked at 100 MHz
- SIPI clocked at 50 MHz

Therefore, time for SIPI to read an address location is:

- (13 x 50MHz clocks) + (6 x 100MHz clocks) = 320nS

49.6.1.1 Read bandwidth

The Zipwire target node has a three message deep receive FIFO and a three message deep transmit FIFO. So, the Target node can simulataneously be receiving a command, processing a command, and sending the Read response.

The time to transmit a command or response is less than the time to process the command. Therefore, the bandwidth is limited by the time to process the message.

Read Bandwidth = four bytes in 320 nS = 12.5 MB/second

49.6.1.2 Read latency

The latency is the time taken to send a Read command, process the command, and send a Read response.

Read Latency = 278 nS + 320 nS + 278 nS = 876 nS

49.6.2 Write performance

A Zipwire write operation consists of three stages:

- Inititator sends SIPI Write command to Target.
- Target parses the received command and runs a master bus cycle to write the data.
- Target sends the SIPI Write response back to the Initiator.

A SIPI Write command consists of:

- 16-bit header
- 32-bit write address
- 32-bit write data
- 16-bit CRC
- 96 bits total

A SIPI Write acknowledge consists of:

- 16-bit header
- 16-bit CRC
- 32 bits total

The SIPI message is encapsulated in an LFAST frame where the LFAST payload is the size of the SIPI message.

The LFAST frame consists of:

- 16-bit synchronisation header
- 8-bit header
- 96-bit payload (command) or 32 bit (acknowledge)

- 1-bit stop bit
- 121 bits total (command) or 57 bits (acknowledge)

LFAST Baud rate is 320 Mbaud which yields a bit time of 3.13 nS

Therefore, the time for an LFAST transmission is:

- 121-bit LFAST transmission of a SIPI 96-bit Write Command takes 378 nS
- 57-bit LFAST transmission of a SIPI 32-bit Write Acknowledge takes 178 nS

The SIPI module takes thirteen system clock cycles to parse a command and create a Write response, plus the time to run the master bus cycle. The time to run the master bus cycle will vary depending on the memory being written.

Assumptions:

- Average of 5 × slow crossbar cycles to write different memories.
- Slow crossbar clocked at 100 MHz
- SIPI clocked at 50 MHz

Therefore, the time for SIPI to write an address location is:

- (13 × 50 MHz clocks) + (5 × 100 MHz clocks) = 320 nS

49.6.2.1 Write bandwidth

The Zipwire target node has a three message deep receive FIFO and a three message deep transmit FIFO. So, the Target node can simulataneously be receiving a command, processing a command, and sending the Write Acknowledge response.

The time to transmit the command is longer than the time to process the command, or the time to transmit the Write Acknowledge. Therefore, the bandwidth is limited by the time to transmit the command.

Write Bandwidth = four bytes in 378 nS = 10.6 MB/second

49.6.2.2 Write latency

The latency is the time taken to send a Write command, process the command, and send a Write Acknowledge.

Write Latency = 378 nS + 310 nS + 178 nS = 866 nS

49.7 LFAST/SIPI configuration

This section summarizes the module configuration in the controller.

The SIPI and LFAST are connected to one another and appear as a single unit called Zipwire. The LFAST portion of the two modules allows for high speed inter-device communications. The SIPI allows memory to be shared between devices which have SIPI and LFAST communication modules.

The LFAST can operate in either slave or master mode configurations. The node running in master mode controls the serial link, but SIPI, in both master and slave modes, can act as both initiator and target for SIPI commands simultaneously. Please see the SIPI and LFAST chapters for detailed information on module configuration and functionality.

The SIPI CHIP ID = 0980_501Dh.

Chapter 50
Serial Interprocessor Interface (SIPI)

50.1 Introduction

The Serial Interprocessor Interface (SIPI) is an application layer protocol which runs on top of the LFAST (LVDS Fast Asynchronous Serial Transmission) module. It is used by the local device to access the shared memory of a remote device. SIPI defines point-to-point full duplex communication between two devices, where LFAST works as a physical medium of communication between both the devices.

50.1.1 Scalability

The SIPI protocol is designed to provide a sophisticated, high bandwidth, multimaster, multi-channel memory interface between 2 devices using few interconnecting signals. But the protocol is designed in such a way that a subset of the protocol can be implemented, where die area is more important than features.

Main scalable features:

- Number of concurrent channels:
  - Full Implementation = 4
  - Minimum Implementation = 1
- Full implementation has a node both as Initiator and Target of commands. Minimum implementation either Initiator or Target.
- Full Implementation includes a block transfer feature, but this feature is optional.

The rest of this section describes a full implementation of SIPI which includes:

- Advanced High-Performance Bus (AHB-Lite) master interface

- Direct Memory Access (DMA) interface
- LFAST Tx/Rx (transmit/receive) interfaces along with Peripheral Bus Interface (IPS)

50.2 Overview

An instance of SIPI can act as initiator, or target or both. SIPI can access the shared memory directly through its AHB master interface or through its DMA interface. DMA interface is used when the node acts as an initiator while the AHB Master interface will be used when the node acts as target. SIPI has four channels, with one channel (Channel 2) having data streaming capability. Payload width for channel 0, 1 and 3 is 32 bits, and for channel 2 data widths can be 32 bits, or 256 bits when streaming. Any of these channels can be used for DMA access or bus interface access depending on the setting of the SIPI_CCR*n*. CRC encoder calculates the CRC on the command frame. Then the SIPI initiator appends the CRC to the end of the frame before transmission.

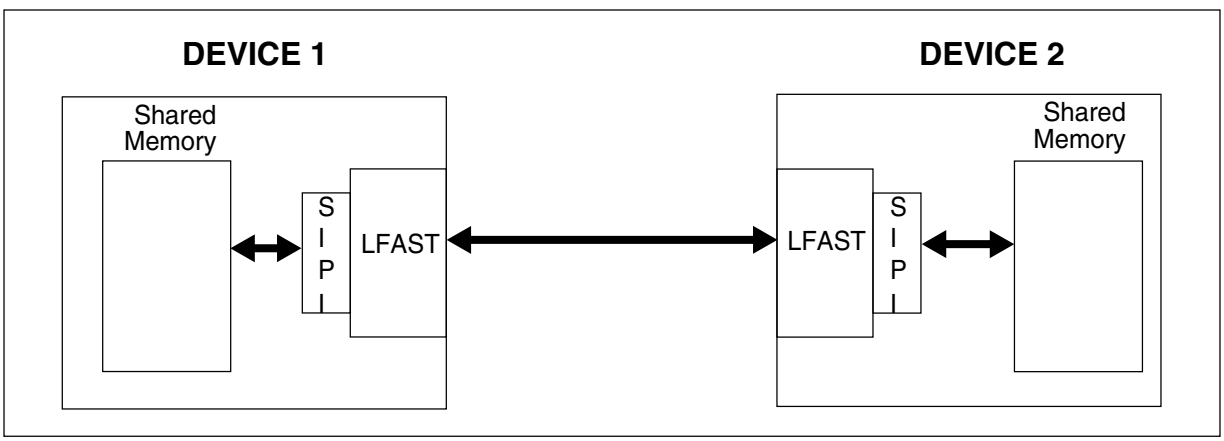


Figure 50-1. Interprocessor communication diagram

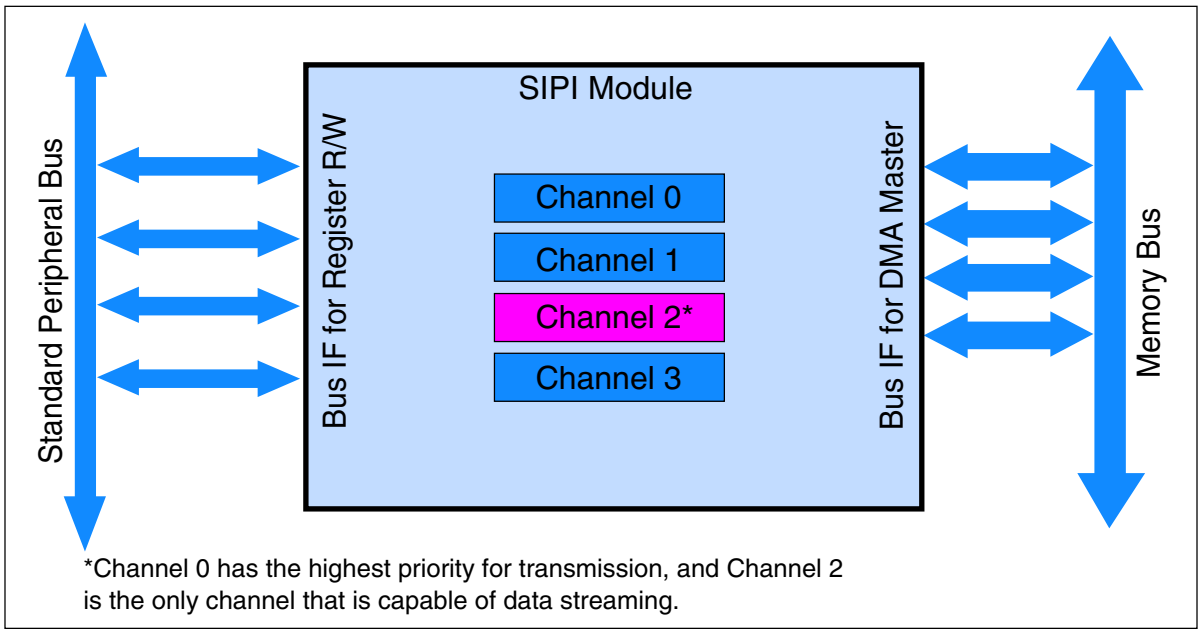
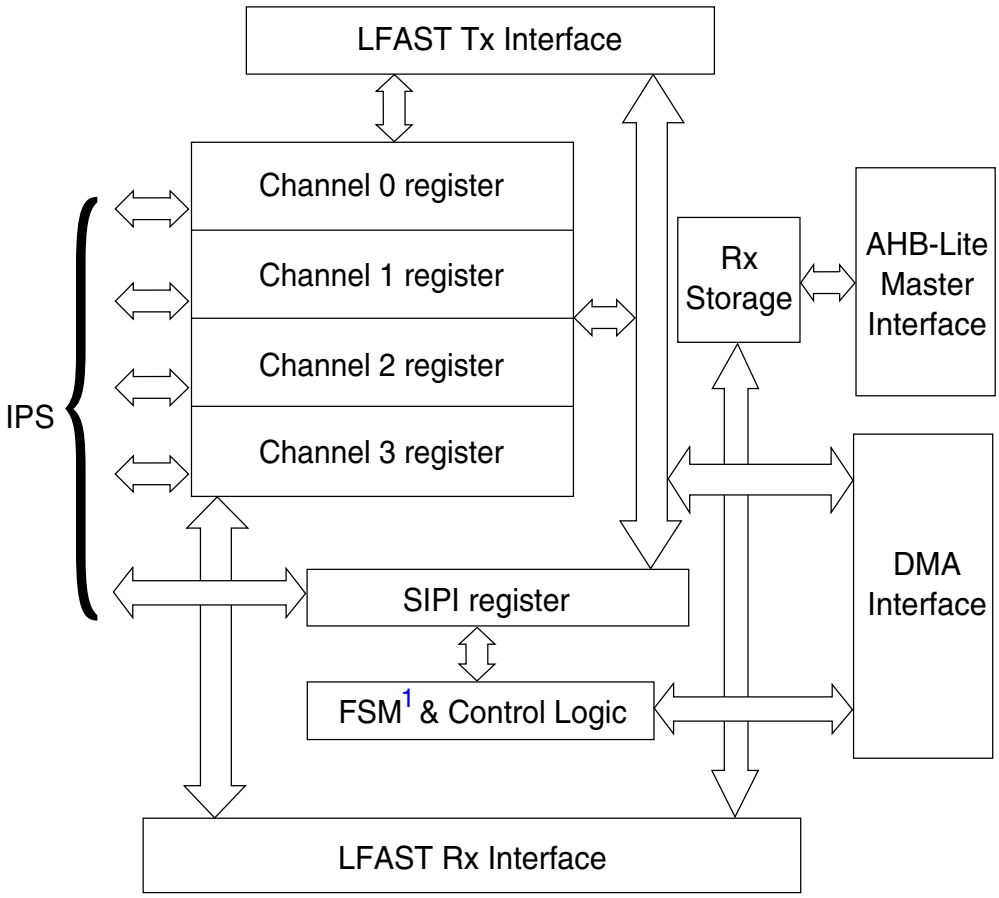


Figure 50-2. SIPI module

50.3 SIPI block diagram



1 Finite State Machine

Figure 50-3. SIPI block diagram

50.4 Feature description

This section describes the features of the SIPI module.

50.4.1 Main features

- Point-to-point communication between two devices
- Full duplex communication
- Four channels, including one channel with data streaming capability

- Configurable DMA access for each channel
- CRC protection mechanism
- Timeout protection mechanism
- Fixed priority channel selection
- Data size up to 256 bits on streaming channel
- Common tag pool for assigning sequential transfer IDs to every new transfer
- AHB master interface which is used by target node to access shared memory
- Target node contains a set of nine 32-bit internal registers to store commands
- Up to two outstanding requests are supported at initiator

50.4.2 Standard features

- IPS bus interface (PBRIDGE)
- SPP DMA2x bus interface
- AHB Master Interface
- LFAST Tx/Rx interfaces
- Cyclic Redundancy Check error detection (CRC16)

50.5 SIPI operation from reset

When the SIPI module exits reset, it is operational and target mode is enabled without the need to configure the control registers.

50.6 Functional description

50.6.1 External signals

The SIPI has no chip external signals.

50.6.2 Frame format

All frames have the same general format:

- 16 bit header
- Address, Address and Data, or nothing
- 16-bit CRC

There are 2 main groups of command; read and write. Within those 2 groups are three read/write formats:

- 32-bit
- 16-bit
- 8-bit

Each command generates one of three different responses:

- Read data
- Write acknowledge
- Error

There is one additional command that requests the module ID from the Target node. The ID is a unique 32-bit ID that is specific to a particular device. It is normally the same as the JTAG ID. The sections below illustrate the four different frame formats that are used to implement all the commands and responses.

The number of frames depends on the data buffers (associated with every channel), the frame data is stored in data buffers at the initiator. Address and data are both transmitted in the same order in which they are stored.

50.6.2.1 Register read request

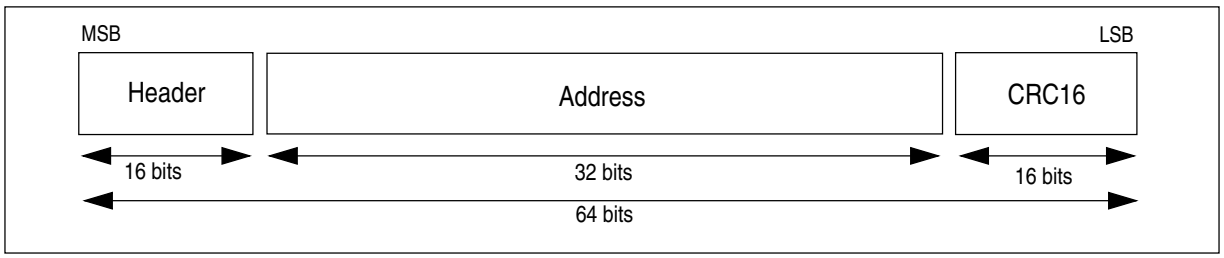


Figure 50-4. SIPI register read request

50.6.2.2 Register read response, ID request response

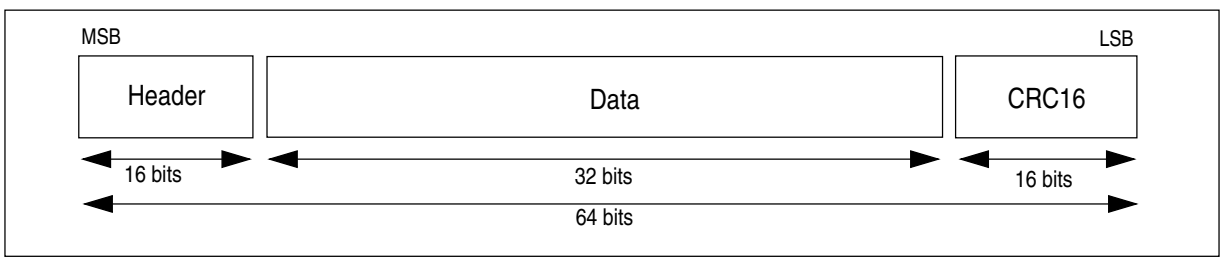


Figure 50-5. SIPI read response

50.6.2.3 Register write request

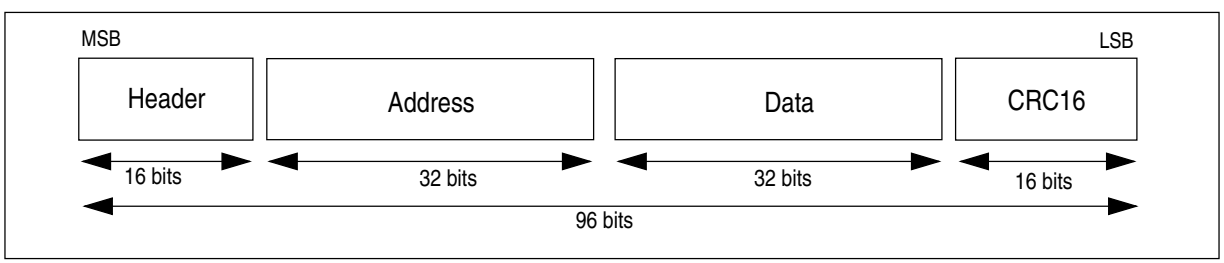


Figure 50-6. SIPI register write request

50.6.2.4 Trigger transfer, ID transfer, write acknowledge and streaming write acknowledge

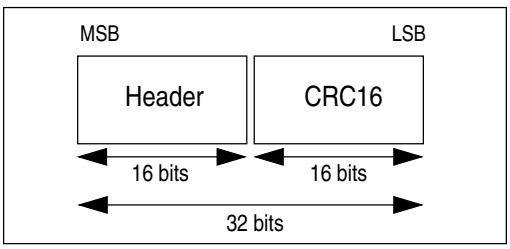


Figure 50-7. SIPI write acknowledge

50.6.2.5 Streaming write request

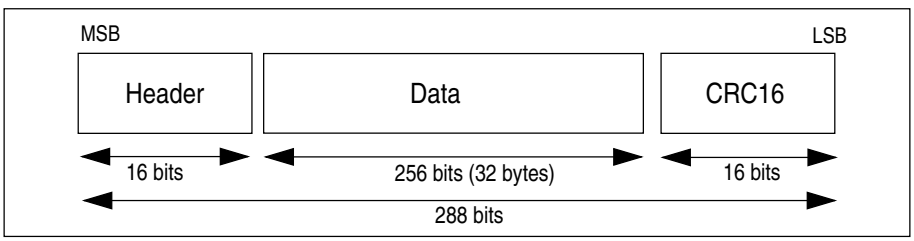


Figure 50-8. Streaming write request format

NOTE

Direct write operations are used to set the streaming address.

Streaming data write is performed using the format shown in Figure 50-8.

50.6.2.6 Header field

Header field contains 16 bits of configuration information. MSB will be transmitted first.

50.6.2.6.1 SIPI header coding

Figure 50-9, Table 50-1, Figure 50-10, and Table 50-2 show how the SIPI header bits are coded.

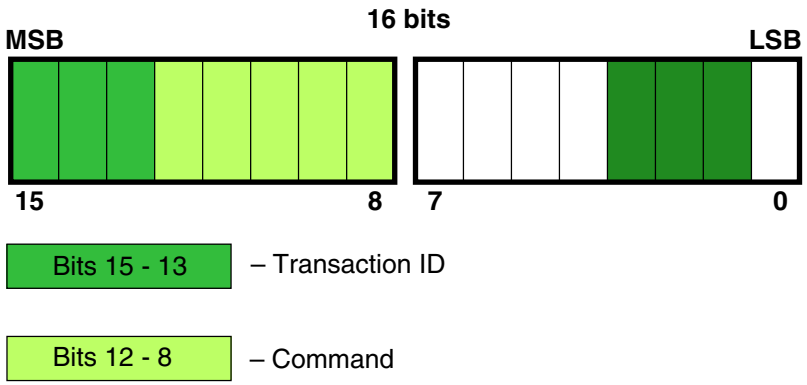


Figure 50-9. SIPI header coding

Table 50-1 shows the command coding for the bits[12:8] of the header.

Table 50-1. SIPI header command coding

b[12:8] (hex)	Command	Payload Size
00	Read 8 bits	64
01	Read 16 bits	64
02	Read 32 Bits	64
03	Reserved	—
04	Write 8 bits with ACK	96
05	Write 16 bits with ACK	96
06	Write 32 bits with ACK	96
07	Reserved	—
08	ACK – OK	32
09	ACK – Fault	32
0A	Read Answer – OK	64
0B	Reserved	—
0C	Trigger comm and with ACK	32
0D	Reserved	—
…	…	…
11	Reserved	—
12	ID Register Read Request	32
13	Reserved	—
…	…	…
16	Reserved	—
17	Stream Data with ACK (32 bytes)	288
18	Reserved	—
…	…	…
1F	Reserved	—

Figure 50-10 shows the channel number field in the header and Table 50-2 shows the channel number coding.

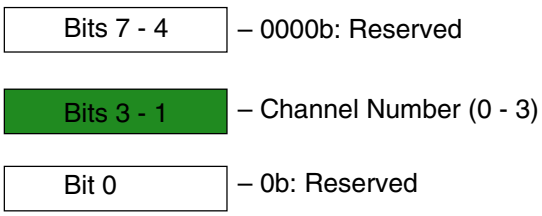


Figure 50-10. SIPI header channel field

Table 50-2. SIPI header channel number coding

SIPI channel name	Channel number coding in header(s)		Comment
	SIPI channel coding select field (SIPI_MCR[CHNSB])		
	Code table I (SIPI_MCR[CHNSB] = 1)	Code table II (SIPI_MCR[CHNSB] = 0)	
0 (Channel A)	000b	100b	In use
1 (Channel B)	001b	101b	In use
2 (Channel C)	010b	110b	In use
3 (Channel D)	011b	111b	In use
4 (Channel E)	100b	000b	Reserved
5 (Channel F)	101b	001b	Reserved
6 (Channel G)	110b	010b	Reserved
7 (Channel H)	111b	011b	Reserved

50.6.2.7 Address field

The address field is 32 bits wide with the MSB transmitted first.

50.6.2.8 Payload field

Table 50-3 shows the payload sizes of LFAST frames.

Table 50-3. Payload size of LFAST channel frame

LFAST Code	SIPI Code	LFAST Payload Size (bits)	LFAST Payload Size (bytes)
000b	—	8	1
001b	—	32	4
010b	010b	64	8
011b	011b	96	12
100b	100b	128	16

Table continues on the next page...

Table 50-3. Payload size of LFAST channel frame (continued)

LFAST Code	SIPI Code	LFAST Payload Size (bits)	LFAST Payload Size (bytes)
101b	101b	256	32
110b	110b	512	64
111b	111b	288	36

Table 50-4 shows the converted coding of LFAST for a given SIPI code.

Table 50-4. Converted coding of LFAST channel code for SIPI headers

LFAST Code	SIPI Code	Channel (hex)[1]
0100b	100b	A
0101b	101b	B
0110b	110b	C
0111b	111b	D
1000b	000b	E (not used by SIPI)
1001b	001b	F (not used by SIPI)
1010b	010b	G (not used by SIPI)
1011b	011b	H (not used by SIPI)

1. SIPI channel 0 sends all commands on LFAST channel A, commands received on LFAST channel A, are processed and the response sent back on LFAST channel A.
SIPI channel 1 sends all commands on LFAST channel B, commands received on LFAST channel B, are processed and the response sent back on LFAST channel B.
SIPI channel 2 sends all commands on LFAST channel C, commands received on LFAST channel C, are processed and the response sent back on LFAST channel C.
SIPI channel 3 sends all commands on LFAST channel D, commands received on LFAST channel D, are processed and the response sent back on LFAST channel D.

50.6.2.9 CRC field

CRC field is 16 bits wide with calculation always enabled using CRC-16-CCITT syndrome (x16 + x12 + x5 + 1). MSB is sent first in the data stream.

50.6.2.9.1 CRC field examples

50.6.2.9.1.1 Example 1 – 32 bit write on channel 1 with ID1

- Header = 260Ah
- Address = 1122_3344h
- Data = CCDD_EEFFh
- CRC = BF7Dh

50.6.2.9.1.2 Example 2 – 32 bit read on channel 2 with ID2

- Header = 420Ch
- Address = 89AB_CDEFh
- CRC = 6B80h

50.6.2.9.1.3 Example 3 – Event command on channel 3 with ID3

- Header = 6C0Eh
- CRC = B286h

50.7 Transfer types

This section describes the available transfer types of the SIPI module. The SIPI frame is inserted inside the payload of the LFAST frame as shown in the figures of the following examples.

50.7.1 Read transfer

A Read transfer can be of two types:

- Read Request Transfer (at initiator node)
- Read Response Transfer (at target node)

50.7.1.1 Register read request transfer

If there is a read request transfer, the initiator node will send header, address and CRC bits as shown in Figure 50-11.

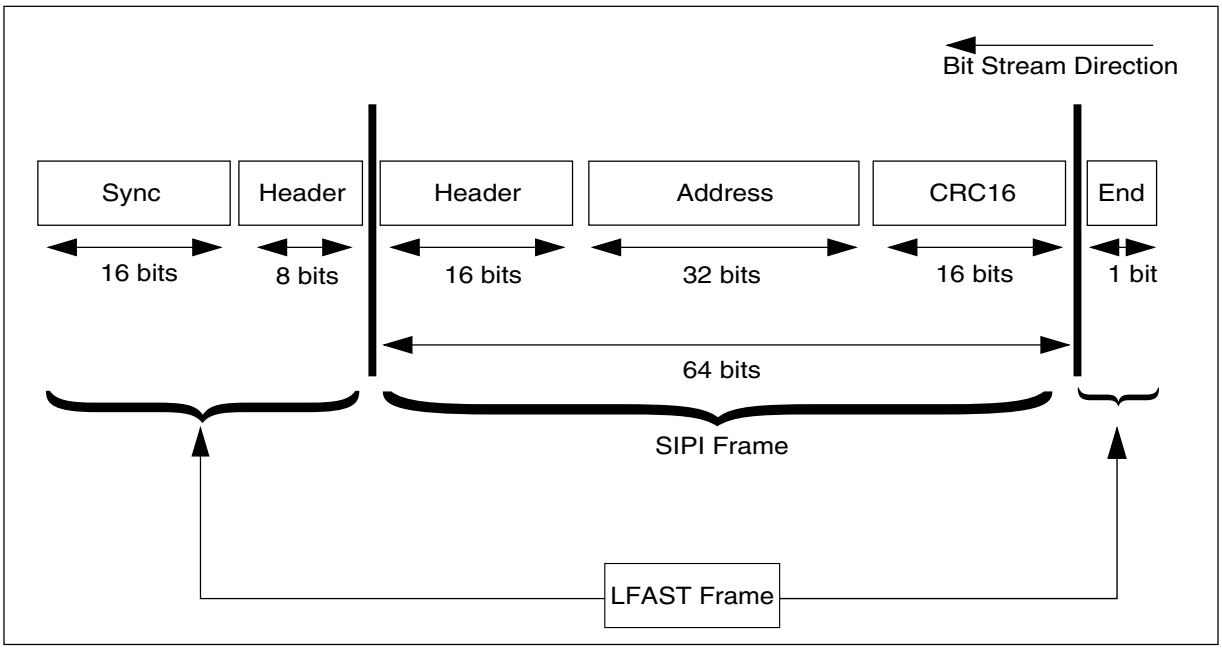


Figure 50-11. Read request transfer

50.7.2 Register read answer transfer

In response to a read request by the initiator node, the target node will send header, payload and CRC16. Data transfer could be in 8-bit, 16-bit or 32-bit modes (see Figure 50-12). In the case of 8-bit or 16-bit modes, copies of the SIPI data is sent in the payload (see Figure 50-13).

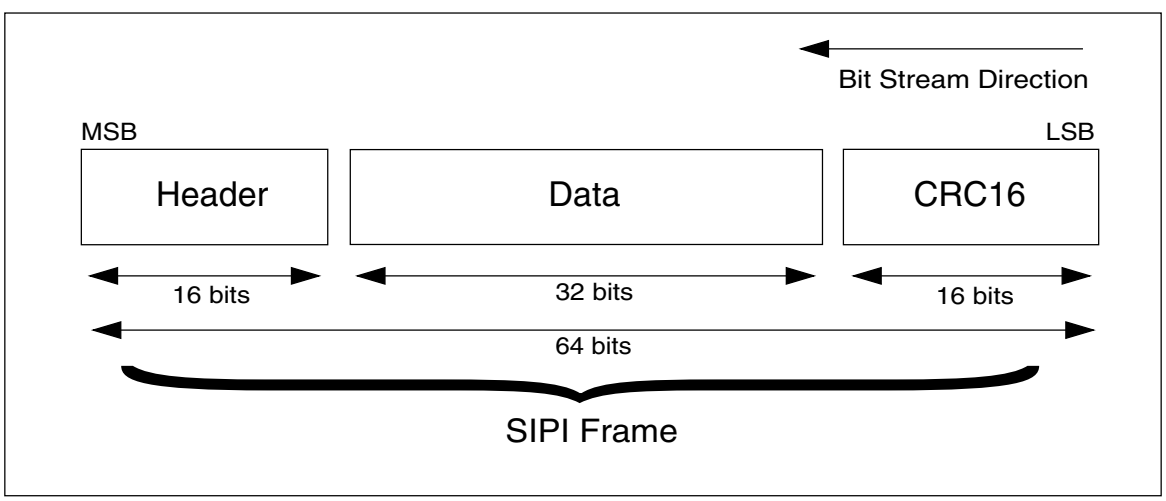


Figure 50-12. Read answer transfer

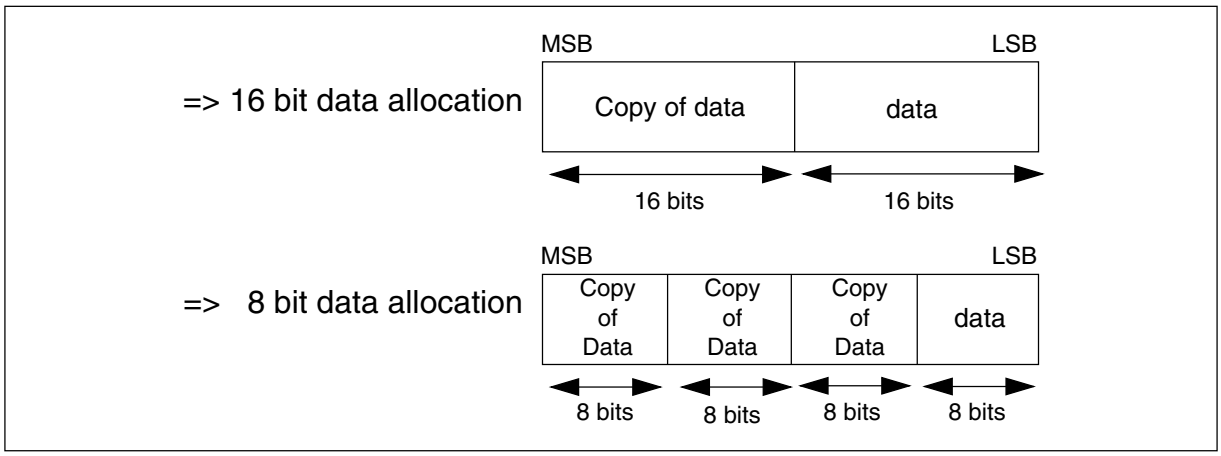


Figure 50-13. Data allocation

Note

- For an 8-bit transfer, address bits [31:2] are used as address and bits [1:0] are used as byte enables.
  - Bits[1:0]:
    - 00 - byte 3 enabled (MSB)
    - 01 - byte 2 enabled
    - 10 - byte 1 enabled
    - 11 - byte 0 enabled (LSB)
- For a 16-bit transfer, address bits [31:1] are used as address and bit [0] is used as halfword enable.
  - Bit[0]:
    - 0 - halfword 1 enabled (MSB)
    - 1 - halfword 0 enabled (LSB)

50.7.3 Register Write transfer

Register Write transfer can be of two types:

- Normal write transfer - channels 0, 1, 2 and 3
- Streaming data transfer - channel 2 only

A Register Write transfer can be done through Normal Write transfer on channels 0, 1, 2 and 3 (see Normal write transfer) as shown in Figure 50-14.

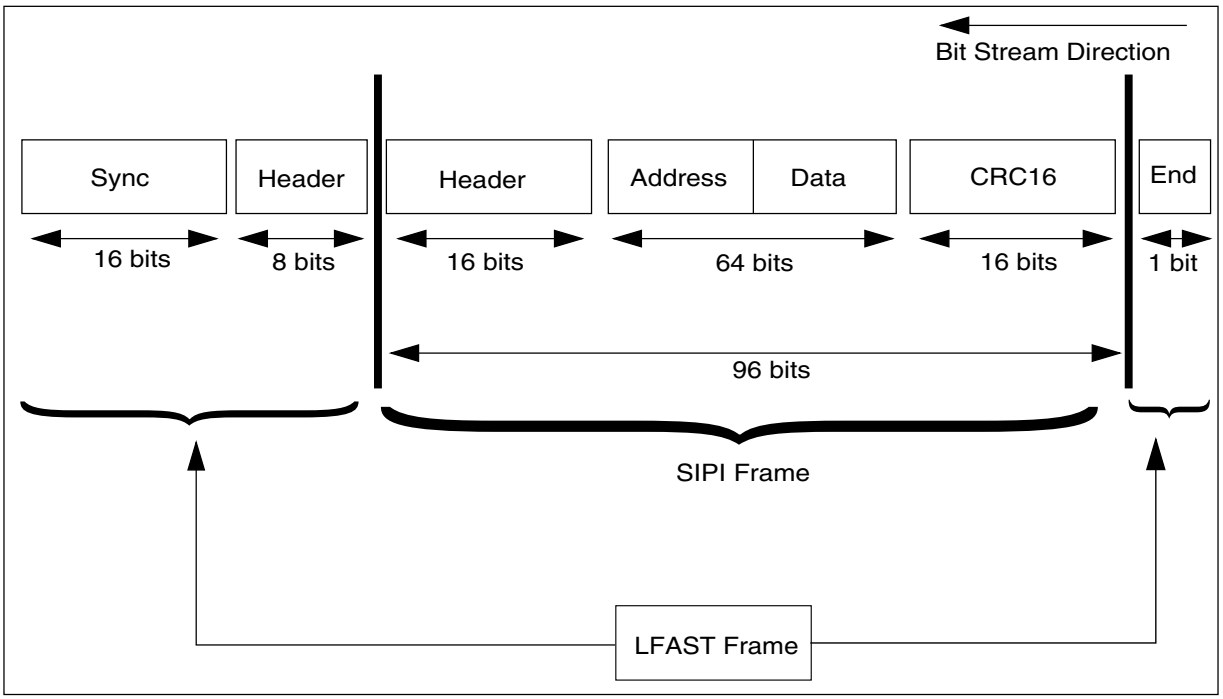


Figure 50-14. Register write transfer (showing LFAST frame encapsulation)

50.7.3.1 Normal write transfer

A normal write transfer contains header, address, data (32-bit) and CRC as shown in Figure 50-15.

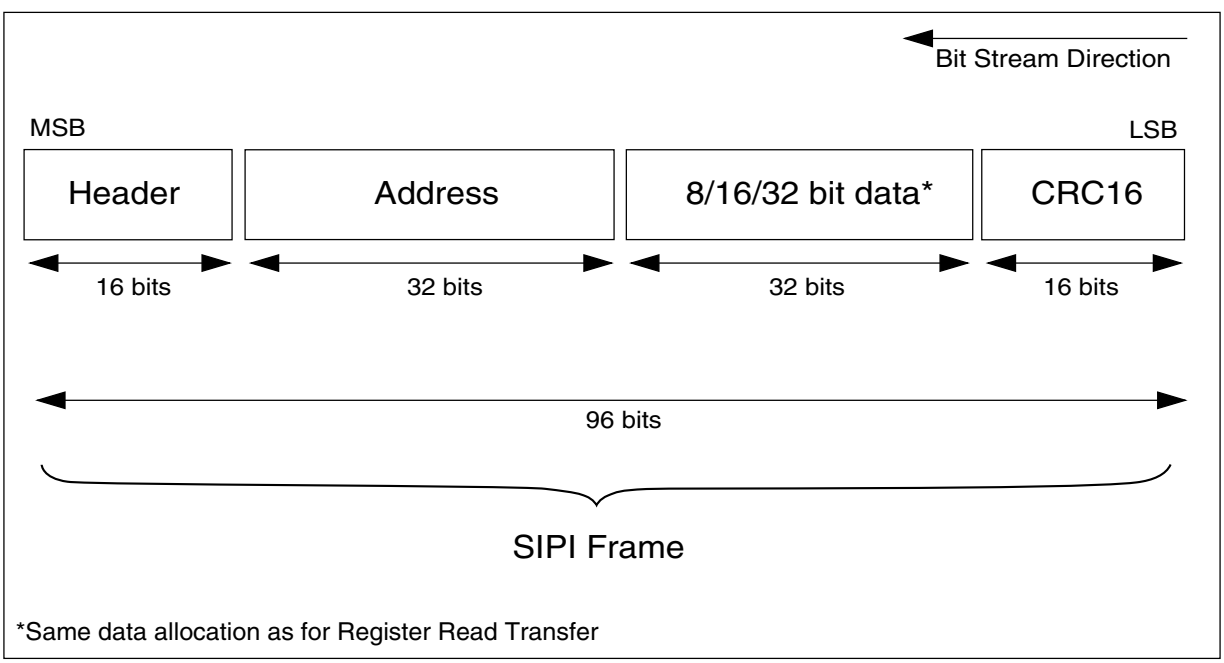


Figure 50-15. Write transfer (LFAST frame encapsulation is not shown)

Note

- For an 8-bit transfer, address bits [31:2] are used as address and bits [1:0] are used as byte enables.
  - Bits[1:0]:
    - 00 - byte 3 enabled (MSB)
    - 01 - byte 2 enabled
    - 10 - byte 1 enabled
    - 11 - byte 0 enabled (LSB)
- For a 16-bit transfer, address bits [31:1] are used as address and bit [0] is used as half word enable.
  - Bit[0]:
    - 0 - half-word 1 enabled (MSB)
    - 1 - half-word 0 enabled (LSB)

50.7.3.2 Streaming write transfer

Streaming write transfer has 256 bits of payload.

50.7.3.2.1 Streaming write transfer with address

Setting the address is performed using a direct write transfer. The SIPI Maximum Count Register (SIPI_MAXCR) and SIPI Address Reload Register (SIPI_ARR) are written before the SIPI Address Count Register (SIPI_ACR). This is to avoid unwanted behavior of the SIPI attempting to access non-shared memory. This is only true for the very first streaming command, subsequent streaming command(s) may not need to write the SIPI_MAXCR and SIPI_ARR (see SIPI Max Count Register (SIPI_MAXCR), SIPI Address Count Register (SIPI_ACR) and SIPI Address Reload Register (SIPI_ARR)).

50.7.3.2.2 Streaming transfer with data

Figure 50-16 shows the packet structure of the streaming transfer containing data.

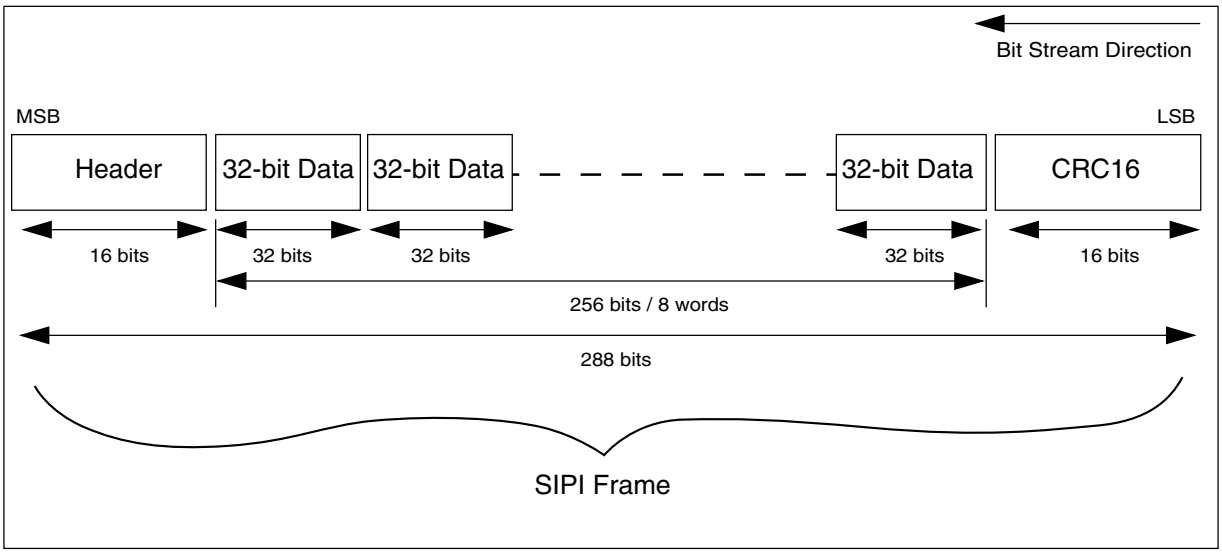


Figure 50-16. Streaming transfer with data

50.7.4 Write Acknowledge transfer

A Write Acknowledge transfer contains only header and CRC bits (see Write Aknowledge transfer. The CRC bits are calculated on the header field.

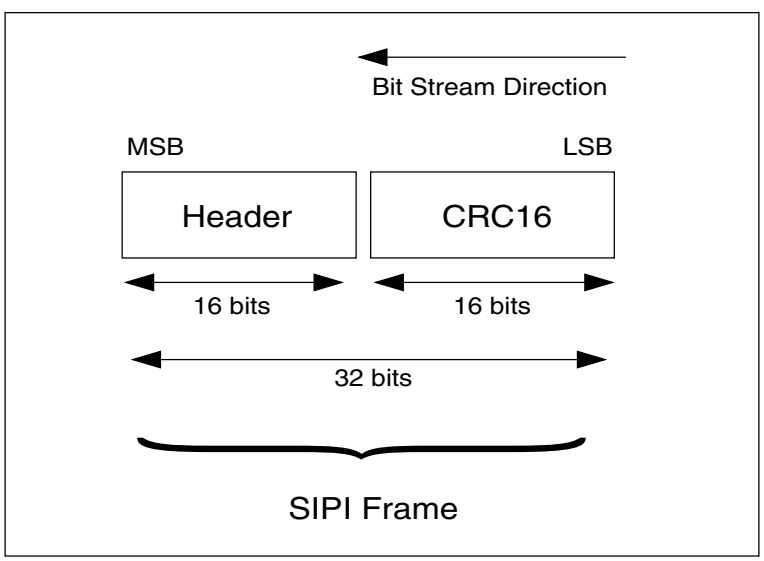


Figure 50-17. Write Aknowledge transfer (LFAST encapsulation is not shown)

50.7.5 ID request response

50.7.5.1 ID request transfer

An ID request is transmitted by the initiator node as shown in Figure 50-18.

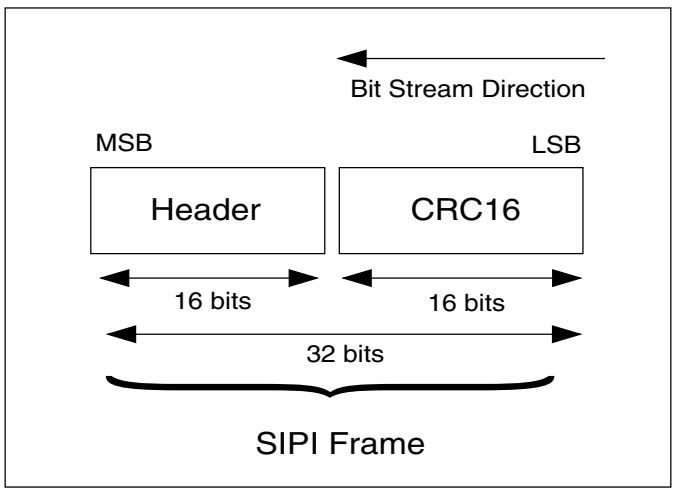


Figure 50-18. ID request transfer (LFAST encapsulation is not shown)

50.7.5.2 ID request response transfer

An ID request response is transmitted by the target node as shown in Figure 50-19.

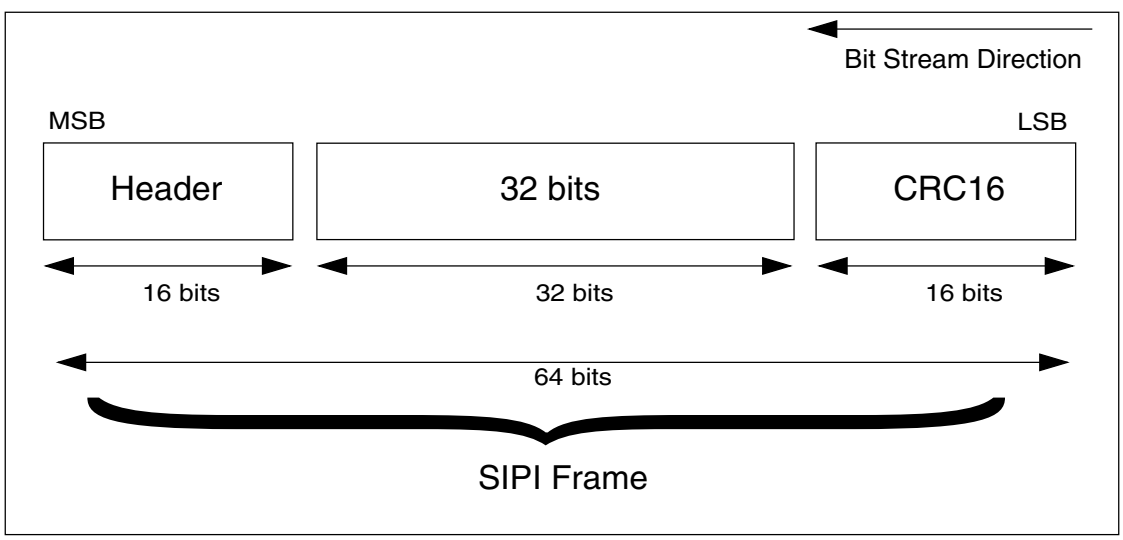


Figure 50-19. ID request response transfer

Note

The ID request response contains the value of the CHIP ID in place of data.

50.8 Transfer API and flow charts

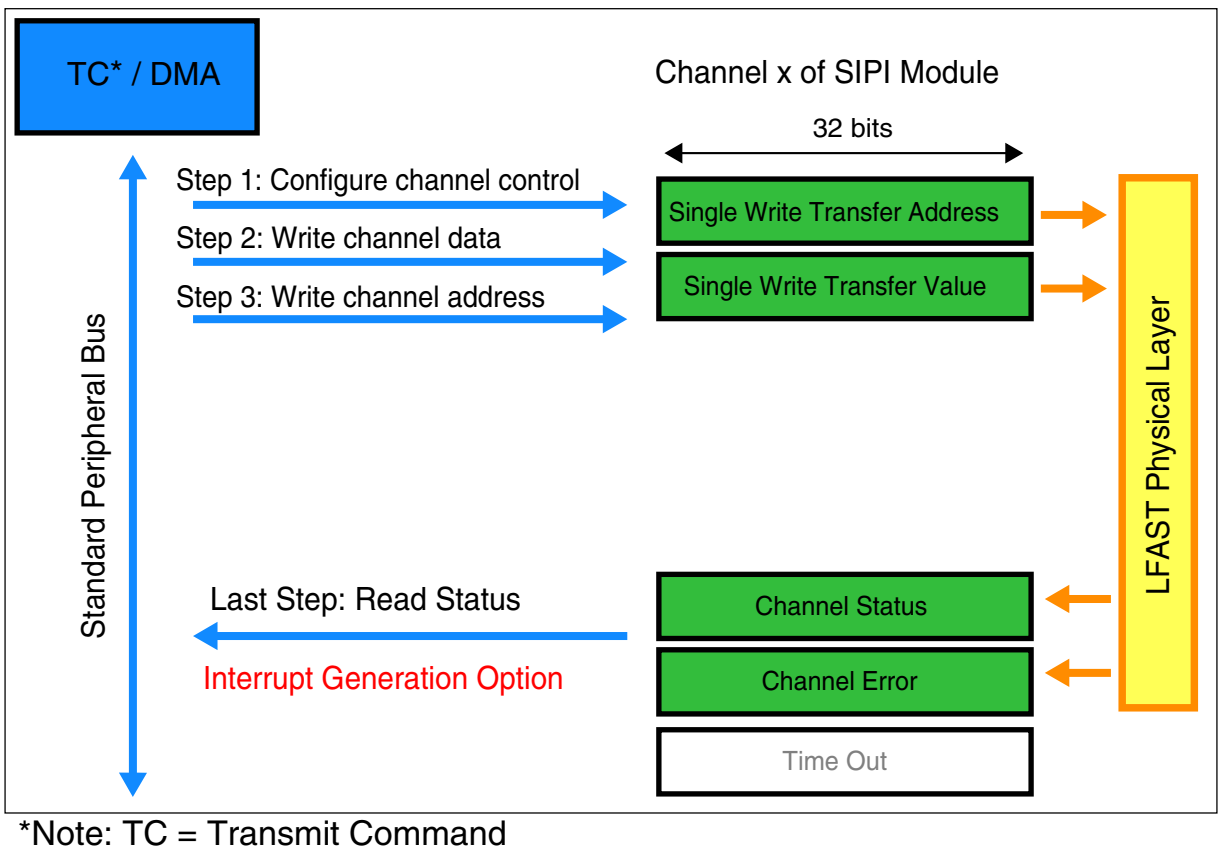


Figure 50-20. SIPI single register write API

Implement the following steps to generate a single Write Transfer Request (Figure 50-20):

1. Configure data and SIPI_CCR*n* at the initiator node.
2. Configure SIPI_CAR*n* at the initiator node.

   As soon as the channel address register (SIPI_CAR*n*) is written and if CCR*n*[CHEN] = 1, the initiator SIPI will calculate the CRC on header, address and data field and will start transmitting data to LFAST.
3. Software polls the status register bits (SIPI_CSR*n*) to determine when the request completes, SIPI_CSR*n*[ACKR] = 1. An interrupt will be generated if the corresponding SIPI_CIR*n*[ACKIE] = 1.

On a single Write transfer request reception (Figure 50-21):

1. Target node will place the address, data and control information on its AHB master interface.
2. When the process in completed, the target node will generate an acknowledge frame and send it back to the LFAST.

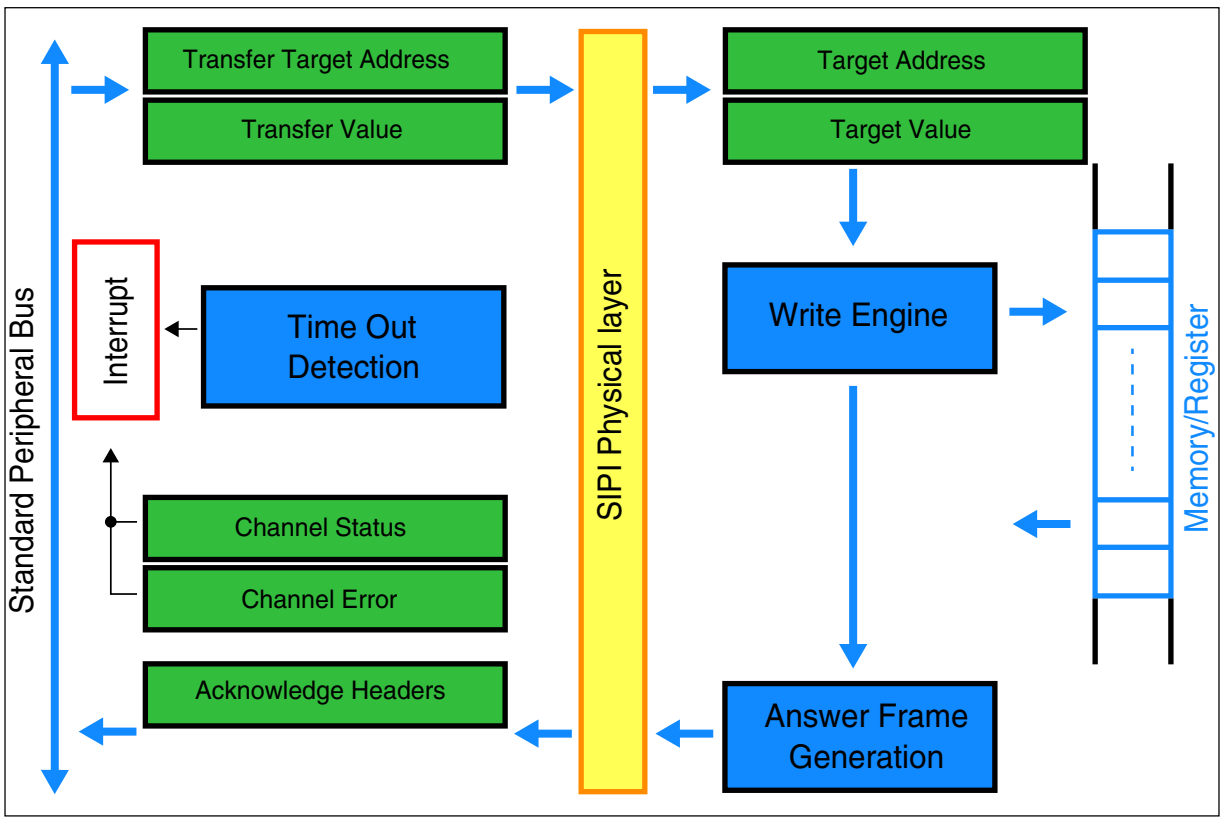


Figure 50-21. SIPI Single Register Write API – Flow Chart

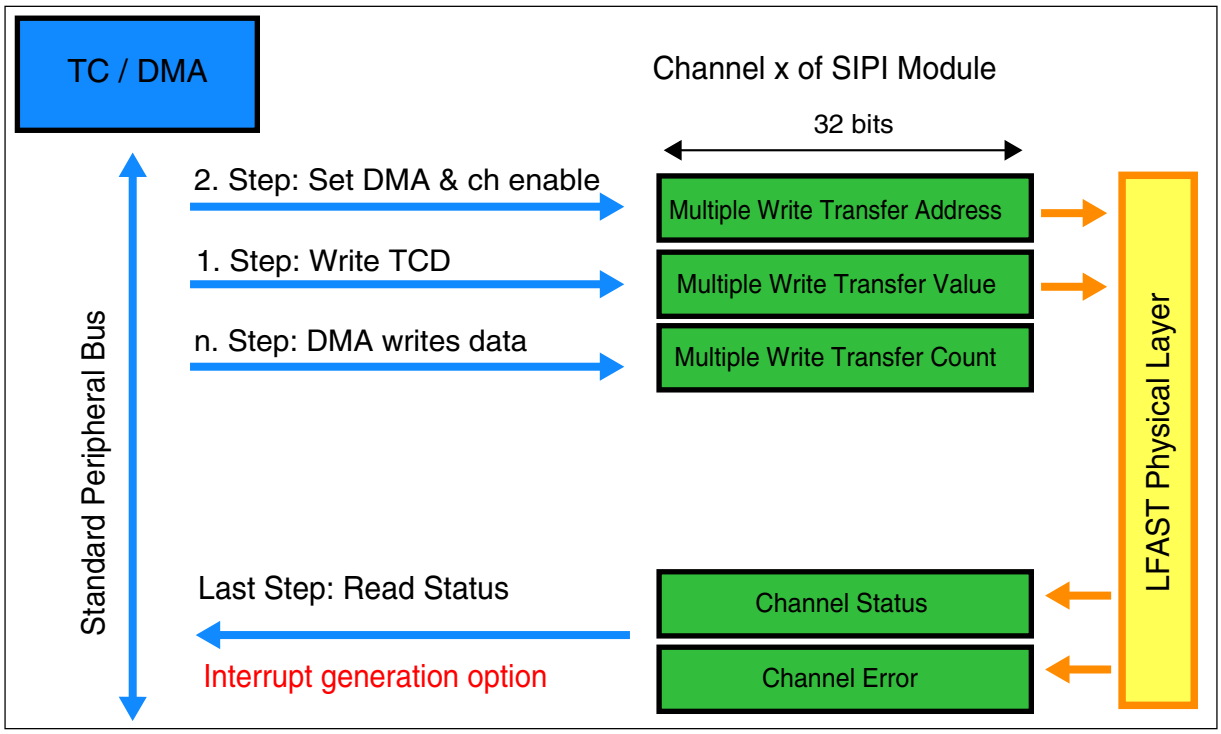


Figure 50-22. SIPI Multiple Register Write API

For multiple write transfer request generation (Figure 50-22):

1. Software will configure the Transfer Control Descriptor (TCD) of the DMA.
2. Software will write SIPI_CCR*n*[CHEN] = 1 and SIPI_CCR*n*[DAN] = 1.
3. SIPI will start copying data into the SIPI_CDR*n* through its DMA interface, depending on the transfer count and data registers size. SIPI_CDR2_0 should be written with the MSB.
4. When the copying process is completer, initiator SIPI will calculate CRC on header, address and data field and start transmitting data to LFAST.
5. Software should poll the SIPI_CSR*n* status register bits to determine if the request has completed. If SIPI_CSR*n*[ACKR] = 1, an interrupt will be generated (if the corresponding SIPI_CIR*n*[ACKIE] = 1).
6. If SIPI_CCR*n*[DEN] = 1, the SIPI request will transfer to the DMA controller. If not, the state machine go idle.
7. Steps 4–7 will be repeated.

On Multiple Write transfer request reception (Figure 50-22):

1. Target node will place the address, data and control information on its AHB Master Interface.
2. When the process in completed, target node will generate an acknowledge frame and send it back to the LFAST.

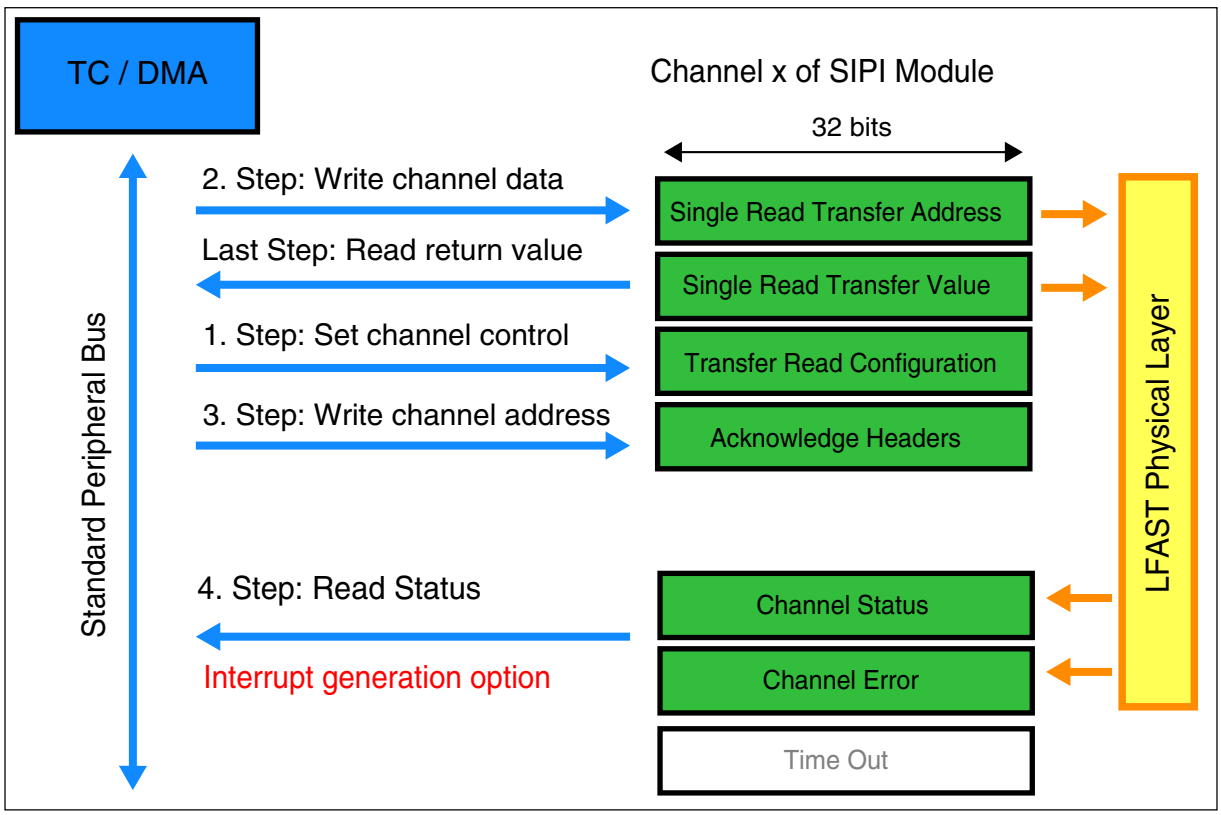


Figure 50-23. SIPI Single Register Read API

For Single Read Transfer Request generation (Figure 50-23):

1. Software/DMA will configure SIPI_CCR*n* and SIPI_CDR*n*, with the last step by writing CAR*n*.
2. As soon as SIPI_CAR is written, the initiator SIPI will calculate CRC of the header and address fields and start transmitting data to the LFAST.
3. Software should poll CSR*n* to determine when the request has completed. If SIPI_CSR*n*[RAR] = 1, an interrupt will be generated (if SIPI_CIR*n*[RAIE] = 1). If SIPI_CSR*n*[RAR] does not set then SIPI_ERR[TOE*n*] = 1 which indicates a timeout has occured.
4. If SIPI_CSR*n*[RAR] = 1 then software can read the data register, or can other necessary action if SIPI_CSR*n*[RAR] = 0.

On Single Read transfer request reception (Figure 50-23):

1. Target node will place the address and control information on its AHB Master Interface.
2. When the process in completed, target node will send read response back to LFAST.

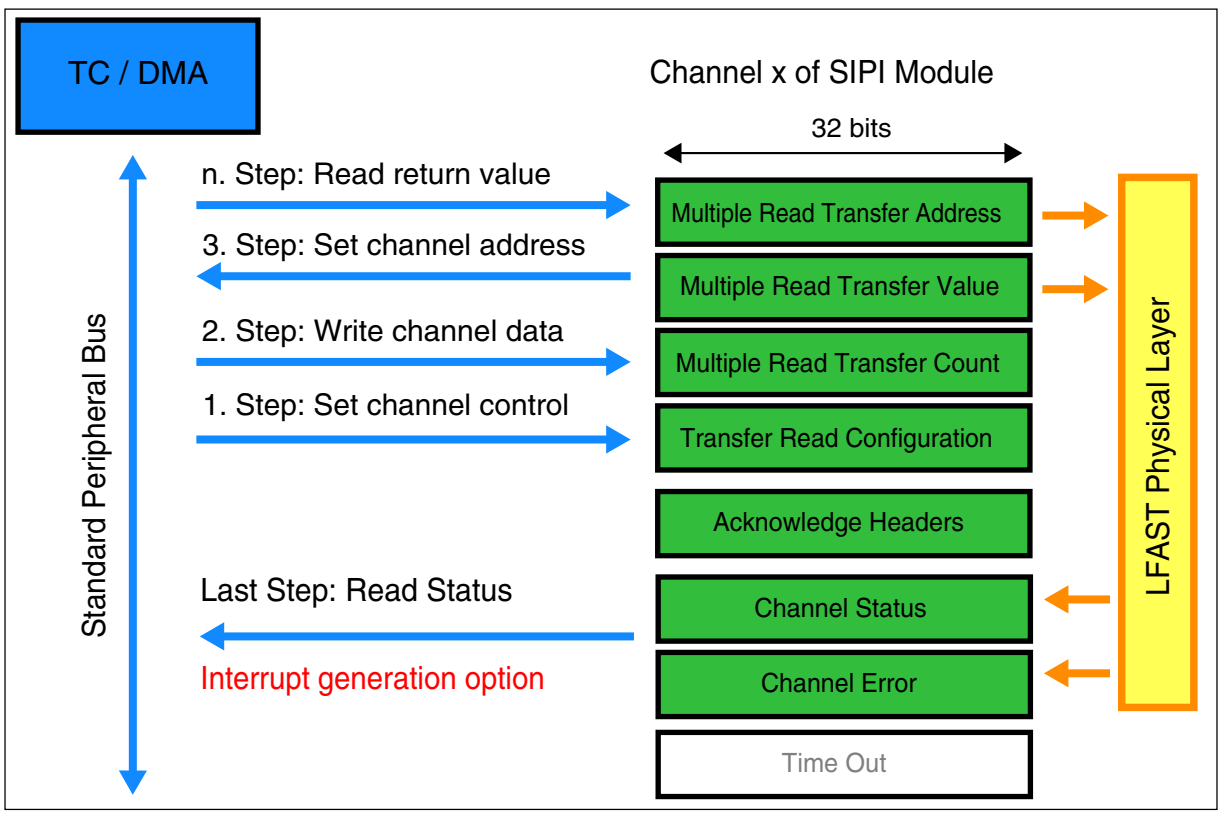


Figure 50-24. SIPI multiple register read API

For multiple read transfer request generation (Figure 50-24):

1. Software/DMA will configure SIPI_CCR*n* and SIPI_CDR*n*, with the last step by writing SIPI_CAR*n*.
2. As soon as channel address register is written, initiator SIPI will calculate CRC on the header and address fields, then starts transmitting data to the LFAST.
3. Software should poll SIPI_CSR*n* to determine when the request has completed. If SIPI_CSR*n*[RAR] = 1 and SIPI_CIR*n*[RAIE] = 1 an interrupt will be generated. If SIPI_CSR*n*[RAR] does not set, then SIPI_ERR[TOE*n*] = 1 which indicates a timeout.
4. Steps 2–3 will be repeated for multiple requests.

On Multiple Read transfer request reception (Figure 50-24):

1. Target node then will start reading data through its AHB interface.
2. When the transfer is completed/all data registers are full, it will calculate CRC and send the response frame back to LFAST.

Figure 50-25, Figure 50-26, Figure 50-27 and Figure 50-28 show the SIPI data streaming of data.

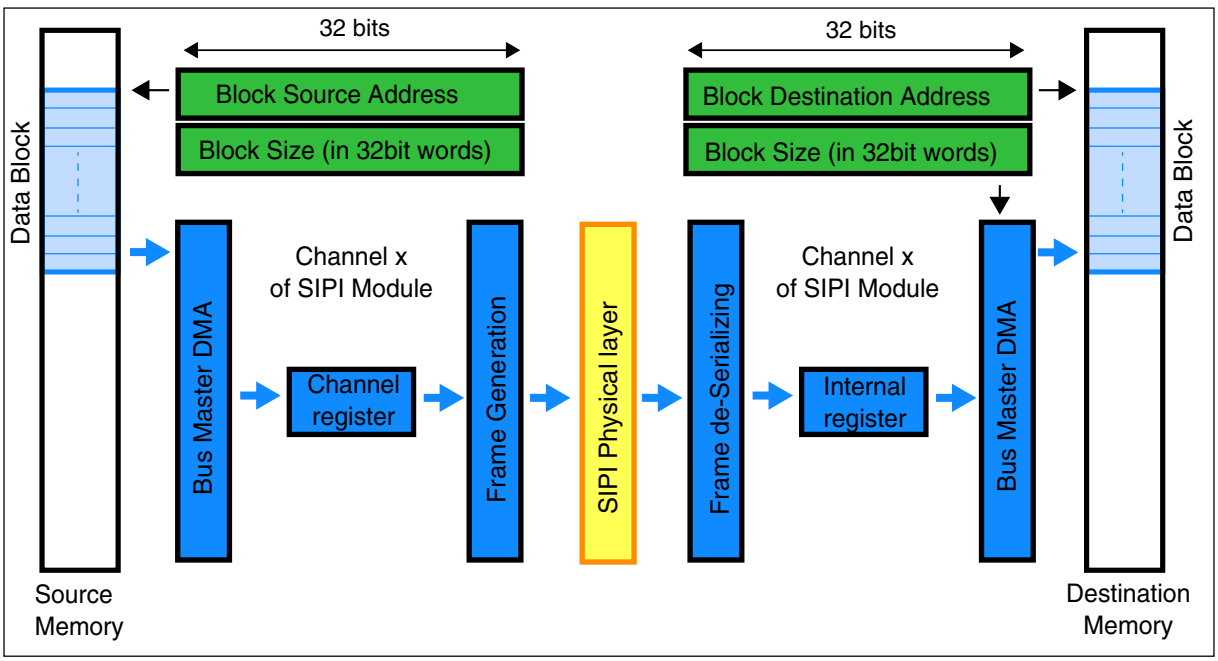


Figure 50-25. SIPI data stream model

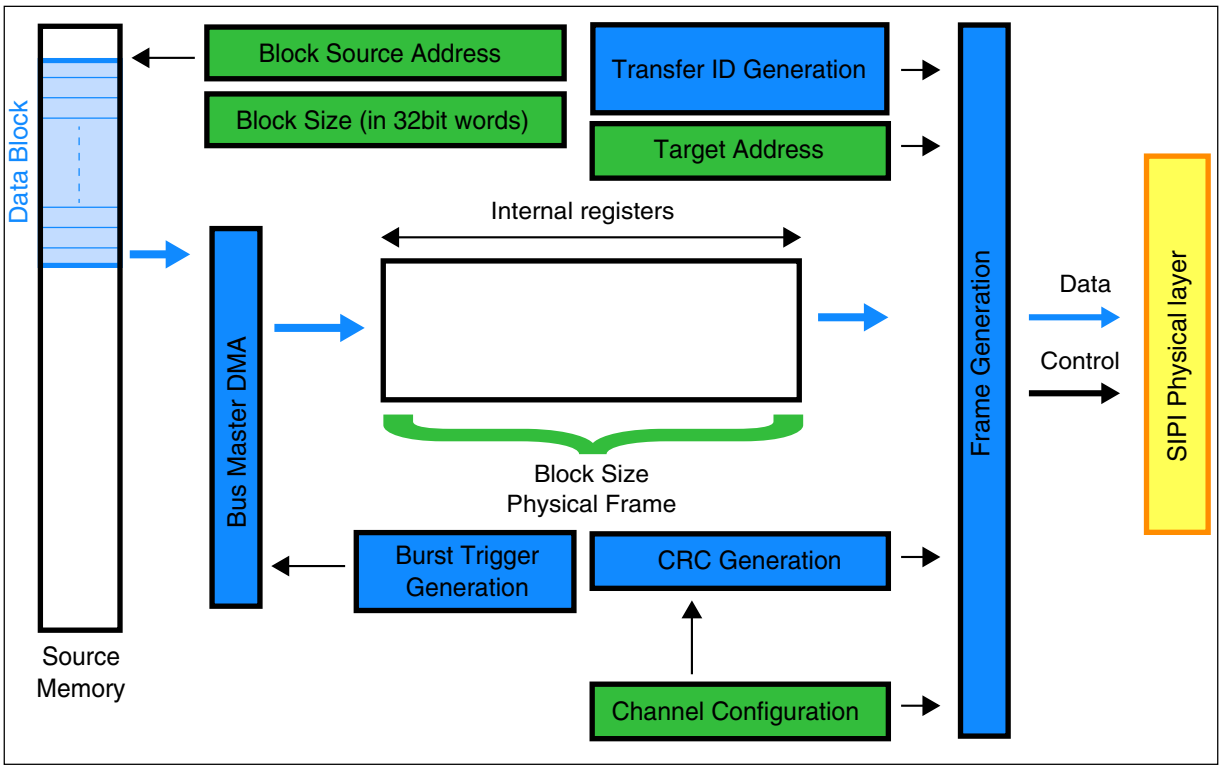


Figure 50-26. SIPI data stream model - Initiator

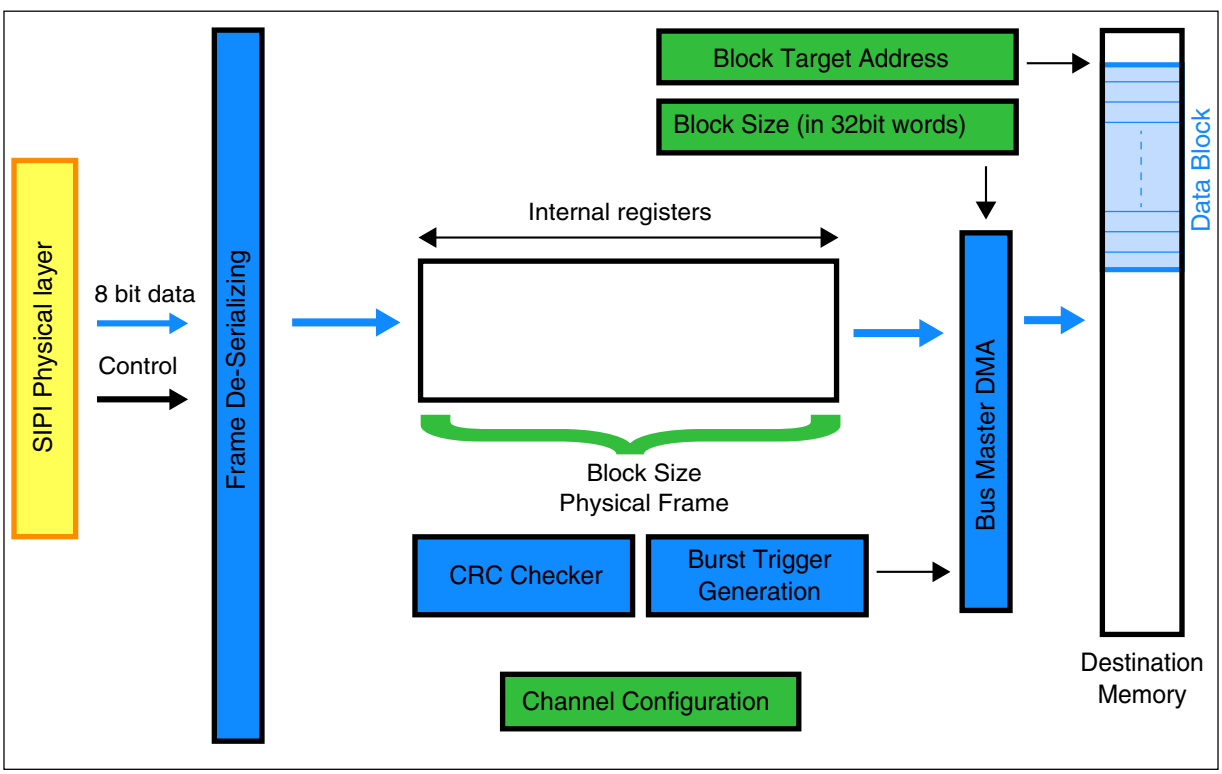


Figure 50-27. SIPI data stream model - Target

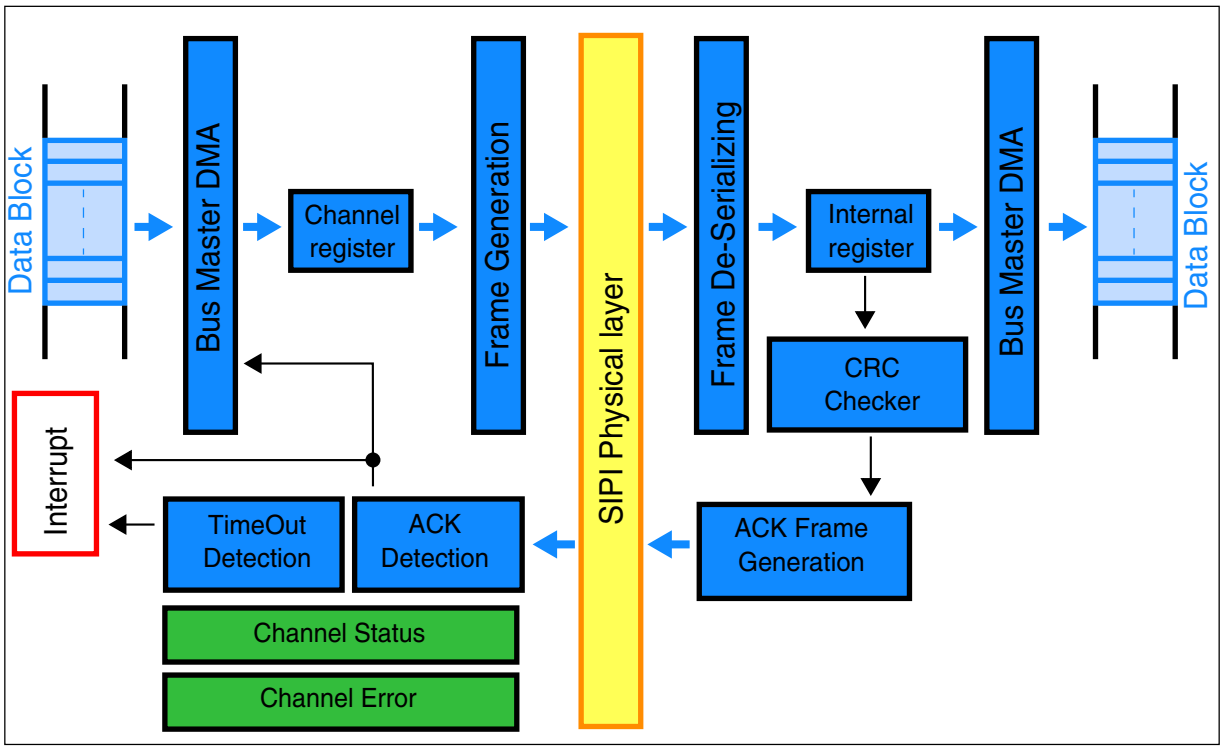


Figure 50-28. SIPI data stream with acknowledge trigger

50.9 DMA programming sequence

The DMA programming sequence is as follows:

1. Software configures TCD (DMA data).
2. DMA transfers from RAM to SIPI registers.
3. SIPI starts transferring the data through LFAST Tx ports.
4. If an acknowledge or fault response is received, SIPI will repeat the process. If any error response is received, SIPI will move to an idle state. An interrupt will be flagged by SIPI, and software will take control of the SIPI.
5. If SIPI has received a read response, and error flags are not set on the LFAST Rx port, SIPI will assert ipd_request. In case of error, move to Step 8.
6. As soon as ipd_request is asserted, DMA transfer starts between the SIPI registers and RAM.
7. If DMA transfer minor loop/major loop is complete, and the request is negated, ipd_request is negated.
8. If an error occurs:

- The SIPI generates an interrupt
- Software takes control of the SIPI

9. Steps 5 – 8 are repeated until the transfer is finished.

50.10 Modes of operation

SIPI has three operating modes:

- Initialization
- Normal
- Module Disable

After hardware reset the SIPI is in module disable mode, which helps reduce power consumption.

50.10.1 Initialization mode

To enter Initialization mode software writes SIPI_MCR[INIT]=1 SIPI_MCR[MOEN] must be set before attempting to write SIPI_MCR[INIT]). To exit Initialization mode software writes SIPI_MCR[INIT]=0 (see SIPI Module Configuration Register (SIPI_MCR)).

All message transfers are stopped when in Initialization mode. If software invokes Initialization mode when a bus transfer is in progress, the transfer will be aborted immediately even if the transfer has not completed.

Note

It is recommended that software checks the state of SIPI (SIPI_MCR[MOEN]) before setting SIPI_MCR[INIT].

50.10.2 Normal mode

Once software has completed initialization of SIPI (SIPI_MCR[INIT]=1), it can enter Normal mode by writing SIPI_MCR[INIT]=0. SIPI needs to be in Normal mode for all data transfers (see SIPI Module Configuration Register (SIPI_MCR)).

Note

SIPI must be enabled before a attempting to write SIPI_MCR[INIT] (SIPI_MCR[MOEN]=1).

50.10.3 Module Disable (MD)

MD mode in the SIPI is used to help reduce power consumption. By default, SIPI is in Disable mode, SIPI_MCR[MOEN]=0, and is exited by writing SIPI_MCR[MOEN]=1 (see SIPI Module Configuration Register (SIPI_MCR)). All the activities on the SIPI Tx and Rx ports are immediately stopped in Module Disable mode.

50.11 Errors

This section describes the potential errors that can occur during SIPI operation.

50.11.1 Timeout error

A timeout error is generated at the initiator node when the acknowledge/response is not received within the time configured in the corresponding CTOR*n*[TOR] field setting (see SIPI Channel Timeout Register 0 (SIPI_CTOR0), SIPI Channel Timeout Register 1 (SIPI_CTOR1), SIPI Channel Timeout Register 2 (SIPI_CTOR2) and SIPI Channel Timeout Register 3 (SIPI_CTOR3)). A timeout error is indicated when ERR[TOE*n*]=1 (see SIPI Error Register (SIPI_ERR)).

Note

SIPI should not drop the response even after a timeout occurs. Software will poll both error and status flags after the transfer to see if there was a timeout error. If there was a timeout error the response received may then be discarded.

50.11.2 CRC error

A CRC error is generated at the target nodes when the CRC received with the frame does not match the calculated CRC, and the SR[GCRCE] is set (see SIPI Status Register (SIPI_SR)). An interrupt will be asserted if MCR[CRCIE]=1 (see SIPI Module Configuration Register (SIPI_MCR)).

Note

The target node will not send an acknowledge to the initiator node when a CRC error is generated. An interrupt will be generated on the target side if the corresponding interrupt

enable bit is set. The initiator node will detect a timeout and take necessary action.

50.11.3 Maximum count reached error

The maximum count reached error is only generated at the target node. It is generated when the value of the SIPI_ACR is equal to the SIPI_MAXCR (see SIPI Max Count Register (SIPI_MAXCR) and SIPI Address Count Register (SIPI_ACR)). When the maximum count is reached, SIPI_SR[MCR] = 1 (see SIPI Status Register (SIPI_SR)). An interrupt will be generated if SIPI_MCR[MCRIE] = 1 (see SIPI Module Configuration Register (SIPI_MCR)).

50.11.4 Transaction ID error

The Transaction ID (TID) error is always generated at the initiator node only. It is generated when header bits 15–13 do not match SIPI_CSR*n*[TID] (transaction ID bits, see SIPI Channel Status Register 0 (SIPI_CSR0), SIPI Channel Status Register 1 (SIPI_CSR1), SIPI Channel Status Register 2 (SIPI_CSR2), and SIPI Channel Status Register 3 (SIPI_CSR3)). SIPI_CSR*n*[TIDE] = 1 when a TID error is detected, and an interrupt will be generated if SIPI_CIR*n*[TIDIE] = 1 (see SIPI Channel Interrupt Register 0 (SIPI_CIR0), SIPI Channel Interrupt Register 1 (SIPI_CIR1), SIPI Channel Interrupt Register 2 (SIPI_CIR2), and SIPI Channel Interrupt Register 3 (SIPI_CIR3)).

50.11.5 Acknowledge error

An incorrect acknowledge is received only from the initiator. When the acknowledge received is incorrect, SIPI_ERR[ACKR*n*] will be set (see SIPI Error Register (SIPI_ERR)). An interrupt will be generated if SIPI_CIR*n*[WAIE]=1 (see SIPI Channel Interrupt Register 0 (SIPI_CIR0), SIPI Channel Interrupt Register 1 (SIPI_CIR1), SIPI Channel Interrupt Register 2 (SIPI_CIR2), and SIPI Channel Interrupt Register 3 (SIPI_CIR3)).

50.12 CRC calculation

Example: If header is AABBh, address is 1122_3344h and data is CCDD_EEFFh. Then CRC calculation will take place as follows:

1) The CRC seed is initialized by FFFF_FFFFh.

2) All the data will be mirrored before sending to CRC engine (for example, MSB will be sent as LSB).

3) So the header will be sent as DD55_0000h.

4) Address will be sent as 22CC_4488h.

5) Data will be sent as FF77_BB33h.

50.13 Interrupt logic

A description of the interrupt logic can be found in the following figure.

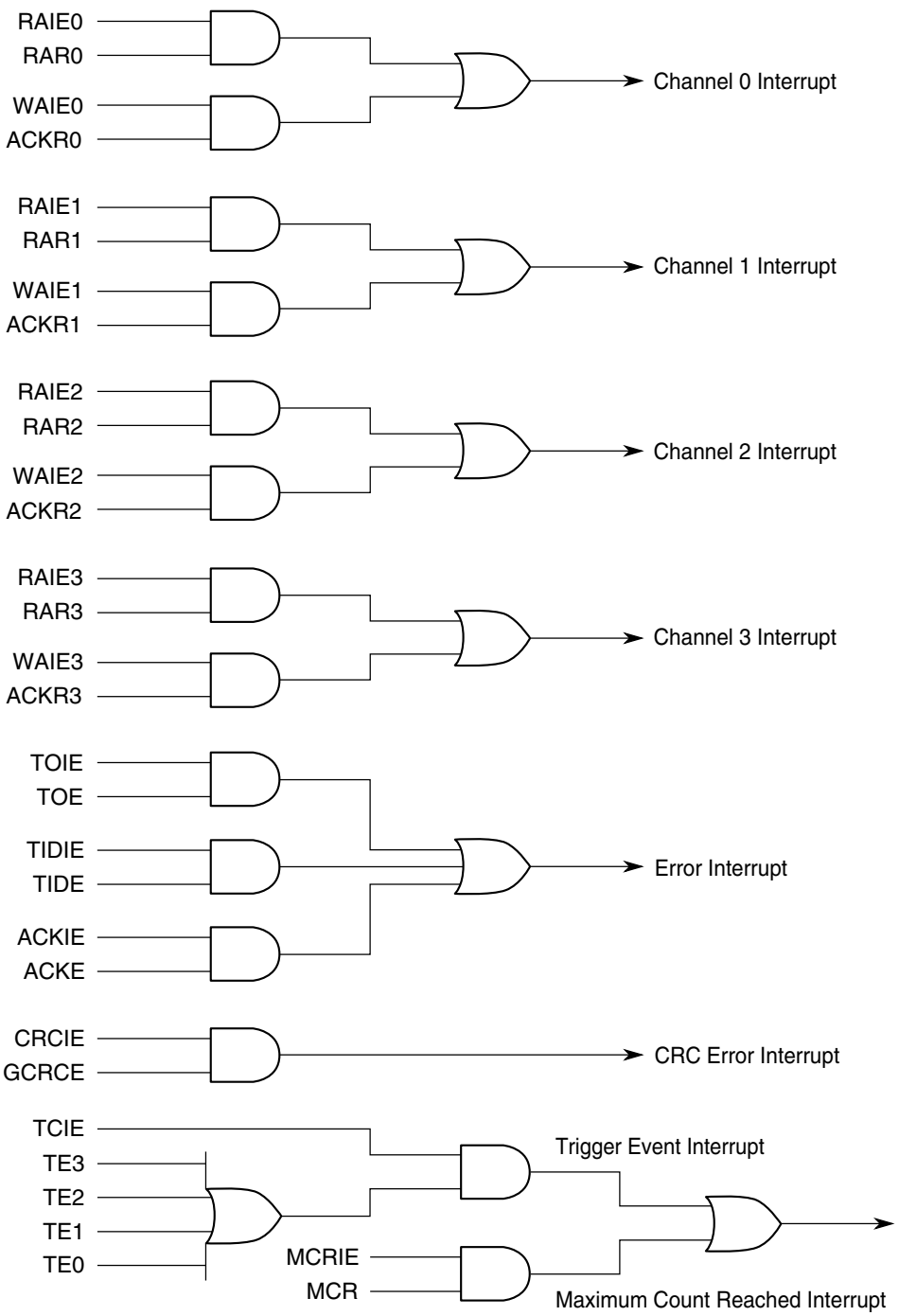


Figure 50-29. Interrupt description

50.14 SIPI control and status overview

The diagram below shows the relationship between transfers and the SIPI control and status registers.

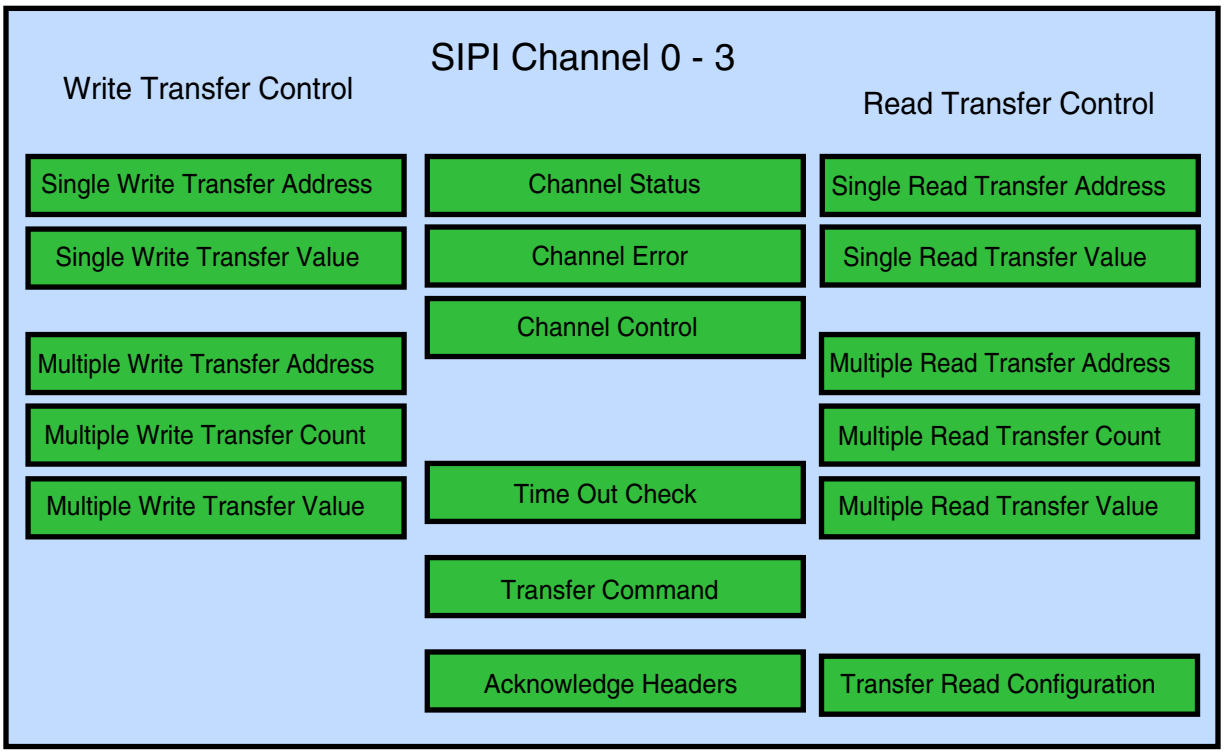


Figure 50-30. SIPI control and status overview – Register transfer

50.15 Memory map and register definition

SIPI memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	SIPI Channel Control Register 0 (SIPI_CCR0)	32	R/W	0000_0000h	50.15.1/ 2091
4	SIPI Channel Status Register 0 (SIPI_CSR0)	32	R	0000_0000h	50.15.2/ 2094
C	SIPI Channel Interrupt Register 0 (SIPI_CIR0)	32	R/W	0000_0000h	50.15.3/ 2095
10	SIPI Channel Timeout Register 0 (SIPI_CTOR0)	32	R/W	0000_00FFh	50.15.4/ 2096
14	SIPI Channel CRC Register 0 (SIPI_CCRC0)	32	R	0000_0000h	50.15.5/ 2097
18	SIPI Channel Address Register 0 (SIPI_CAR0)	32	R/W	0000_0000h	50.15.6/ 2097
1C	SIPI Channel Data Register 0 (SIPI_CDR0)	32	R/W	0000_0000h	50.15.7/ 2098
20	SIPI Channel Control Register 1 (SIPI_CCR1)	32	R/W	0000_0000h	50.15.8/ 2098

Table continues on the next page...

SIPI memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
24	SIPI Channel Status Register 1 (SIPI_CSR1)	32	R	0000_0000h	50.15.9/ 2101
2C	SIPI Channel Interrupt Register 1 (SIPI_CIR1)	32	R/W	0000_0000h	50.15.10/ 2103
30	SIPI Channel Timeout Register 1 (SIPI_CTOR1)	32	R/W	0000_00FFh	50.15.11/ 2104
34	SIPI Channel CRC Register 1 (SIPI_CCRC1)	32	R	0000_0000h	50.15.12/ 2105
38	SIPI Channel Address Register 1 (SIPI_CAR1)	32	R/W	0000_0000h	50.15.13/ 2105
3C	SIPI Channel Data Register 1 (SIPI_CDR1)	32	R/W	0000_0000h	50.15.14/ 2106
40	SIPI Channel Control Register 2 (SIPI_CCR2)	32	R/W	0000_0000h	50.15.15/ 2106
44	SIPI Channel Status Register 2 (SIPI_CSR2)	32	R	0000_0000h	50.15.16/ 2109
4C	SIPI Channel Interrupt Register 2 (SIPI_CIR2)	32	R/W	0000_0000h	50.15.17/ 2111
50	SIPI Channel Timeout Register 2 (SIPI_CTOR2)	32	R/W	0000_00FFh	50.15.18/ 2112
54	SIPI Channel CRC Register 2 (SIPI_CCRC2)	32	R	0000_0000h	50.15.19/ 2113
58	SIPI Channel Address Register 2 (SIPI_CAR2)	32	R/W	0000_0000h	50.15.20/ 2113
5C	SIPI Channel Data Register 2 (SIPI_CDR2_0)	32	R/W	0000_0000h	50.15.21/ 2114
60	SIPI Channel Data Register 2 (SIPI_CDR2_1)	32	R/W	0000_0000h	50.15.21/ 2114
64	SIPI Channel Data Register 2 (SIPI_CDR2_2)	32	R/W	0000_0000h	50.15.21/ 2114
68	SIPI Channel Data Register 2 (SIPI_CDR2_3)	32	R/W	0000_0000h	50.15.21/ 2114
6C	SIPI Channel Data Register 2 (SIPI_CDR2_4)	32	R/W	0000_0000h	50.15.21/ 2114
70	SIPI Channel Data Register 2 (SIPI_CDR2_5)	32	R/W	0000_0000h	50.15.21/ 2114
74	SIPI Channel Data Register 2 (SIPI_CDR2_6)	32	R/W	0000_0000h	50.15.21/ 2114
78	SIPI Channel Data Register 2 (SIPI_CDR2_7)	32	R/W	0000_0000h	50.15.21/ 2114
7C	SIPI Channel Control Register 3 (SIPI_CCR3)	32	R/W	0000_0000h	50.15.22/ 2114
80	SIPI Channel Status Register 3 (SIPI_CSR3)	32	R	0000_0000h	50.15.23/ 2118

Table continues on the next page...

SIPI memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
88	SIPI Channel Interrupt Register 3 (SIPI_CIR3)	32	R/W	0000_0000h	50.15.24/ 2119
8C	SIPI Channel Timeout Register 3 (SIPI_CTOR3)	32	R/W	0000_00FFh	50.15.25/ 2120
90	SIPI Channel CRC Register 3 (SIPI_CCRC3)	32	R	0000_0000h	50.15.26/ 2121
94	SIPI Channel Address Register 3 (SIPI_CAR3)	32	R/W	0000_0000h	50.15.27/ 2121
98	SIPI Channel Data Register 3 (SIPI_CDR3)	32	R/W	0000_0000h	50.15.28/ 2122
9C	SIPI Module Configuration Register (SIPI_MCR)	32	R/W	See section	50.15.29/ 2123
A0	SIPI Status Register (SIPI_SR)	32	R	0000_0000h	50.15.30/ 2126
A4	SIPI Max Count Register (SIPI_MAXCR)	32	R/W	FFFF_FFFCh	50.15.31/ 2128
A8	SIPI Address Reload Register (SIPI_ARR)	32	R/W	0000_0000h	50.15.32/ 2128
AC	SIPI Address Count Register (SIPI_ACR)	32	R/W	0000_0000h	50.15.33/ 2129
B0	SIPI Error Register (SIPI_ERR)	32	R	0000_0000h	50.15.34/ 2130

50.15.1 SIPI Channel Control Register 0 (SIPI_CCR0)

NOTE

Back to back transactions are not supported on a single channel (for example, if the channel busy bit, SIPI_CSR0[CB] = 1, is configured for a channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, SIPI_CSR0[CB] = 0).

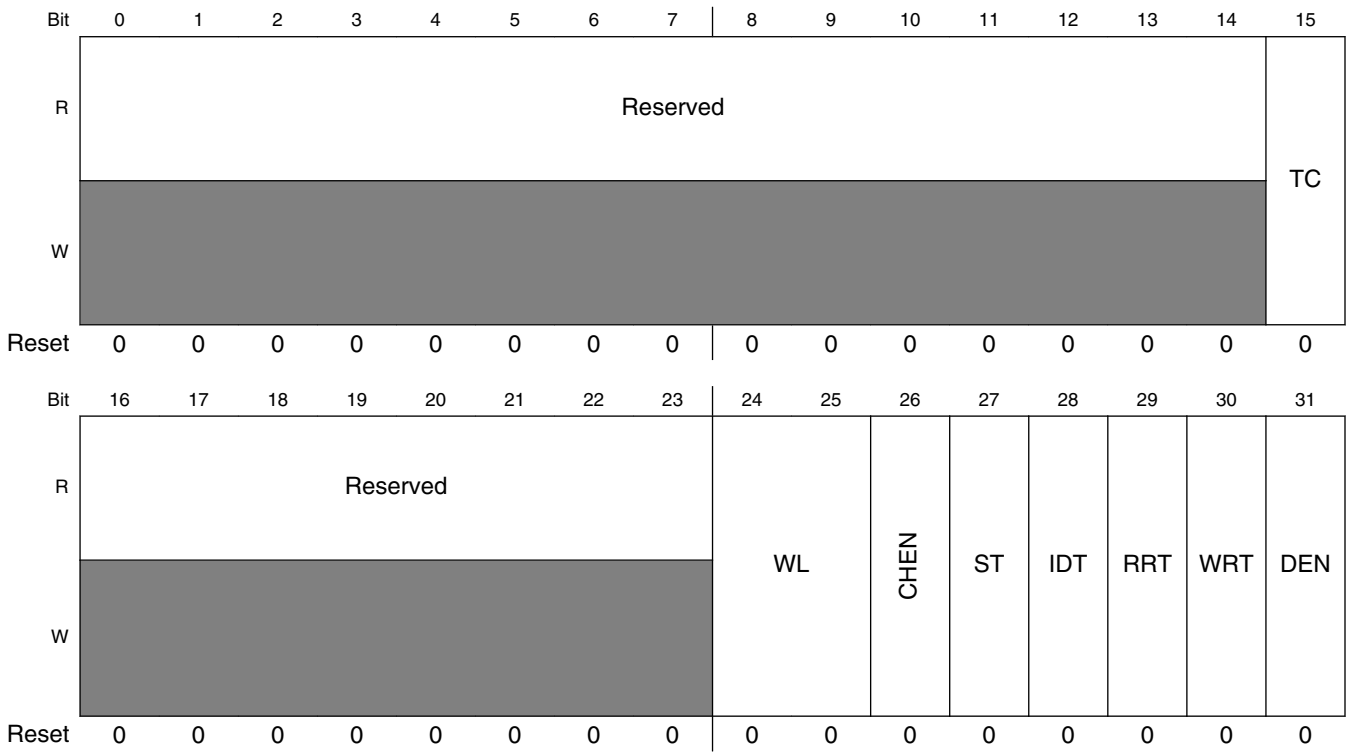
PRIORITY SCHEDULING: The channel whose SIPI_CAR*n* is written first will gain access first irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Scheduling will occur each time the LFAST is not ready to receive data and no channel is pending for data transmission. The reason for the scheduling is that a common CRC module is shared by all channels. If the LFAST is ready to receive data during the entire process, it will happen only in the beginning and later on priority will be resolved as per the priority scheme.

NOTE

This register is only writable in Initialization mode (SIPI_MCR[INIT] = 1), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted (SIPI_CSR*n*[CB] = 1 (see SIPI Channel Status Register 0 (SIPI_CSR0)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine clock cycles. The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding SIPI_CAR*n* (see SIPI Channel Address Register 0 (SIPI_CAR0)).

Address: 0h base + 0h offset = 0h



SIPI_CCR0 field descriptions

Field	Description
0–14 Reserved	This field is reserved.
15 TC	Send Trigger Command. A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in SIPI_CAR0. 0 Trigger command not sent 1 Trigger command sent
16–23 Reserved	This field is reserved.
24–25 WL	Word Length Transfer. For Streaming write, WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable. If all channel enables are simultaneously written to enable the channels, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority.

Table continues on the next page...

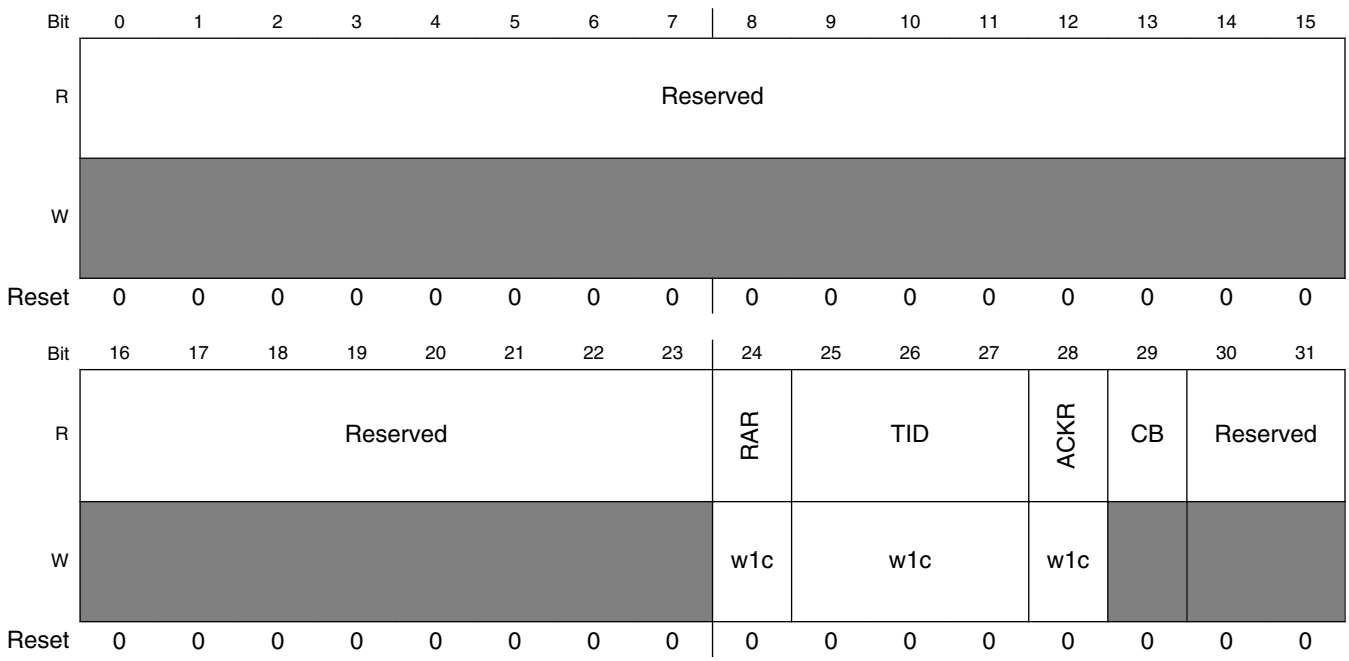
SIPI_CCR0 field descriptions (continued)

Field	Description
	0 Channel is disabled 1 Channel is enabled
27 ST	Streaming Transfer. This bit is hard-coded to 0. It can only be written for channel 2 (SIPI_CCR2). This bit can only be written when the corresponding SIPI_CCR*n*[WRT] = 1. **NOTE:** Only transfers with channel 2 can write 1 to this bit. All other channels force SIPI_CCR*n*[ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming. 0 Streaming transfer is disabled 1 Streaming transfer is enabled
28 IDT	ID Read Request Transfer. This request returns the value of the CHIP ID. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 ID read request not sent 1 ID read request sent
29 RRT	Read Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 Read request will not be sent 1 Read request transfer by the initiator. This bit can not be written if SIPI_CCR0[IDT] = 1.
30 WRT	Write Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 No write request will be sent 1 Write request transfer by the initiator. This bit can not be written if SIPI_CCR0[IDT] = 1 or SIPI_CCR0[RRT] = 1.
31 DEN	DMA Enable. When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after one major loop has completed. 0 Channel will be used for bus interface access. 1 Channel will be used for DMA access

50.15.2 SIPI Channel Status Register 0 (SIPI_CSR0)

CSR0 contains the status bits for the current transfer.

Address: 0h base + 4h offset = 4h



SIPI_CSR0 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24 RAR	Read Answer Reception. 0 Read answer not received 1 Read answer received
25–27 TID	Transaction ID of transmitted frame. Transaction ID is a random number associated with every Tx frame to match the received response/ack with the command.
28 ACKR	Acknowledge Received. 0 Acknowledge not received 1 Acknowledge received
29 CB	Channel Busy. Indicates channel 0 status. 0 Channel 0 free 1 Channel 0 busy

Table continues on the next page...

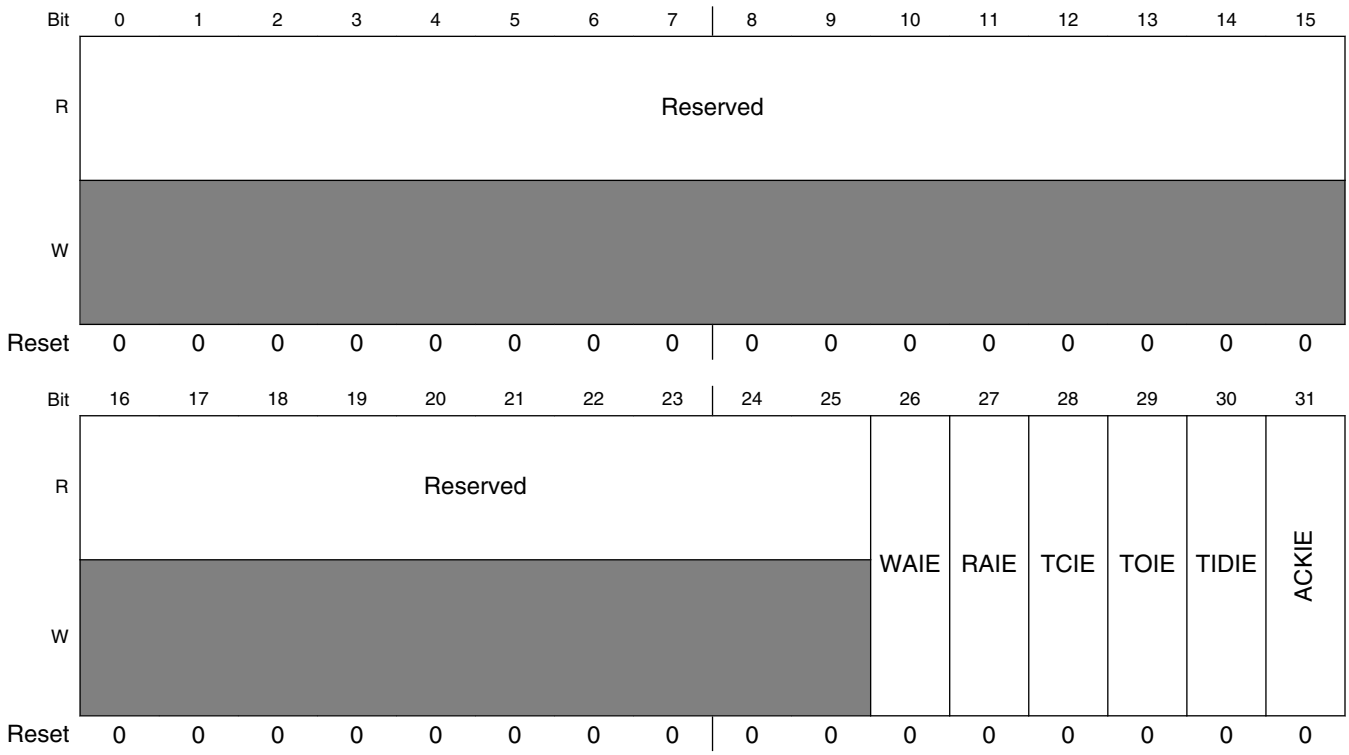
SIPI_CSR0 field descriptions (continued)

Field	Description
30–31 Reserved	This field is reserved.

50.15.3 SIPI Channel Interrupt Register 0 (SIPI_CIR0)

SIPI_CIR0 contains the interrupt enable bits for channel 0.

Address: 0h base + Ch offset = Ch



SIPI_CIR0 field descriptions

Field	Description
0–25 Reserved	This field is reserved.
26 WAIE	Write Acknowledge Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

Table continues on the next page...

SIPI_CIR0 field descriptions (continued)

Field	Description
28 TCIE	Trigger Command Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled. 0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

50.15.4 SIPI Channel Timeout Register 0 (SIPI_CTOR0)

SIPI_CTOR0 contains the timeout value for Tx requests.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																								TOR							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

SIPI_CTOR0 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24–31 TOR	Timeout value for transmitted requests. Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPI_MCR[PRSCLR]. **NOTE:** Field can only be written during Initialization mode (SIPI_MCR[INIT] = 1).

50.15.5 SIPI Channel CRC Register 0 (SIPI_CCRC0)

SIPI_CCRC*n* is a read only which returns the CRC calculated value on the header, address and data bits.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRCI																CRCT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCRC0 field descriptions

Field	Description
0–15 CRCI	Reflects received CRC value at initiator
16–31 CRCT	Reflects received CRC value at target

50.15.6 SIPI Channel Address Register 0 (SIPI_CAR0)

SIPI_CAR0 is the address target for data transmission.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CAR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CAR0 field descriptions

Field	Description
0–31 CAR	These bits contain the address of the target node.

50.15.7 SIPI Channel Data Register 0 (SIPI_CDR0)

SIPI_CDR0 contains the data to be transmitted, and can be written anytime by software or the DMA. The data can be written in 8, 16 or 32 bit formats.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CDR0 field descriptions

Field	Description
0–31 CDR	Data register bits. Contains the data that will be transmitted, or received.

50.15.8 SIPI Channel Control Register 1 (SIPI_CCR1)

NOTE

Back to back transactions are not supported on a single channel (for example, if the channel busy bit, SIPI_CSR1[CB] = 1, is configured for a channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, SIPI_CSR1[CB] = 0).

PRIORITY SCHEDULING: The channel whose SIPI_CAR*n* is written first will gain access first irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Scheduling will occur each time the LFAST is not ready to receive data and no channel is pending for data transmission. The reason for the scheduling is that a common CRC module is shared by all channels. If the LFAST is ready to receive data during the entire process, it will happen only in the beginning and later on priority will be resolved as per the priority scheme.

NOTE

This register is only writable in Initialization mode (SIPI_MCR[INIT] = 1), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted (SIPI_CSR*n*[CB] = 1 (see SIPI Channel Status Register 1 (SIPI_CSR1)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine clock cycles. The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding SIPI_CAR*n* (see SIPI Channel Address Register 1 (SIPI_CAR1)).

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															TC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								WL		CHEN	ST	IDT	RRT	WRT	DEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCR1 field descriptions

Field	Description
0–14 Reserved	This field is reserved.
15 TC	Send Trigger Command. A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in SIPI_CAR1. 0 Trigger command not sent 1 Trigger command sent
16–23 Reserved	This field is reserved.
24–25 WL	Word Length Transfer. For Streaming write WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable. If all channel enables are simultaneously written to enable the channels, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority. 0 Channel is disabled 1 Channel is enabled
27 ST	This bit is hard-coded to 0. It can only be written for channel 2 (SIPI_CCR2). This bit can only be written when the corresponding SIPI_CCR*n*[WRT] = 1. **NOTE:** Only transfers with channel 2 can write 1 to this bit. All other channels force SIPI_CCR*n*[ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming. 0 Streaming transfer is disabled 1 Streaming transfer is enabled
28 IDT	ID Read Request Transfer. This request returns the value of the CHIP ID. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 ID read request not sent 1 ID read request sent
29 RRT	Read Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT

Table continues on the next page...

SIPI_CCR1 field descriptions (continued)

Field	Description
	0 Read request will not be sent 1 Read request transfer by the initiator. This bit can not be written if SIPI_CCR1[IDT] = 1.
30 WRT	Write Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 No write request will be sent 1 Write request transfer by the initiator. This bit can not be written if SIPI_CCR1[IDT] = 1 or SIPI_CCR1[RRT] = 1.
31 DEN	DMA Enable. When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after one major loop has completed. 0 Channel will be used for bus interface access. 1 Channel will be used for DMA access

50.15.9 SIPI Channel Status Register 1 (SIPI_CSR1)

CSR1 contains the status bits for the current transfer.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								RAR	TID			ACKR	CB	Reserved	
W									w1c	w1c			w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

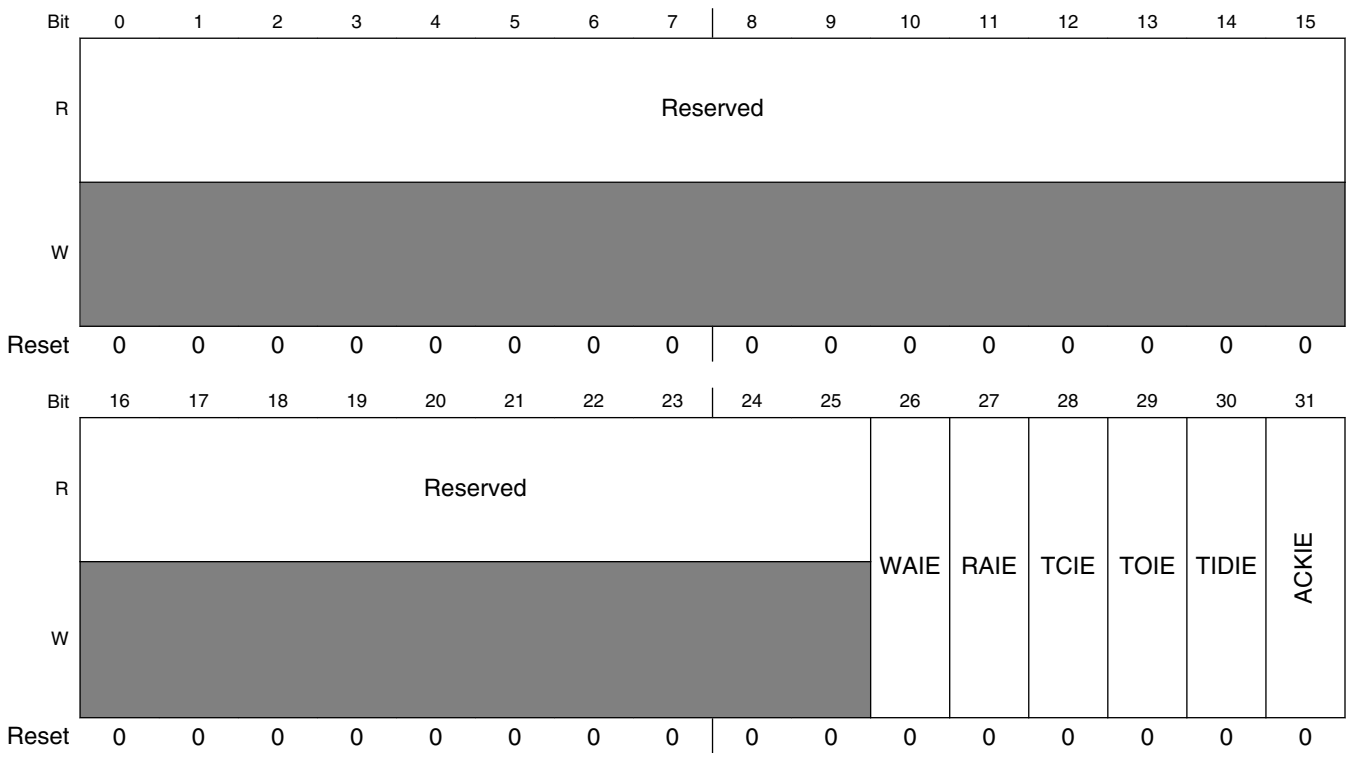
SIPI_CSR1 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24 RAR	Read Answer Reception. 0 Read answer not received 1 Read answer received
25–27 TID	Transaction ID of transmitted frame. Transaction ID is a random number associated with every Tx frame to match the received response/ack with the command.
28 ACKR	Acknowledge Received. 0 Acknowledge not received 1 Acknowledge received
29 CB	Channel Busy. Indicates channel 1 status. 0 Channel 1 free 1 Channel 1 busy
30–31 Reserved	This field is reserved.

50.15.10 SIPI Channel Interrupt Register 1 (SIPI_CIR1)

The CIR1 contains the interrupt enable bits for channel 1.

Address: 0h base + 2Ch offset = 2Ch



SIPI_CIR1 field descriptions

Field	Description
0–25 Reserved	This field is reserved.
26 WAIE	Write Acknowledge Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
28 TCIE	Trigger Command Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled.

Table continues on the next page...

SIPI_CIR1 field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

50.15.11 SIPI Channel Timeout Register 1 (SIPI_CTOR1)

SIPI_CTOR1 contains the timeout value for Tx requests.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																								TOR							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

SIPI_CTOR1 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24–31 TOR	Timeout value for transmitted requests. Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPI_MCR[PRSCLR]. **NOTE:** Field can only be written during Initialization mode (SIPI_MCR[INIT] = 1).

50.15.12 SIPI Channel CRC Register 1 (SIPI_CCRC1)

SIPI_CCRC*n* is a read only which returns the CRC calculated value on the header, address and data bits.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRCI																CRCT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCRC1 field descriptions

Field	Description
0–15 CRCI	Reflects received CRC value at initiator
16–31 CRCT	Reflects received CRC value at target

50.15.13 SIPI Channel Address Register 1 (SIPI_CAR1)

SIPI_CAR1 is the address target for data transmission.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CAR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CAR1 field descriptions

Field	Description
0–31 CAR	These bits contain the address of the target node.

50.15.14 SIPI Channel Data Register 1 (SIPI_CDR1)

SIPI_CDR1 contains the data to be transmitted, and can be written anytime by software or the DMA. The data can be written in 8, 16 or 32 bit formats.

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CDR1 field descriptions

Field	Description
0–31 CDR	Data register bits. Contains the data that will be transmitted, or received.

50.15.15 SIPI Channel Control Register 2 (SIPI_CCR2)

NOTE

Back to back transactions are not supported on a single channel (for example, if the channel busy bit, SIPI_CSR2[CB] = 1, is configured for a channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, SIPI_CSR2[CB] = 0).

PRIORITY SCHEDULING: The channel whose SIPI_CAR*n* is written first will gain access first irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Scheduling will occur each time the LFAST is not ready to receive data and no channel is pending for data transmission. The reason for the scheduling is that a common CRC module is shared by all channels. If the LFAST is ready to receive data during the entire process, it will happen only in the beginning and later on priority will be resolved as per the priority scheme.

NOTE

This register is only writable in Initialization mode (SIPI_MCR[INIT] = 1), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted (SIPI_CSR*n*[CB] = 1 (see SIPI Channel Status Register 2 (SIPI_CSR2)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine clock cycles. The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding SIPI_CAR*n* (see SIPI Channel Address Register 2 (SIPI_CAR2)).

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															TC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								WL		CHEN	ST	IDT	RRT	WRT	DEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCR2 field descriptions

Field	Description
0–14 Reserved	This field is reserved.
15 TC	Send Trigger Command. A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in SIPI_CAR2. 0 Trigger command not sent 1 Trigger command sent
16–23 Reserved	This field is reserved.
24–25 WL	Word Length Transfer. For Streaming write, WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable. If all channel enables are simultaneously written to enable the channels, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority. 0 Channel is disabled 1 Channel is enabled
27 ST	This bit is hard-coded to 0. It can only be written for channel 2 (SIPI_CCR2). This bit can only be written when the corresponding SIPI_CCR*n*[WRT] = 1. **NOTE:** Only transfers with channel 2 can write 1 to this bit. All other channels force SIPI_CCR*n*[ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming. 0 Streaming transfer is disabled 1 Streaming transfer is enabled
28 IDT	ID Read Request Transfer. This request returns the value of the CHIP ID. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 ID read request not sent 1 ID read request sent
29 RRT	Read Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT

Table continues on the next page...

SIPI_CCR2 field descriptions (continued)

Field	Description
	0 Read request will not be sent 1 Read request transfer by the initiator. This bit can not be written if SIPI_CCR2[IDT] = 1.
30 WRT	Write Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 No write request will be sent 1 Write request transfer by the initiator. This bit can not be written if SIPI_CCR2[IDT] = 1 or SIPI_CCR2[RRT] = 1.
31 DEN	DMA Enable. When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after one major loop has completed. 0 Channel will be used for bus interface access. 1 Channel will be used for DMA access

50.15.16 SIPI Channel Status Register 2 (SIPI_CSR2)

SIPI_CSR2 contains the status bits for the current transfer.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								RAR	TID			ACKR	CB	Reserved	
W									w1c	w1c			w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

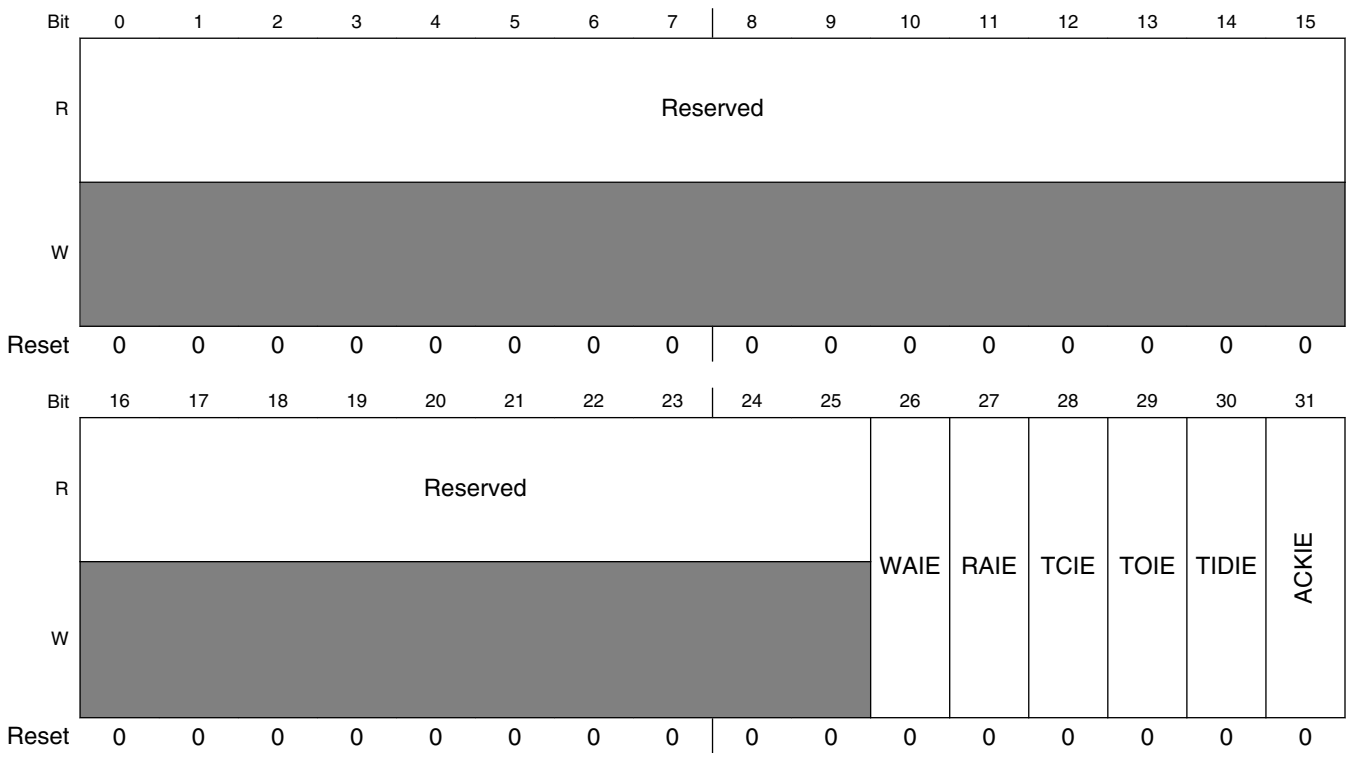
SIPI_CSR2 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24 RAR	Read Answer Reception. 0 Read answer not received 1 Read answer received
25–27 TID	Transaction ID of transmitted frame. Transaction ID is a random number associated with every Tx frame to match the received response/ack with the command.
28 ACKR	Acknowledge Received. 0 Acknowledge not received 1 Acknowledge received
29 CB	Channel Busy. Indicates channel 2 status. 0 Channel 2 free 1 Channel 2 busy
30–31 Reserved	This field is reserved.

50.15.17 SIPI Channel Interrupt Register 2 (SIPI_CIR2)

The SIPI_CIR2 contains the interrupt enable bits for channel 2.

Address: 0h base + 4Ch offset = 4Ch



SIPI_CIR2 field descriptions

Field	Description
0–25 Reserved	This field is reserved.
26 WAIE	Write Acknowledge Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
28 TCIE	Trigger Command Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled.

Table continues on the next page...

SIPI_CIR2 field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

50.15.18 SIPI Channel Timeout Register 2 (SIPI_CTOR2)

SIPI_CTOR2 contains the timeout value for Tx requests.

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																								TOR							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

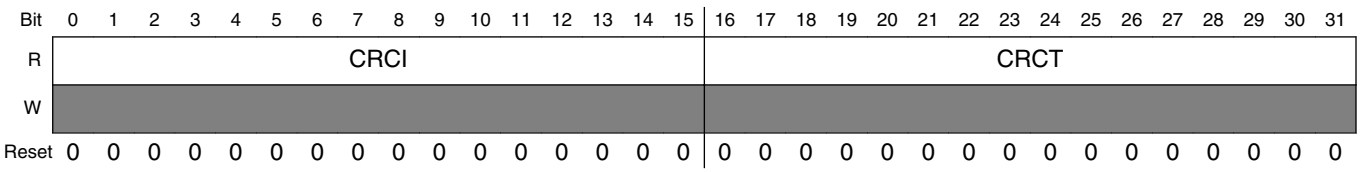
SIPI_CTOR2 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24–31 TOR	Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPI_MCR[PRSCLR]. **NOTE:** Field can only be written during Initialization mode (SIPI_MCR[INIT] = 1).

50.15.19 SIPI Channel CRC Register 2 (SIPI_CCRC2)

SIPI_CCRC*n* is a read only which returns the CRC calculated value on the header, address and data bits.

Address: 0h base + 54h offset = 54h



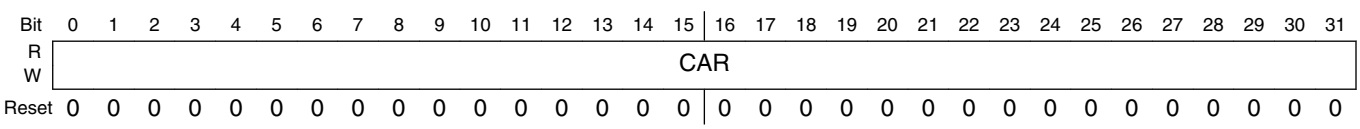
SIPI_CCRC2 field descriptions

Field	Description
0–15 CRCI	Reflects received CRC value at initiator
16–31 CRCT	Reflects received CRC value at target

50.15.20 SIPI Channel Address Register 2 (SIPI_CAR2)

SIPI_CAR2 is the address target for data transmission. For streaming operations this register is the start address.

Address: 0h base + 58h offset = 58h



SIPI_CAR2 field descriptions

Field	Description
0–31 CAR	These bits contain the address of the target node.

50.15.21 SIPI Channel Data Register 2 (SIPI_CDR2_*n*)

SIPI_CDR2_*n*contains the data to be transmitted, and can be written anytime by software or the DMA. The data can be written in 8, 16 or 32 bit formats.

Address: 0h base + 5Ch offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CDR2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CDR2_*n* field descriptions

Field	Description
0–31 CDR2	Data register bits. Contains the data that will be transmitted, or received.

50.15.22 SIPI Channel Control Register 3 (SIPI_CCR3)

NOTE

Back to back transactions are not supported on a single channel (for example, if the channel busy bit, SIPI_CSR3[CB] = 1, is configured for a channel, it cannot be triggered for another transfer until the data buffers are empty and the channel is free, SIPI_CSR3[CB] = 0).

PRIORITY SCHEDULING: The channel whose SIPI_CAR*n* is written first will gain access first irrespective of its priority at the start of the transfer. As more transfers are requested, priority will be resolved as per the priority scheme (for example, channel 0 – channel 1 – channel 2 – channel 3). Scheduling will occur each time the LFAST is not ready to receive data and no channel is pending for data transmission. The reason for the scheduling is that a common CRC module is shared by all channels. If the LFAST is ready to receive data during the entire process, it will happen only in the beginning and later on priority will be resolved as per the priority scheme.

NOTE

This register is only writable in Initialization mode (SIPI_MCR[INIT] = 1), except for command bits (for example, TC, WL, ST, IDT, RRT and WRT). Also, the command bits can be written only when the corresponding Channel busy bit is not asserted (SIPI_CSR*n*[CB] = 1 (see SIPI Channel Status Register 3 (SIPI_CSR3)). Software will need to poll the respective channel busy bits before attempting to change the command type.

Only one channel at a time can transmit. Therefore, only one channel can remain busy. Maximum time allowed to transmit a full command is nine clock cycles. The Command type of a channel can only be changed when it is no longer busy. Also, to send a command, we need to write to the corresponding SIPI_CAR*n* (see SIPI Channel Address Register 3 (SIPI_CAR3)).

Address: 0h base + 7Ch offset = 7Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															TC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								WL		CHEN	ST	IDT	RRT	WRT	DEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCR3 field descriptions

Field	Description
0–14 Reserved	This field is reserved.
15 TC	Send Trigger Command. A Trigger Command requires that only this bit to be set by software, the trigger does not depend on settings in SIPI_CAR3. 0 Trigger command not sent 1 Trigger command sent
16–23 Reserved	This field is reserved.
24–25 WL	Word Length Transfer. For Streaming write WL bits should be written 10. 00 8-bit 01 16-bit 10 32-bit 11 not used
26 CHEN	Channel Enable. If all channel enables are simultaneously written to enable the channels, channel 0 will be given the highest priority for transmission. For channels, the lowest channel number is given highest priority. 0 Channel is disabled 1 Channel is enabled
27 ST	Streaming Transfer. This bit is hard-coded to 0. It can only be written for channel 2 (SIPI_CCR2). This bit can only be written when the corresponding SIPI_CCR*n*[WRT] = 1. **NOTE:** Only transfers with channel 2 can write 1 to this bit. All other channels force SIPI_CCR*n*[ST] = 0. The SIPI_CCR2[WRT] and SIPI_CCR2[ST] bits must be set for streaming. 0 Streaming transfer is disabled 1 Streaming transfer is enabled
28 IDT	ID Read Request Transfer. This request returns the value of the CHIP ID. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 ID read request not sent 1 ID read request sent
29 RRT	Read Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT

Table continues on the next page...

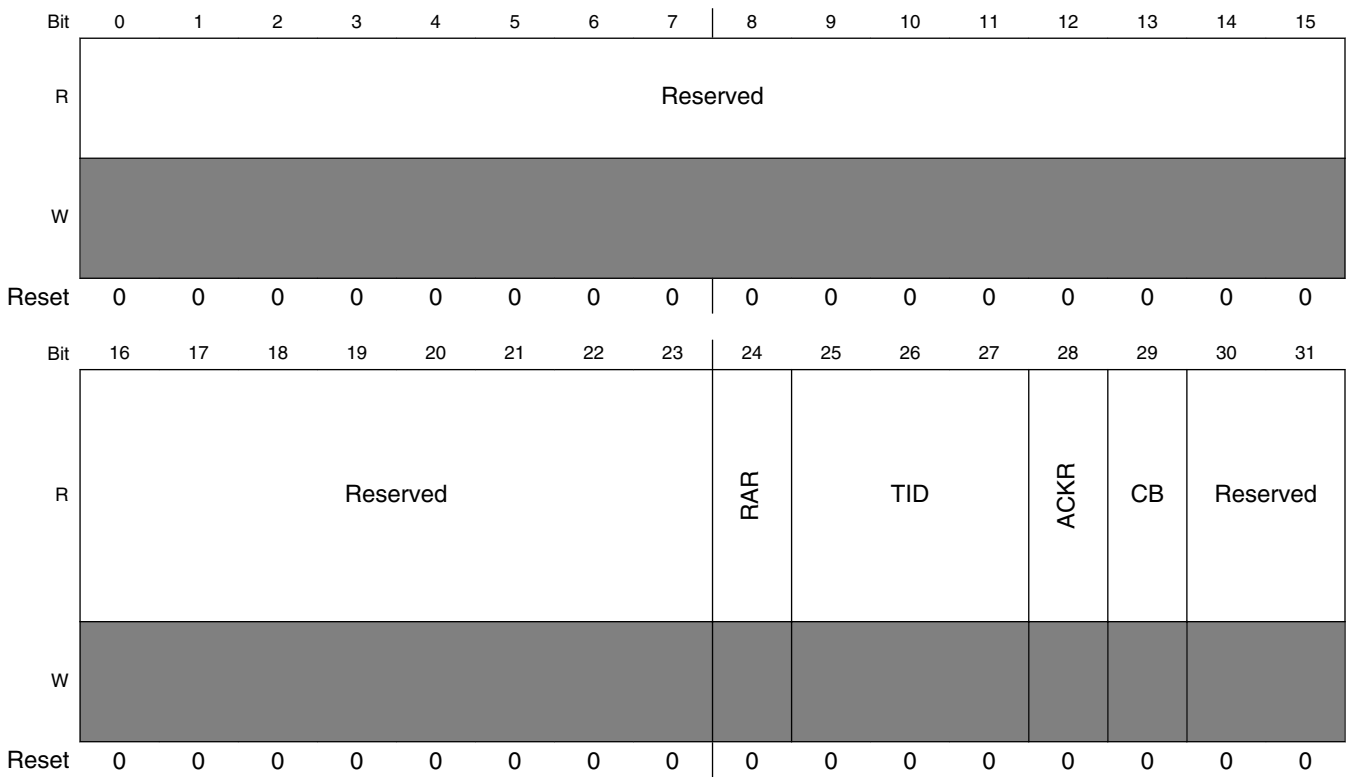
SIPI_CCR3 field descriptions (continued)

Field	Description
	3. WRT 0 Read request will not be sent 1 Read request transfer by the initiator. This bit can not be written if SIPI_CCR3[IDT] = 1.
30 WRT	Write Request Transfer. **NOTE:** The order of priority for writing to IDT, WRT and RRT must be in this order: 1. IDT 2. RRT 3. WRT 0 No write request will be sent 1 Write request transfer by the initiator. This bit can not be written if SIPI_CCR3[IDT] = 1 or SIPI_CCR3[RRT] = 1.
31 DEN	DMA Enable. When enabling DMA mode, this bit should be set by software. It will then be cleared by hardware after one major loop has completed. 0 Channel will be used for bus interface access. 1 Channel will be used for DMA access

50.15.23 SIPI Channel Status Register 3 (SIPI_CSR3)

SIPI_CSR3 contains the status bits for the current transfer.

Address: 0h base + 80h offset = 80h



SIPI_CSR3 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24 RAR	Read Answer Reception. 0 Read answer not received 1 Read answer received
25–27 TID	Transaction ID of transmitted frame. Transaction ID is a random number associated with every Tx frame to match the received response/ack with the command.
28 ACKR	Acknowledge Received. 0 Acknowledge not received 1 Acknowledge received

Table continues on the next page...

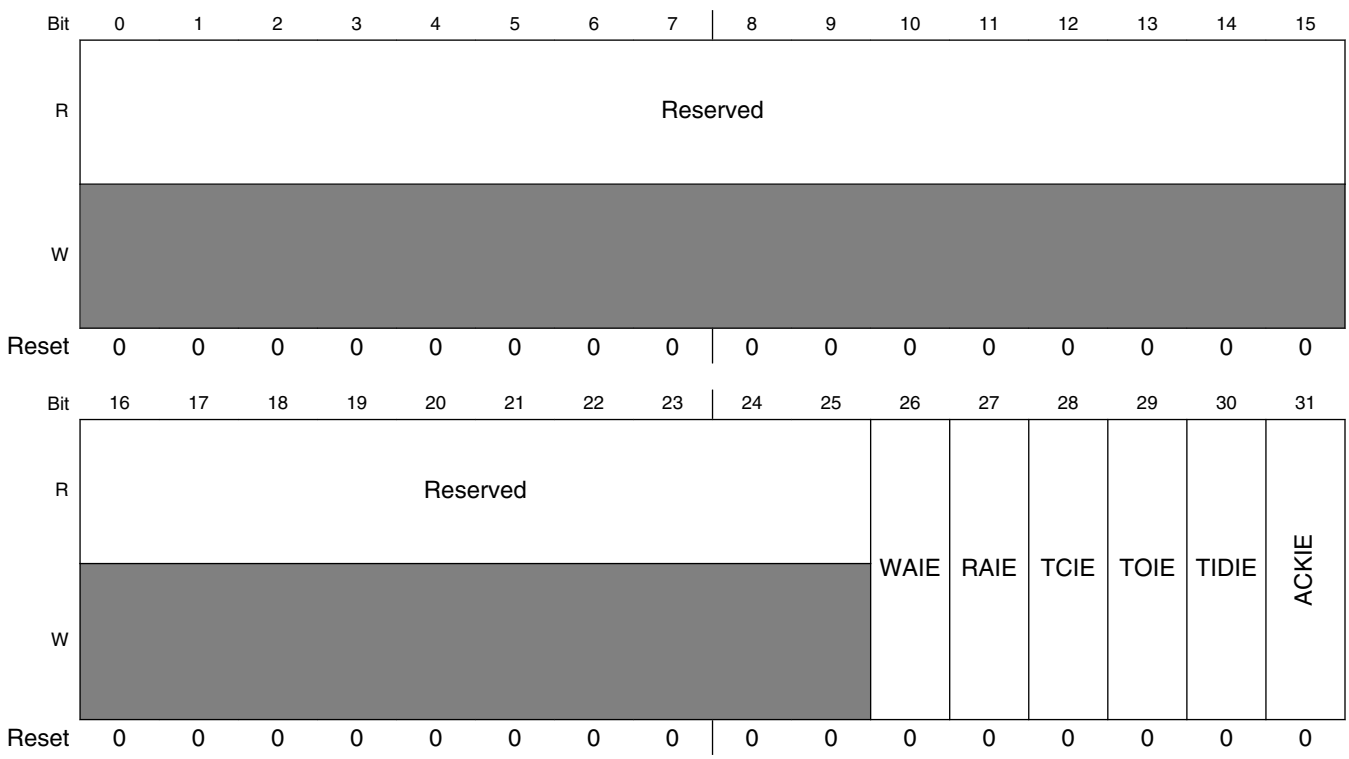
SIPI_CSR3 field descriptions (continued)

Field	Description
29 CB	Channel Busy. Indicates channel 3 status. 0 Channel 3 free 1 Channel 3 busy
30–31 Reserved	This field is reserved.

50.15.24 SIPI Channel Interrupt Register 3 (SIPI_CIR3)

SIPI_CIR3 contains the interrupt enable bits for channel 3.

Address: 0h base + 88h offset = 88h



SIPI_CIR3 field descriptions

Field	Description
0–25 Reserved	This field is reserved.
26 WAIE	Write Acknowledge Interrupt Enable.

Table continues on the next page...

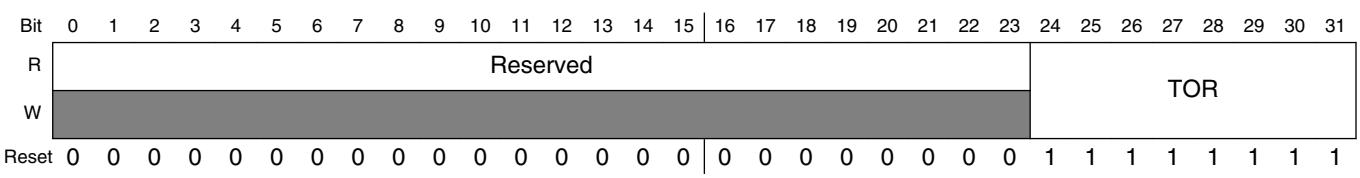
SIPI_CIR3 field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
27 RAIE	Read Answer Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
28 TCIE	Trigger Command Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
29 TOIE	Timeout Error Interrupt Enabled. 0 Interrupt is disabled 1 Interrupt is enabled
30 TIDIE	Transaction ID Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
31 ACKIE	Acknowledge Error Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled

50.15.25 SIPI Channel Timeout Register 3 (SIPI_CTOR3)

SIPI_CTOR3 contains the timeout value for Tx requests.

Address: 0h base + 8Ch offset = 8Ch



SIPI_CTOR3 field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24–31 TOR	Timeout value for transmitted requests. Timeout counter runs on the prescaled peripheral clock. Prescaler is defined by SIPI_MCR[PRSCLR]. **NOTE:** Field can only be written during Initialization mode (SIPI_MCR[INIT] = 1).

50.15.26 SIPI Channel CRC Register 3 (SIPI_CCRC3)

SIPI_CCRC*n* is a read only which returns the CRC calculated value on the header, address and data bits.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRCI																CRCT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CCRC3 field descriptions

Field	Description
0–15 CRCI	Reflects received CRC value at initiator
16–31 CRCT	Reflects received CRC value at target

50.15.27 SIPI Channel Address Register 3 (SIPI_CAR3)

SIPI_CAR3 is the address target for data transmission.

Address: 0h base + 94h offset = 94h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CAR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_CAR3 field descriptions

Field	Description
0–31 CAR	These bits contain the address of the target node.

50.15.28 SIPI Channel Data Register 3 (SIPI_CDR3)

SIPI_CDR3 contains the data to be transmitted, and can be written anytime by software or the DMA. The data can be written in 8, 16 or 32 bit formats.

Address: 0h base + 98h offset = 98h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

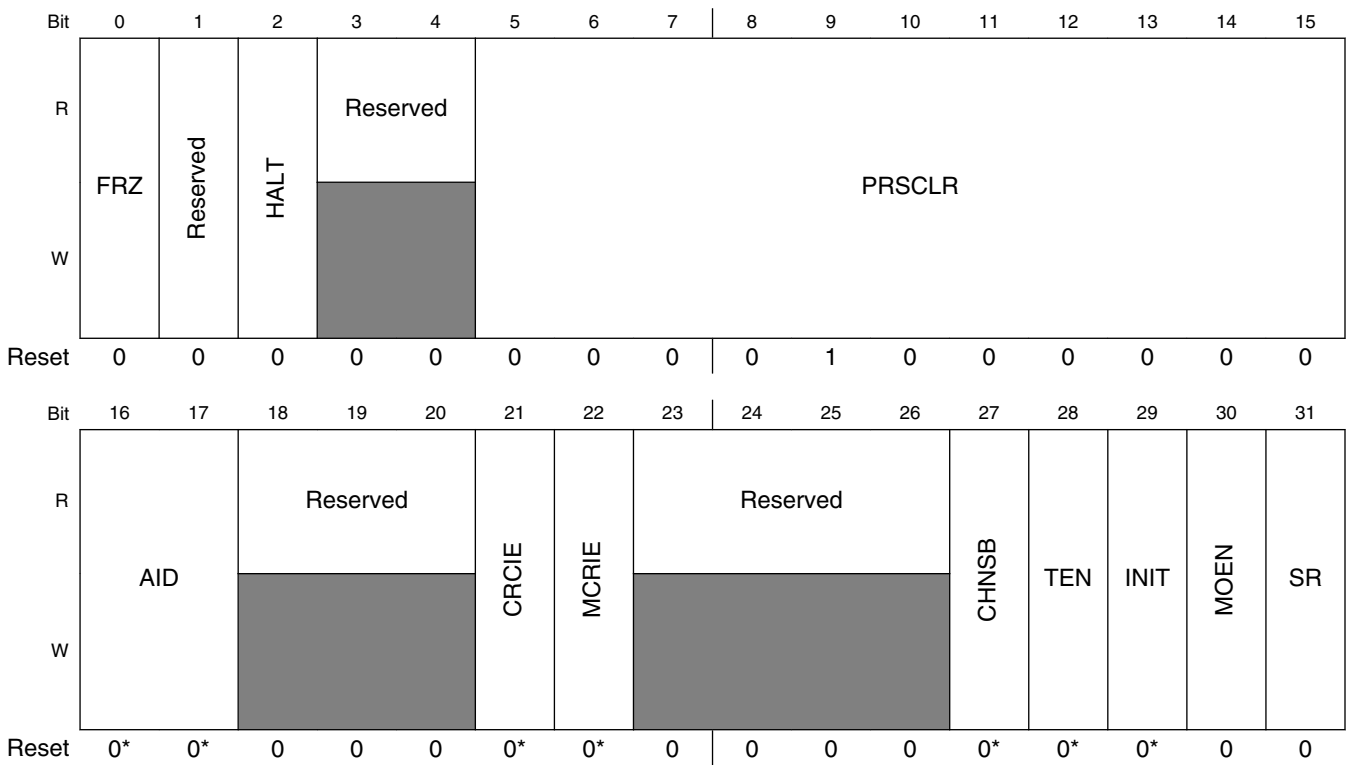
SIPI_CDR3 field descriptions

Field	Description
0–31 CDR	Data register bits. Contains the data that will be transmitted, or received.

50.15.29 SIPI Module Configuration Register (SIPI_MCR)

The SIPI_MCR is a global 32-bit configuration register.

Address: 0h base + 9Ch offset = 9Ch



* Notes:
- INIT field: Can only be written when SIPI_MCR[MOEN] = 1.
- TEN field: Can only be written when SIPI_MCR[MOEN] = 1.
- CHNSB field: Can be written only once after reset.
- MCRIE field: Can only be written when SIPI_MCR[MOEN] = 1.
- CRCIE field: Can only be written when SIPI_MCR[MOEN] = 1.
- AID field: Can be written in initialization mode only (SIPI_MCR[INIT] = 1).

SIPI_MCR field descriptions

Field	Description
0 FRZ	Freeze Enable The FRZ bit specifies the SIPI behavior when Debug mode is requested at the MCU level. When FRZ is asserted, the SIPI is enabled to enter Freeze mode. Negation of this bit field causes SIPI to exit Freeze mode. **NOTE:** Can only be written when SIPI_MCR[MOEN] = 1.

Table continues on the next page...

SIPI_MCR field descriptions (continued)

Field	Description
	0 Not enabled to enter Freeze mode 1 Enabled to enter Freeze mode
1 Reserved	This field is reserved. **Important:** Always write the default value to this field.
2 HALT	Halt Mode Enable Assertion of this bit puts SIPI into Freeze mode. No Rx or Tx is performed in the SIPI until this bit is cleared. If this bit is enabled in during Tx or Rx communications, the current activity will finish, then the SIPI will enter Freeze mode. **NOTE:** Can only be written when SIPI_MCR[MOEN] = 1. 0 No Freeze mode request 1 Enters Freeze mode if FRZ bit is asserted.
3–4 Reserved	This field is reserved.
5–15 PRSCLR	Timeout counter prescaler The timeout counter runs on the prescaled system clock. Default value is 64 (040h). The allowed programmable values are (all other values are ignored): 040h 64 (default) 080h 128 100h 256 200h 512 400h 1024 **NOTE:** This field should be programmed by software during initialization mode, SIPI_MCR[INIT] = 1. Writes during other times are ignored. **NOTE:** Writes to SIPI_MCR[PRSCLR] can only be accomplished with 16-bit or 32-bit writes.
16–17 AID	Address Increment/Decrement These bits define the type of address change at the target node. **NOTE:** This bits should be programmed by software in initialization mode, SIPI_MCR[INIT] = 1. Writes during other times are ignored. 00 no change. address stays same 01 address increments by 4 10 address decrements by 4 11 not used
18–20 Reserved	This field is reserved.
21 CRCIE	CRC Error Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
22 MCRIE	Max Count Reached Interrupt Enable

Table continues on the next page...

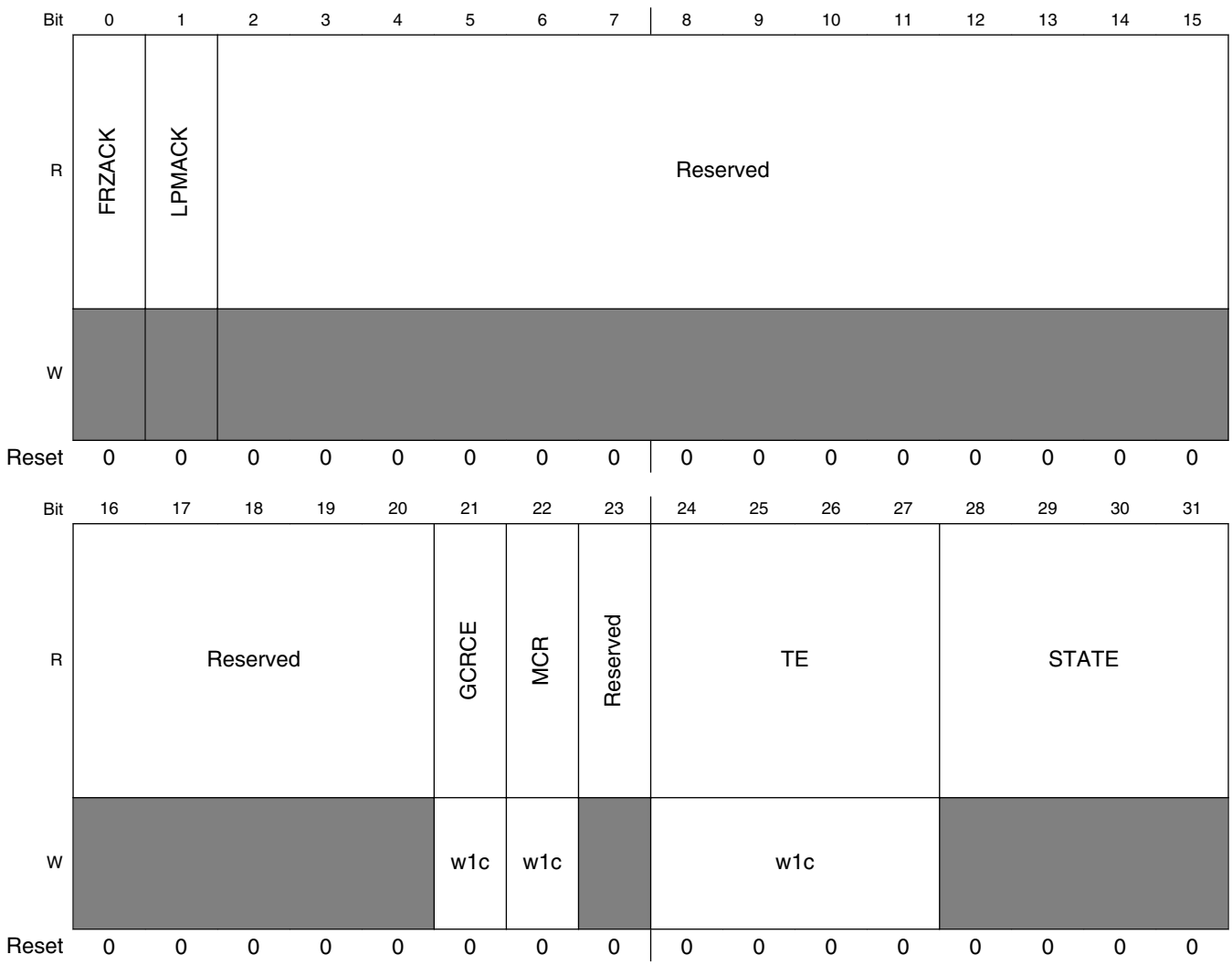
SIPI_MCR field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
23–26 Reserved	This field is reserved.
27 CHNSB	Channel coding select bit. 0 Code Table II (see Table 50-2) 1 Code Table I (see Table 50-2)
28 TEN	Target Enable Setting this bit enables the target functionality at SIPI. This bit can be read or written anytime. This bit is automatically negated by hardware when target detects an error in "streaming without ACK" mode. This bit has to be enabled for the transmission operations also.
29 INIT	Initialization Mode Setting this bit puts the module in initialization mode. This bit should be cleared by software. Most register bits are configured using this bit. The SIPI_MCR[MOEN] bit needs to be set first, and then the INIT bit can be set and both bits can't be enabled together. 0 Normal Mode 1 Initialization Mode
30 MOEN	Module Enable This bit should be set or cleared by software. When this bit is negated, all SIPI operations are immediately stopped.
31 SR	Soft Reset Setting this bit clears all status and error registers, and FSMs are moved to idle state. This bit is automatically cleared by hardware once the reset operation is complete.

50.15.30 SIPI Status Register (SIPI_SR)

The SIPI_SR is the global status register of SIPI.

Address: 0h base + A0h offset = A0h



SIPI_SR field descriptions

Field	Description
0 FRZACK	Freeze Mode Acknowledge. This read-only bit indicates that SIPI is in Freeze mode. The Freeze mode request cannot be granted until current transmission or reception processes have finished. The software can poll the FRZACK bit to find when the SIPI has actually entered Freeze mode. If Freeze mode is requested while SIPI is in any of the low power modes, then the FRZACK bit will only be set when the low power mode is exited.

Table continues on the next page...

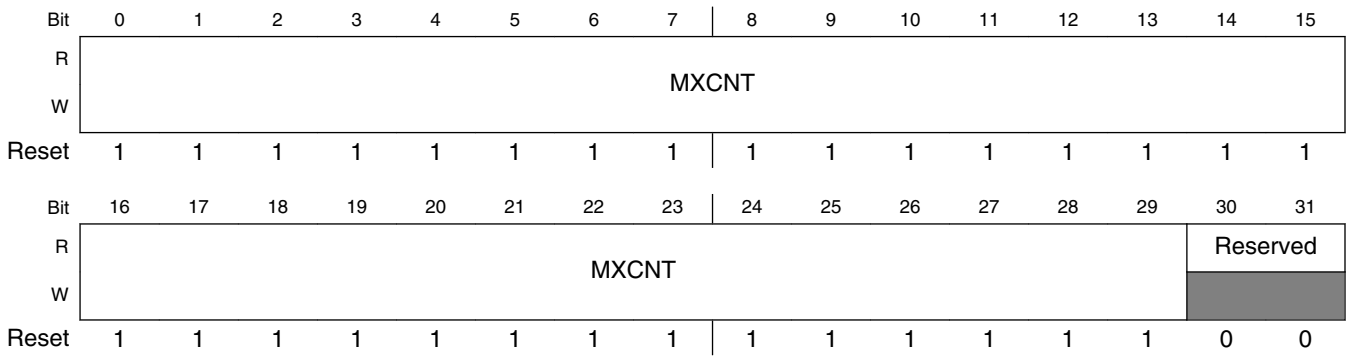
SIPI_SR field descriptions (continued)

Field	Description
	0 SIPI not in Freeze mode 1 SIPI in Freeze mode
1 LPMACK	Low Power Mode Acknowledge. This read-only bit indicates that SIPI is in Disable Mode. Disable mode can not be entered until all current transmission or reception processes have finished. The CPU can poll the LPMACK bit to know when SIPI has actually entered low power mode. 0 SIPI is not in low power mode 1 SIPI is in Disable Mode.
2–20 Reserved	This field is reserved.
21 GCRCE	Global CRC Error Bit. 0 No CRC error 1 CRC error occurred
22 MCR	Maximum Count Reached. This bit will be set whenever SIPI_ACR[ADCNT] = SIPI_MAXCR[MXCNT]. An interrupt will be generated when this bit is set only if SIPI_MCR[MCRIE] = 1. This it should be cleared by software.
23 Reserved	This field is reserved.
24–27 TE	Trigger Event on Respective Channels. This field enables interrupts for target nodes. xxx1 TE0 = 1 - Channel 0 trigger event xx1x TE1 = 1 - Channel 1 trigger event x1xx TE2 = 1 - Channel 2 trigger event 1xxx TE3 = 1 - Channel 3 trigger event
28–31 STATE	These bits reflect the transmit state machine status. They can be polled for determination of state machine status. 0000 IDLE 0001 HEADER_AND_ADDRESS_FIELD 0010 HEADER_AND_DATA_FIELD 0011 HEADER_AND_CRC_FIELD 0100 ADDRESS_AND_CRC_FIELD 0101 ADDRESS_AND_DATA_FIELD 0110 DATA_AND_CRC_FIELD 0111 DATA_FIELD

50.15.31 SIPI Max Count Register (SIPI_MAXCR)

SIPI_MAXCR contains the maximum address count value at target node. It is programmed via direct write request through the initiator. This register can be read or written by software anytime.

Address: 0h base + A4h offset = A4h



SIPI_MAXCR field descriptions

Field	Description
0–29 MXCNT	This field contains the maximum address count value at the target node. It should be programmed via direct write request through the initiator.
30–31 Reserved	This field is reserved.

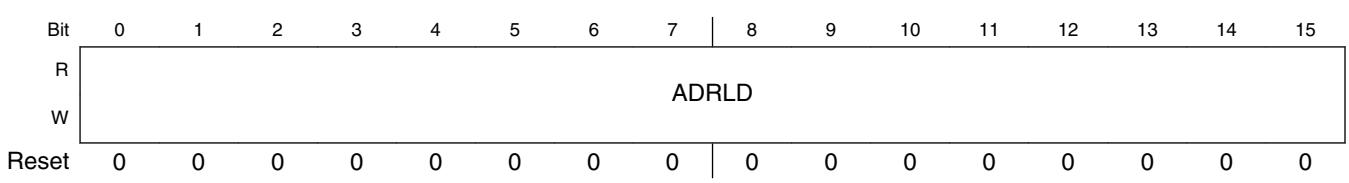
50.15.32 SIPI Address Reload Register (SIPI_ARR)

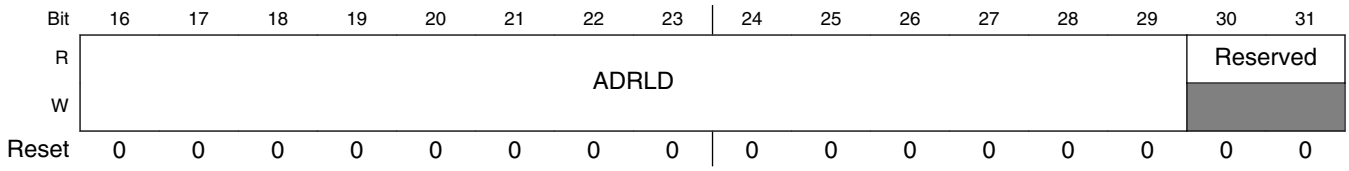
The SIPI_ARR contains the reload value for the address counter at the target node. It should be configured by direct write request from the initiator.

NOTE

This register is writeable only when SIPI_MCR[INIT] = 1.

Address: 0h base + A8h offset = A8h





SIPI_ARR field descriptions

Field	Description
0–29 ADRLD	ADRLD contains the reload value for the address counter at the target node. It should be configured by direct write request from the initiator.
30–31 Reserved	This field is reserved.

50.15.33 SIPI Address Count Register (SIPI_ACR)

This register reflects the count value of address counter at target node. It should be configured by direct write request from initiator. This register can be read/written by software anytime.

NOTE

SIPI_ARR, SIPI_ACR and SIPI_MAXCR will be configured by direct write operation through the initiator.

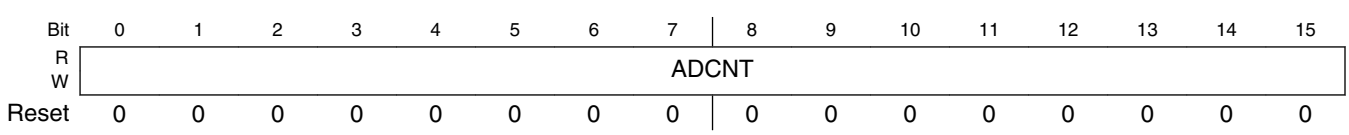
NOTE

When a streaming write command is received, the target will start the write operation from the address present in SIPI_ACR (address count register).

NOTE

After each 32-bit write has completed, SIPI_ACR[ADCNT] is compared against SIPI_MAXCR[MXCNT]. If they are equal, SIPI_ACR[ADCNT] is loaded with the value stored in SIPI_ARR[ADRLD]. If they are not equal, the SIPI_ACR[ADCNT] will increment by 4, decrement by 4 or remain the same (depending on configuration). In both cases, the SIPI_ACR[ADCNT] will contain the address at which the next streaming write will occur.

Address: 0h base + ACh offset = ACh



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ADCNT														Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_ACR field descriptions

Field	Description
0–29 ADCNT	Feflects the count value of address counter at target node. It should be configured by direct write request from the initiator. This register can be read/written by software anytime. At the end of the streaming operation, SIPI_ACR will point to the next address to be written.
30–31 Reserved	This field is reserved.

50.15.34 SIPI Error Register (SIPI_ERR)

This register contains the error bits for the last transaction(s) for all the channels. Communication errors are generated only for out of range address accesses, areas that are read only and where accesses do not lead to generation of transfer errors. Also, writing to read only registers will not generate errors.

Address: 0h base + B0h offset = B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved					TOE3	TIDE3	ACKE3	Reserved					TOE2	TIDE2	ACKE2
W						w1c	w1c	w1c						w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved					TOE1	TIDE1	ACKE1	Reserved					TOE0	TIDE0	ACKE0
W						w1c	w1c	w1c						w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIPI_ERR field descriptions

Field	Description
0–4 Reserved	This field is reserved.
5 TOE3	Timeout Error for Channel 3. 0 Timeout error occured 1 Timeout error didn't occur
6 TIDE3	Transaction ID Error for Channel 3. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
7 ACKE3	Acknowledge Error for Channel 3. 0 Acknowledge received is correct. 1 Acknowledge received is not correct.
8–12 Reserved	This field is reserved.
13 TOE2	Timeout Error for Channel 2. 0 Timeout error occured 1 Timeout error didn't occur
14 TIDE2	Transaction ID Error for Channel 2. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
15 ACKE2	Acknowledge Error for Channel 2. 0 Acknowledge received is correct 1 Acknowledge received is not correct
16–20 Reserved	This field is reserved.
21 TOE1	Timeout Error for Channel 1. 0 Timeout error occured 1 Timeout error didn't occur
22 TIDE1	Transaction ID Error for Channel 1. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
23 ACKE1	Acknowledge Error for Channel 1. 0 Acknowledge received is correct 1 Acknowledge received is not correct
24–28 Reserved	This field is reserved.
29 TOE0	Timeout Error for Channel 0. 0 Timeout error occured 1 Timeout error did not occur

Table continues on the next page...

SIPI_ERR field descriptions (continued)

Field	Description
30 TIDE0	Transaction ID Error for Channel 0. 0 Received transaction ID matched with the stored ID 1 Received transaction ID didn't match with the stored ID
31 ACKE0	Acknowledge Error for Channel 0. 0 Acknowledge received is correct. 1 Acknowledge received is not correct.

Chapter 51
LVDS Fast Asynchronous Serial Transmission (LFAST) – Interprocessor Communications

51.1 Introduction

This chapter describes the specifications of the LFAST module, which implements the LVDS Fast Asynchronous Serial Transmission (LFAST) module. LFAST is used in dual mode (software configurable master/slave operation) for interprocessor communications.

51.2 Block diagram

The following figure depicts LFAST interaction with other modules on the device.

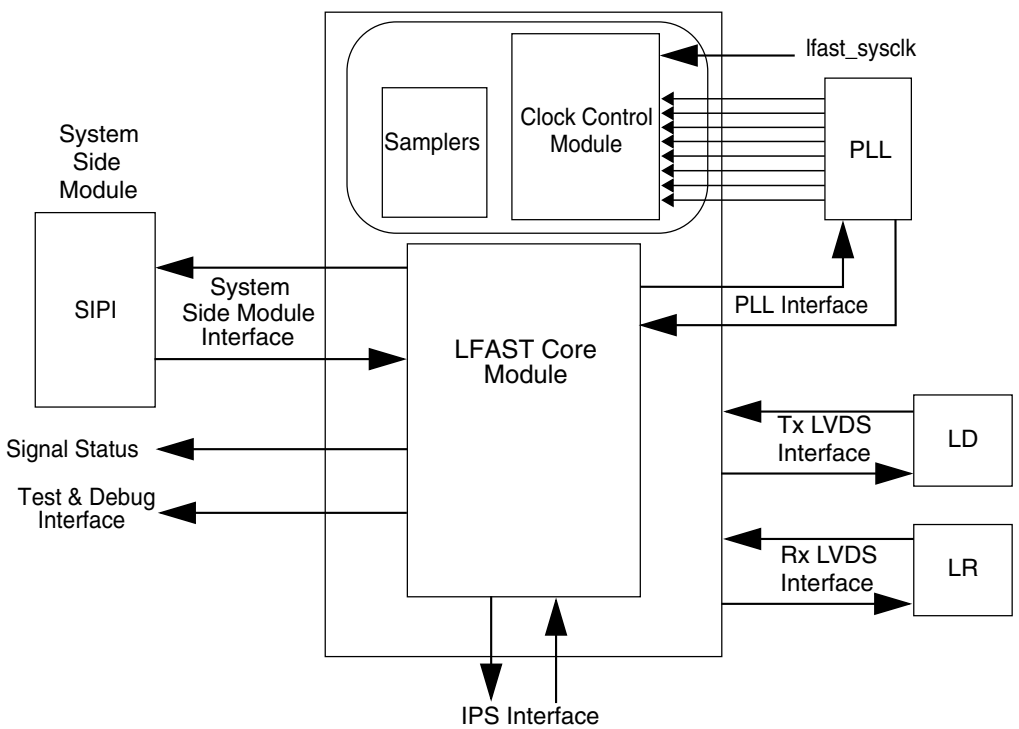


Figure 51-1. LFAST block diagram

51.3 External signals

LFAST is a five pin interface with the following signals :

- lfast_sysclk
  - Reference clock of the LFAST master and slave
- txdatap/txdatan
  - Differential transmit (Tx) interface pair
- rxdatap/rxdatan
  - Differential receive (Rx) interface pair

LFAST interface is an asynchronous high speed LVDS interface.

51.3.1 LFAST operating data rates

The change of data rate is controlled by the LFAST master by issuing appropriate Interface Control Logical Channel (ICLC) packets to the LFAST slave. Henceforth, the data rate 6.5 Mbps/5 Mbps (lfast_sysclk ÷ 4 or lfast_sysclk ÷ 2) is referred to as low data rate, and the high data rate is in the Data Sheet.

51.4 LFAST frame structure

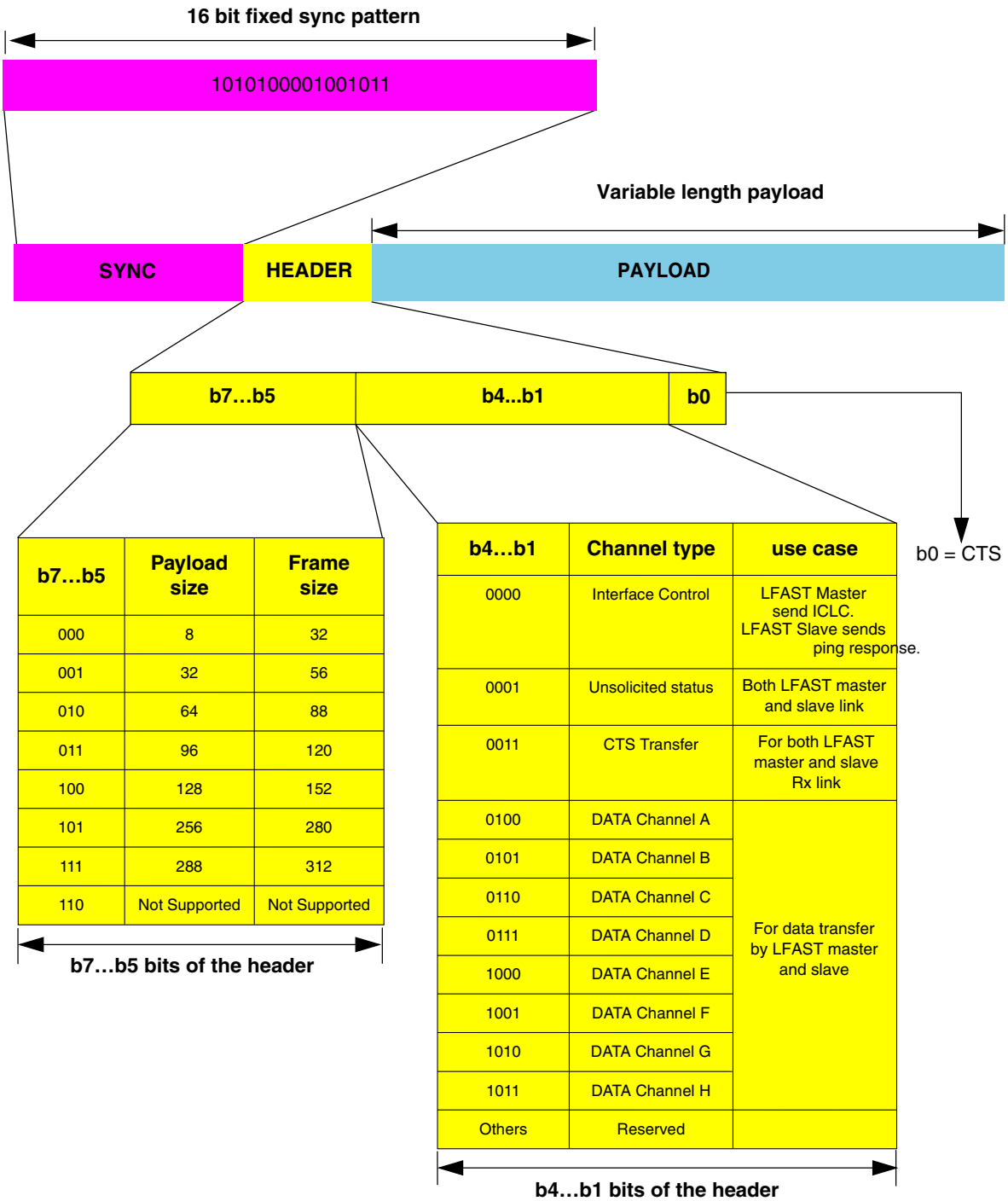
A LFAST frame is made up of three fields:

- Sync pattern
- Header
- Payload

Sync pattern and header fields are of fixed length.

Sync pattern is used to synchronize incoming data stream in LFAST module.

Header field of the frame distinguishes various types of data and control transferred and also contains information about the length of the payload. The payload field is the actual data that is transferred across the channel. See Figure 51-2 for details of the LFAST frame.



b7…b5	Payload size	Frame size
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
111	288	312
110	Not Supported	Not Supported

b4…b1	Channel type	use case
0000	Interface Control	LFAST Master send ICLC. LFAST Slave sends ping response.
0001	Unsolicited status	Both LFAST master and slave link
0011	CTS Transfer	For both LFAST master and slave Rx link
0100	DATA Channel A	For data transfer by LFAST master and slave
0101	DATA Channel B	
0110	DATA Channel C	
0111	DATA Channel D	
1000	DATA Channel E	
1001	DATA Channel F	
1010	DATA Channel G	
1011	DATA Channel H	
Others	Reserved	

Figure 51-2. LFAST frame structure

The same protocol is used on both transmit and receive interfaces for communications of data, control and status information. A synchronization pattern (16 bits) and header (8 bits) are present in every frame.

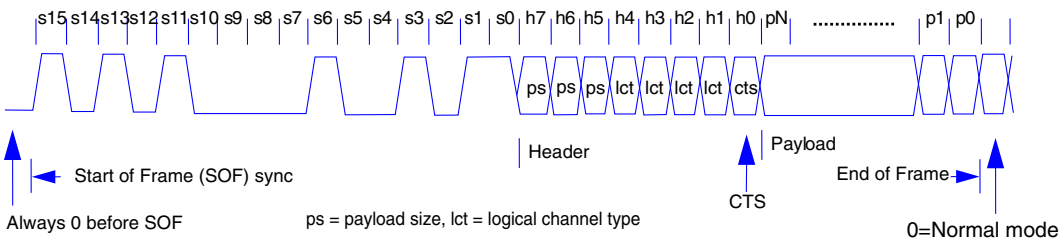


Figure 51-3. Serial frame structure

The frame structure is shown in Figure 51-3 and consists of the following:

- **16-bit synchronization pattern:** Used for clock synchronization and pattern recognition. The frame synchronization code at the start of every frame is a unique reserved word that is used to identify whether the data received is the start of the frame and for synchronization (clock phase extraction) with the stream data. A synchronous sequence is used to give a high quality auto correlation. The LFAST 16-bit synchronization sequence = 1010_1000_0100_1011b = A84Bh.
- **Header - 3 MSB (b7 - b5):** Defines the payload size as shown in Table 51-1 :

Table 51-1. Header payload sizes

b7-b5(bin)	Payload Size	Frame Size
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
110	—	—
111	288	312

- **Header - (b4 - b1):** Defines the logical channel types, which indicate the type of payload that the frame carries. How the payload field of a frame is used for any other logical channel type is system side module specific except in the case of the interface control logical channel type and clear to send (CTS) frame.
- **Header - (b0):** CTS on the both LFAST master and slave devices.

- **Payload:** Content dependent upon frame type.
- **Bit after frame:** This bit determines entry into Sleep mode (1 = Sleep mode, 0 = normal mode)

51.5 Features

- Supports dual mode (register configurable Master/Slave).
- Supports asynchronous data transfer up to the maximum data rate shown in the product Data Sheet.
- Transmits and receives data, CTS, ICLC and unsolicited frames.
- Receives ICLC frames
- Provision of five interrupts for Tx and Rx channels.
- Supports processor controlled transfer of ICLC frame with 8-bit payload size to implement the data rate changes and test modes.
- Supports LFAST defined CTS controlled data transfer. No dependency between data transmission and reception unless CTS mode is enabled.
- Supports flow control using sliding window protocol.
- Provides configurable frame length for data frame with variable payload sizes of 32, 64, 96, 128, 256 or 288 bits.
- Provides transmit of data frame length with 96 bits of payload size and reception of data frame with 128 bits of payload size.
- Provides configurable frame length for unsolicited frame with variable payload sizes of 8, 32, 64, 96, 128, 256 or 288 bits.
- Supports PLL configuration (for example, feedback loop divider, etc.) through registers.
- Supports LVDS configuration through registers.
- Supports multiple loopback modes for checking the physical interface.
- Supports automatic ping response generation in slave mode.
- Supports for detection of unsupported channel number and unsupported payload size.

51.6 Memory map and register definition

All registers are 32 bits wide.

NOTE

Read/Write accesses to all unimplemented registers and write accesses to Read only register will return a Transfer Error.

LFAST memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	LFAST Mode Configuration Register (LFAST_MCR)	32	R/W	See section	51.6.1/2140
4	LFAST Speed Control Register (LFAST_SCR)	32	R/W	0001_0000h	51.6.2/2142
8	LFAST Correlator Control Register (LFAST_COCR)	32	R/W	0000_000Eh	51.6.3/2143
C	LFAST Test Mode Control Register (LFAST_TMCR)	32	R/W	0000_0000h	51.6.4/2145
10	LFAST Auto Loopback Control Register (LFAST_ALCR)	32	R/W	0000_0000h	51.6.5/2146
14	LFAST Rate Change Delay Control Register (LFAST_RCDCR)	32	R/W	000F_0000h	51.6.6/2147
18	LFAST Wakeup Delay Control Register (LFAST_SLCR)	32	R/W	1201_5F02h	51.6.7/2147
1C	LFAST ICLC Control Register (LFAST_ICR)	32	R/W	0000_0000h	51.6.8/2149
20	LFAST Ping Control Register (LFAST_PICR)	32	R/W	0000_80CAh	51.6.9/2150
2C	LFAST Rx FIFO CTS Control Register (LFAST_RFCR)	32	R/W	000F_0009h	51.6.10/ 2150
30	LFAST Tx Interrupt Enable Register (LFAST_TIER)	32	R/W	0000_0000h	51.6.11/ 2151
34	LFAST Rx Interrupt Enable Register (LFAST_RIER)	32	R/W	0000_0000h	51.6.12/ 2152
38	LFAST Rx ICLC Interrupt Enable Register (LFAST_RIIER)	32	R/W	0000_0000h	51.6.13/ 2154
3C	LFAST PLL Control Register (LFAST_PLLCR)	32	R/W	0000_005Ch	51.6.14/ 2156
40	LFAST LVDS Control Register (LFAST_LCR)	32	R/W	See section	51.6.15/ 2158
44	LFAST Unsolicited Tx Control Register (LFAST_UNSTCR)	32	R/W	0000_0000h	51.6.16/ 2161
48	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR0)	32	R/W	0000_0000h	51.6.17/ 2161
4C	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR1)	32	R/W	0000_0000h	51.6.17/ 2161
50	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR2)	32	R/W	0000_0000h	51.6.17/ 2161
54	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR3)	32	R/W	0000_0000h	51.6.17/ 2161

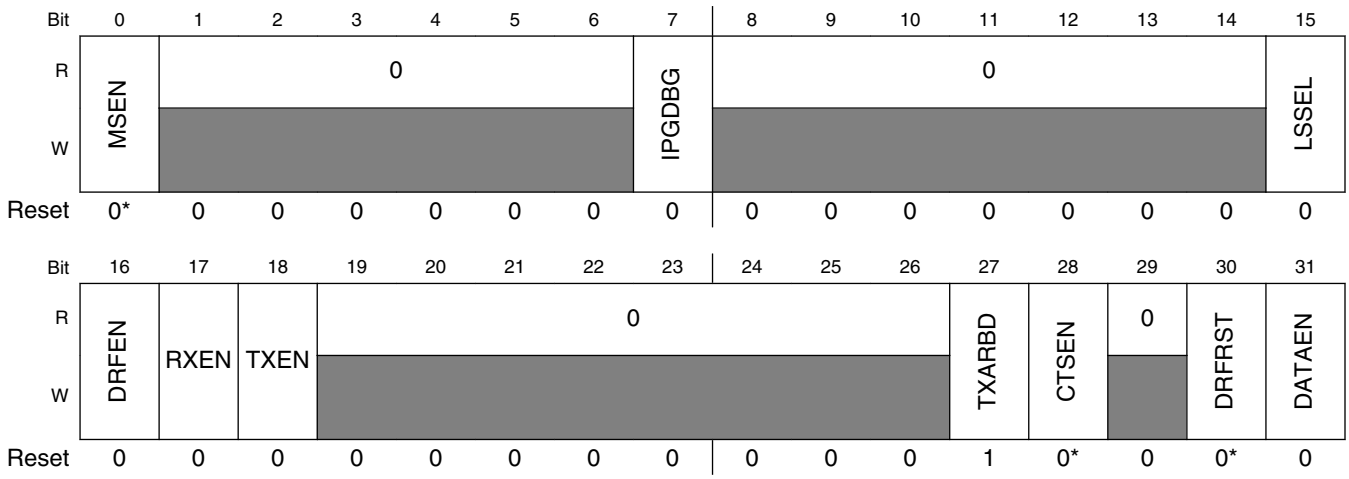
Table continues on the next page...

LFAST memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
58	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR4)	32	R/W	0000_0000h	51.6.17/ 2161
5C	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR5)	32	R/W	0000_0000h	51.6.17/ 2161
60	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR6)	32	R/W	0000_0000h	51.6.17/ 2161
64	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR7)	32	R/W	0000_0000h	51.6.17/ 2161
68	LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR8)	32	R/W	0000_0000h	51.6.17/ 2161
80	LFAST Global Status Register (LFAST_GSR)	32	R	See section	51.6.18/ 2162
84	LFAST Ping Status Register (LFAST_PISR)	32	R	0000_0000h	51.6.19/ 2163
94	LFAST Data Frame Status Register (LFAST_DFSR)	32	R	0000_0000h	51.6.20/ 2164
98	LFAST Tx Interrupt Status Register (LFAST_TISR)	32	R/W	0000_0000h	51.6.21/ 2165
9C	LFAST Rx Interrupt Status Register (LFAST_RISR)	32	R/W	0000_0000h	51.6.22/ 2166
A0	LFAST Rx ICLC Interrupt Status Register (LFAST_RIISR)	32	w1c	0000_0000h	51.6.23/ 2168
A4	LFAST PLL and LVDS Status Register (LFAST_PLLLSR)	32	R	0002_0003h	51.6.24/ 2170
A8	LFAST Unsolicited Rx Status Register (LFAST_UNSRSR)	32	R/W	0000_0000h	51.6.25/ 2171
AC	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR0)	32	R	0000_0000h	51.6.26/ 2172
B0	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR1)	32	R	0000_0000h	51.6.26/ 2172
B4	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR2)	32	R	0000_0000h	51.6.26/ 2172
B8	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR3)	32	R	0000_0000h	51.6.26/ 2172
BC	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR4)	32	R	0000_0000h	51.6.26/ 2172
C0	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR5)	32	R	0000_0000h	51.6.26/ 2172
C4	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR6)	32	R	0000_0000h	51.6.26/ 2172
C8	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR7)	32	R	0000_0000h	51.6.26/ 2172
CC	LFAST Unsolicited Rx Data Register (LFAST_UNSRDR8)	32	R	0000_0000h	51.6.26/ 2172

51.6.1 LFAST Mode Configuration Register (LFAST_MCR)

Address: 0h base + 0h offset = 0h



* Notes:
- DRFRST field: Set by user software and then cleared by system hardware.
- CTSEN field: Only writable when MCR[DRFEN] = 0.
- MSEN field: Writable only once after the asynchronous reset.

LFAST_MCR field descriptions

Field	Description
0 MSEN	LFAST Master or Slave mode Enable. This bit selects either the LFAST master or slave functionality. 0 Enable the modules LFAST Slave functionality only. 1 Enable the modules LFAST Master functionality only.
1–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 IPGDBG	Control bit to enable support for IPG Debug mode. This mode is indicated by assertion of IPG debug mode signal. 0 IPG debug mode enable signal will be ignored. 1 IPG debug mode enable signal will not be ignored.
8–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 LSSEL	Selects the fraction of sysclk in Low Speed Select mode (see Slow speed clock for details). 0 Low Speed Mode in which the lfast_sysclk input is used to generate 4 phases of lfast_sysclk/2. 1 Low Speed Mode in which the lfast_sysclk input is used to generate 4 phases of lfast_sysclk/4.
16 DRFEN	LFAST Enable. This bit enables/disables the reception and transfer of LFAST device. 0 LFAST is immediately disabled. All current/pending requests are terminated and the Tx and Rx data FIFOs are flushed. If this bit is cleared in the middle of a transmit/receive operation, then that operation is terminated immediately and nothing is transmitted/received further. All the programmable

Table continues on the next page...

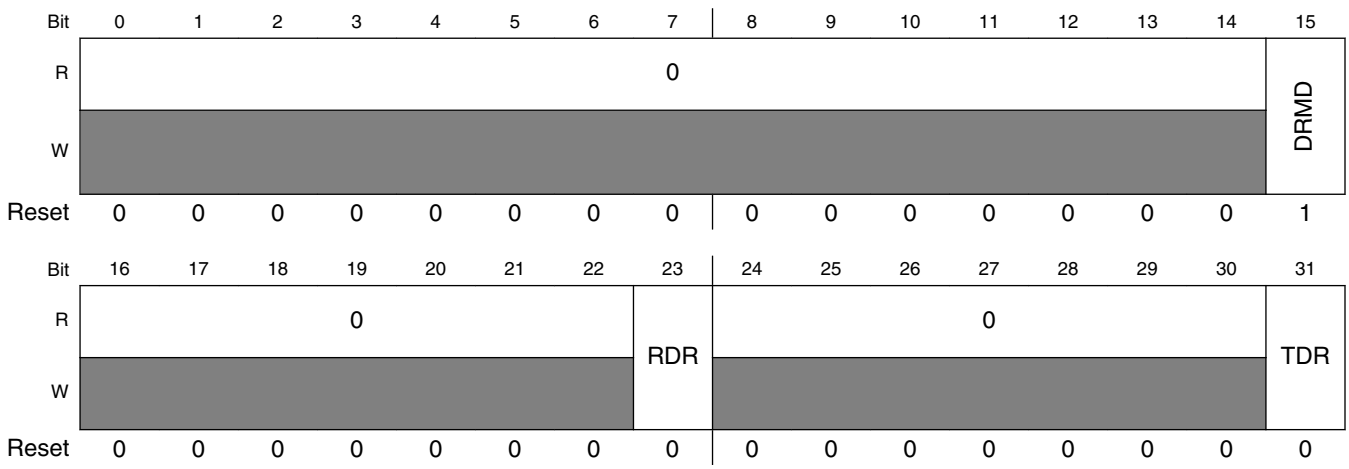
LFAST_MCR field descriptions (continued)

Field	Description
	registers retain their values and status registers are cleared to their reset values. Registers read/write operations can be performed through the IPS Bus. 1 LFAST is Enabled.
17 RXEN	LFAST Receiver Enable. This bit controls the reception of the frames and decoding on the LFAST device. This bit also disables the Rx LVDS Line Receiver (LR). 0 Receiver Interface is disabled. If this bit is cleared during a data transfer, the current frame is received and then the Rx block is disabled. After the Rx block is disabled, all new frames from LFAST peer device are ignored. System Side Module Rx interface isn't disabled by this bit. 1 Receiver Interface is Enabled.
18 TXEN	LFAST Transmitter Enable. This bit controls the transmission of frames from the LFAST device and disables the Tx LVDS LD. This bit can also be modified by LFAST slave H/W on reception of an ICLC command frame. This field can only be written in dual mode (LFAST_GSR[DUALMD] = 1). 0 LFAST transmitter Interface is disabled. No new request is accepted but ongoing request is served. 1 LFAST transmitter Interface is enabled. New requests are accepted.
19–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 TXARBD	Tx Arbiter Disable. This bit enables/disables the Tx block arbiter. Current frame transfer is completed, but new frame requests are ignored. 0 Enable Tx arbiter and framer. When enabled it takes all the frame request and services based on priority. 1 Disable Tx arbiter and framer. All frame requests are ignored.
28 CTSEN	CTS Enable. This bit defines the Push-Pull mode of the LFAST devices receiver. This bit is used to enable/disable CTS mode of the Tx block. This bit is only writable when MCR[DRFEN] = 0. 0 CTS mode is disabled. Indicates that the device is in Push mode. The CTS bit of frames transmitted is 1. The CTS bit doesn't represent the status of Rx FIFO. 1 CTS mode is enabled. The CTS bit of all transmit frames is set when the Rx FIFO empty space is on or above higher threshold, and cleared when the Rx FIFO empty space is on or below lower threshold.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 DRFRST	LFAST Soft Reset. This bit is automatically cleared after Reset.. 0 No Soft Reset 1 Soft Reset to LFAST is asserted. When set it causes a reset of the LFAST module; all the registers will be reset to their default values and all the FIFOs will be flushed.
31 DATAEN	DATA Frame Enable. This bit enables/disables the transmission and reception of data frames between the LFAST master and slave devices. 0 Data frame transmission and reception is disabled. Tx data frame requests are ignored by the transmitter. Frame with LCT of data frame is ignored by the receiver. 1 Data frame transmission and reception is enabled. Tx data frame requests are serviced by the transmitter. Frame with LCT of data frame is received and placed into the Rx data FIFO.

51.6.2 LFAST Speed Control Register (LFAST_SCR)

The SCR is used to configure the Rx and Tx data rate of the LFAST.

Address: 0h base + 4h offset = 4h



LFAST_SCR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 DRMD	Data Rate Controller mode. Defines the mode setting for LFAST slave device by S/W or LFAST master. 0 S/W controls the Data Rate controller mode. In LFAST Slave the ICLC frames for rate change have no affect on the Data rate. 1 In LFAST Slave the reception of ICLC frame for rate change sets appropriate speed mode. In LFAST Master the SCR[DRMD] should be 0.
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 RDR	Receiver Data Rate. This bit defines the receiver data rate. For LFAST Master: • S/W should program this bit only after transmission of an ICLC frame changing the speed mode of the slaves Tx interface. For LFAST Slave: • When SCR[DRMD] = 1, the H/W programs this bit on reception of ICLC frame for changing speed mode of Slaves Rx interface. • The S/W can program this bit when SCR[DRMD] = 0. • This bit is cleared on MCR[DRFEN] negation. 0 Data rate of Rx block is low speed. 1 Data rate of Rx block is high speed.

Table continues on the next page...

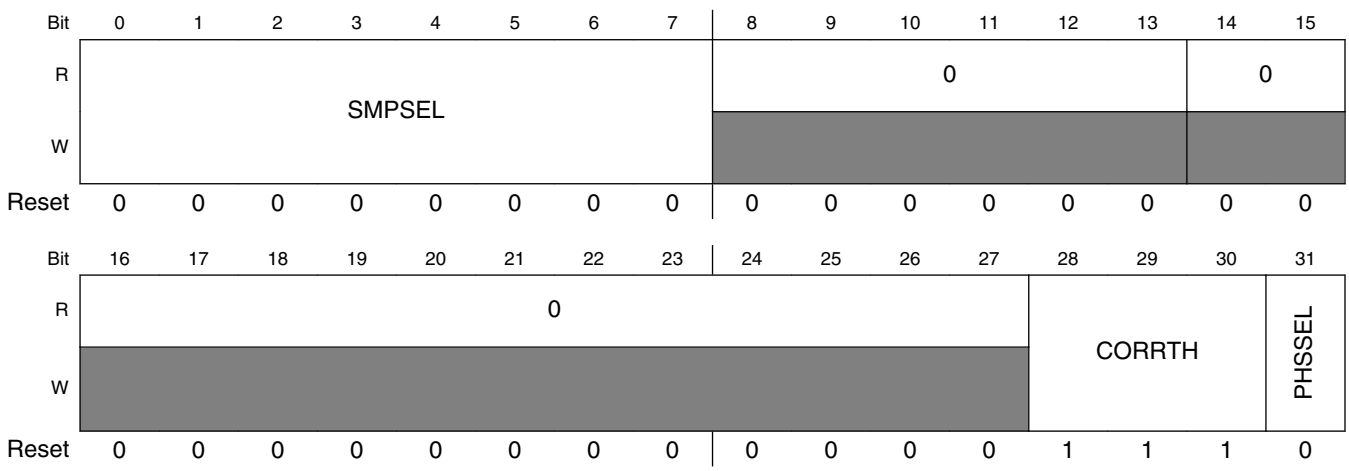
LFAST_SCR field descriptions (continued)

Field	Description
24–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TDR	Transmit Data Rate. This bit defines the transmitter data rate. For LFAST Master: • S/W should program this bit only after transmission of an ICLC frame changing the speed mode of the slaves Rx interface. For LFAST Slave: • When SCR[DRMD] = 1, the H/W programs this bit on reception of ICLC frame for changing speed mode of Slaves Tx interface. • The S/W can program this bit when SCR[DRMD] = 0. • This bit is cleared on MCR[DRFEN] negation. 0 Data rate of Tx block is low speed. 1 Data rate of Tx block is high speed.

51.6.3 LFAST Correlator Control Register (LFAST_COCR)

The COCR is used to select the sampler data path and the number of bits of correlation to be used.

Address: 0h base + 8h offset = 8h



LFAST_COCR field descriptions

Field	Description
0–7 SMPSEL	Sampler Data Path Selector (overrides the correlator selection). Defines the sampler data path to be activated at all the times. All the bits should be 0 (00h) for Sampler Data Path to be selected by the correlator. In Low Speed mode only Sampler Data Paths 0-3 are valid. This field can only be written when MCR[RXEN] = 0.

Table continues on the next page...

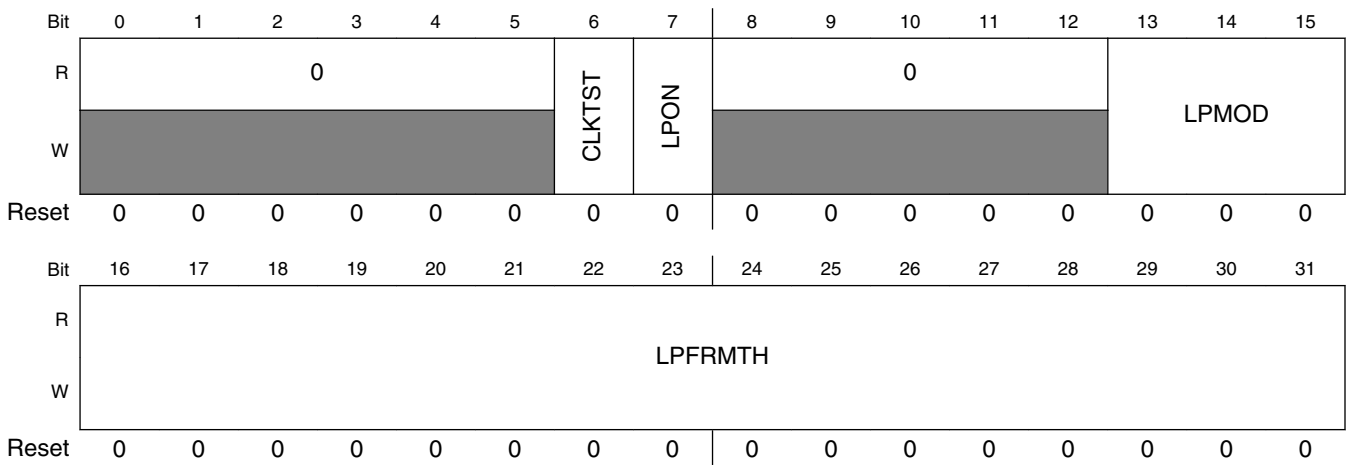
LFAST_COCR field descriptions (continued)

Field	Description
	00h Sampler Data Path selected by correlator 01h Sampler Data Path 0 selected 02h Sampler Data Path 1 selected 04h Sampler Data Path 2 selected 08h Sampler Data Path 3 selected 10h Sampler Data Path 4 selected 20h Sampler Data Path 5 selected 40h Sampler Data Path 6 selected others Sampler Data Path 7 selected
8–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–30 CORRTH	Correlator threshold level. Defines the correlation threshold level. This field can only be written when MCR[RXEN] = 0. 000 9 Bits of correlation 001 10 Bits of correlation ... 110 15 Bits of correlation 111 16 Bits of correlation
31 PHSSEL	Polyphase 8 or 4 phase selection. Defines the number of phases for the polyphase generator used. This bit is ignored in low speed mode since only 4 Phases are used in low speed mode. In High Speed mode phase 0, 2, 4 and 6 are used when 4 phase mode is selected. This field can only be written when MCR[RXEN] = 0 0 8 phases 1 4 phases

51.6.4 LFAST Test Mode Control Register (LFAST_TMCR)

The TMCR enables and configures the LFAST clock test and loopback modes.

Address: 0h base + Ch offset = Ch



LFAST_TMCR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 CLKTST	Clock Test mode. S/W can define when the clock test mode is enabled. This bit can also be set and cleared by LFAST slave H/W on reception of an ICLC command. 1 Clock Test mode enabled 0 Clock Test mode disabled
7 LPON	Loopback mode Logic Enable. S/W can define when the loopback logic is enabled. This bit can also be written by LFAST slave H/W on reception of an ICLC command. 1 Loopback mode is enabled 0 Loopback mode is disabled
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 LPMOD	Loopback mode. Defines the type of loopback mode enabled. This field can only be written when TMCR[LPON] = 0. 000 Rx loopback 001 Rx LVDS loopback 010 Tx loopback without automatic frame generation 011 Tx loopback with automatic frame generation 100 Tx LVDS loopback (external) with automatic frame generation 101 Reserved

Table continues on the next page...

LFAST_TMCR field descriptions (continued)

Field	Description
	110 Reserved 111 Reserved
16–31 LPFRMTH	Loopback check mode valid pass frames threshold value. Defines the number of frames to verify before setting GCR[LPFPDV] when running in Automatic Loopback Frame mode. The loopback frame is considered pass when the payload is CBh, header is 13h and sync is valid. This mode is valid only when TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. This field can only be written when TMCR[LPON] = 0. 0000h Reserved.(Not to be used) 0001h Check 1 frame have correct sync, header and payload. ... FFFEh Check 65534 frames have correct sync, header and payload. FFFFh Check 65535 frames have correct sync, header and payload.

51.6.5 LFAST Auto Loopback Control Register (LFAST_ALCR)

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															LPCNTEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LPFMCNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_ALCR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 LPCNTEN	Auto Loopback Frame Transmission Count Enable. Enables fixed number of auto pre-defined loopback frame transmission. This field can only be written when TMCR[LPON] = 0. 0 Infinite pre-defined loopback frame transmission enabled 1 Fixed count of pre-defined loopback frame transmission enabled
16–31 LPFMCNT	Auto Loopback Frame Transmission Count. Defines the number of pre-defined auto loopback frames to be sent. The pre-defined loopback frame has payload CBh, header 13h and valid sync. This mode is valid if TMCR[LPMOD] = 011b or TMCR[LPMOD] = 100b. This field can only be written when TMCR[LPON] = 0. 0000h Reserved.(Not to be used) 0001h Send 1 frame with correct sync, header and payload

Table continues on the next page...

LFAST_ALCR field descriptions (continued)

Field	Description
	... FFFEh Send 65534 frames with correct sync, header and payload FFFFh Send 65535 frames with correct sync, header and payload

51.6.6 LFAST Rate Change Delay Control Register (LFAST_RCDCR)

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DRCNT				0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RCDCR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DRCNT	Data Rate Controller Counter Value. Defines the number of cycles of Phase 0 clock needed by the Tx interface Data rate change controller to switch from one speed mode to another. The arbitrator ignores all requests during this period. This field can only be written when MCR[DRFEN] = 1. Number of cycles = DRCNT value.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

51.6.7 LFAST Wakeup Delay Control Register (LFAST_SLCR)

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HSCNT								0				LSCNT				HWKCNT								0				LWKCNT			
W																																
Reset	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	1	1	1	1	1	0	0	0	0	0	0	1	0

LFAST_SLCR field descriptions

Field	Description
0–7 HSCNT	High Speed Sleep mode Exit Time. Defines 1/4 of the number of the high speed clock cycle wait after the negation of the LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of the Line Driver (LD) after negation of its sleep signal, and the wakeup time of the LR of the peer LFAST. This field can only be written when MCR[DRFEN] = 0. 00h 0 cycle 01h 1 cycle ...

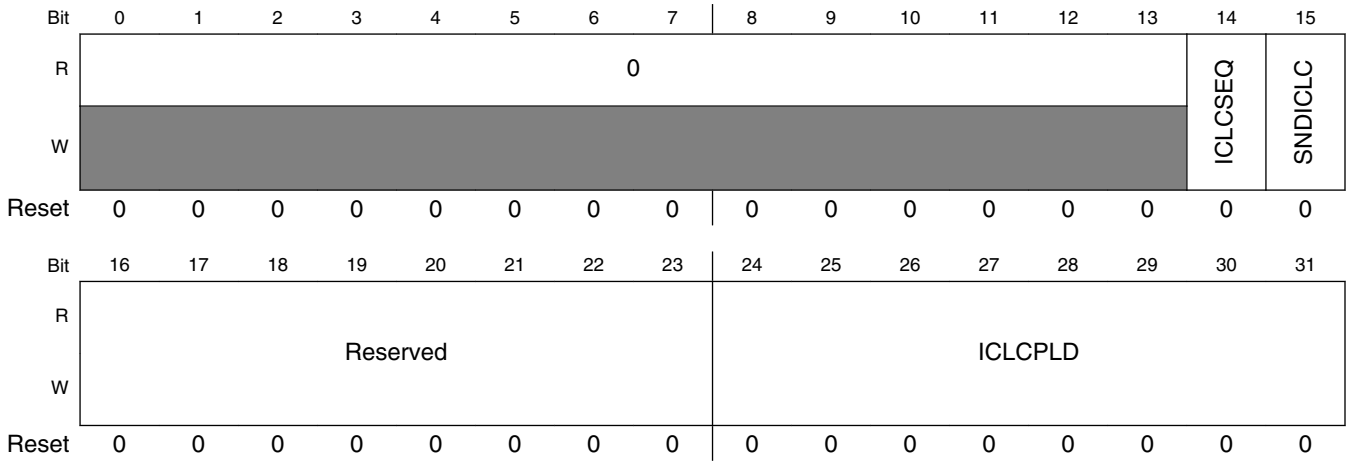
Table continues on the next page...

LFAST_SLCR field descriptions (continued)

Field	Description
	12h 18 cycles (200 ns + 8 cycles of High speed clock) ... FEh 254 cycles FFh 255 cycles
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 LSCNT	Low Speed Sleep mode Exit Time. Defines 1/4 of the number of Low speed clock cycle wait after the negation of LVDS LD sleep signal, and before the start of new frame. This wait is the summation of settling time of LD after negation of LVDS LD sleep signal, and the wakeup time of the LR of the peer LFAST. This field can only be written when MCR[DRFEN] = 0. 0h 0 cycle 1h 1 cycle (200ns + 1 cycle of Low speed clock) ... Eh 14 cycles Fh 15 cycles
16–23 HWKCNT	Wake Up time for the LD. Defines the 1/4 of the number of High speed clock cycles used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from moving from Shutdown to Normal State in High speed mode. This field can only be written when MCR[DRFEN] = 0. 00h 0 cycles 01h 1 cycle ... 5Fh 95 cycles (1.18 µs) ... FFh 255 cycles Maximum
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 LWKCNT	Wake Up time for the LD. Defines the 1/4 of the number Low speed clock used during the wakeup of the LD from shutdown mode. This is time required by the LD to move from Shutdown to Normal State in Low speed mode. This field can only be written when MCR[DRFEN] = 0. 0h 0 cycle 1h 1 cycle 2h 2 cycles (1.18 µs) ... Eh 14 cycles Fh 15 cycles

51.6.8 LFAST ICLC Control Register (LFAST_ICR)

Address: 0h base + 1Ch offset = 1Ch

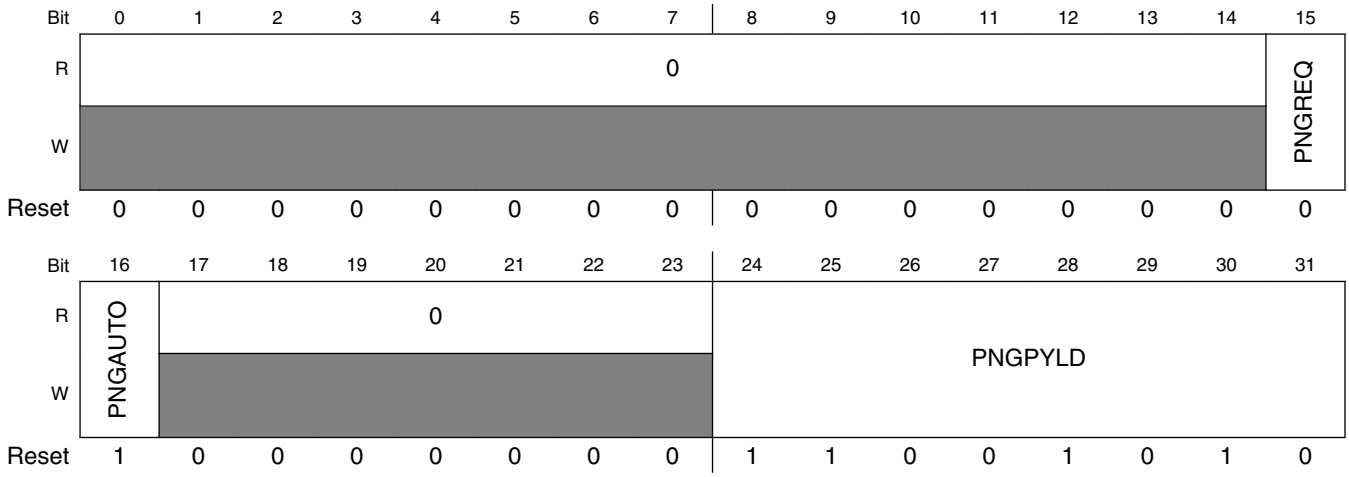


LFAST_ICR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 ICLCSEQ	ICLC enabled. This bit should be set whenever the S/W is performing a series of ICLC frame transfers. This bit ensures only ICLC frame transmission request is serviced, other pending frames request are ignored. 0 Single ICLC frame transfer. 1 S/W is performing ICLC frame transfers. Only the ICLC frames will be scheduled during this period. All the other frames will be scheduled only after ICR[ICLCSEQ] = 0.
15 SNDICLC	ICLC frame request. This bit is set to initiate the transfer of ICLC frame by LFAST master. This bit should be set (ICR[SNDICLC] = 1) after writing the required ICLC payload to be transmitted in the ICLCPLD field of the register. This bit is self clearing, which will be cleared when ICLC frame transfer is complete. Set by user software and cleared by system hardware. 0 No Valid ICLC frame for transfer. 1 Valid ICLC frame for transfer.
16–23 Reserved	This field is reserved.
24–31 ICLCPLD	ICLC Payload. This field is used to program the payload of the ICLC frame to be transmitted. New ICLC payload should be set when ICR[SNDICLC] = 0. This field can only be written when ICR[SNDICLC] = 0 (see Table 51-53 for supported ICLC payloads).

51.6.9 LFAST Ping Control Register (LFAST_PICR)

Address: 0h base + 20h offset = 20h

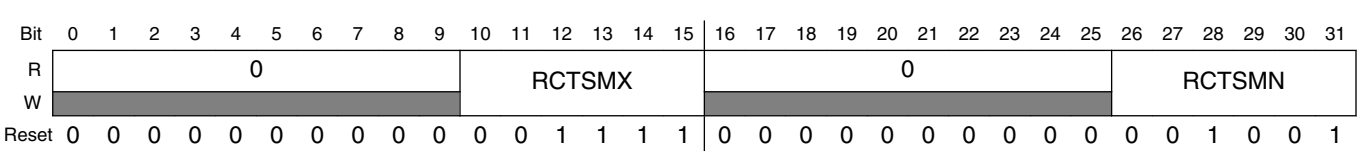


LFAST_PICR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 PNGREQ	Ping Response Frame Request. This bit is set to initiate the transfer of Ping response frame. Cleared after transmission of Ping response frame. Set by user software and cleared by system hardware. 1 Ping response frame transmission request is queued 0 No pending Ping response frame transmission request
16 PNGAUTO	Ping Response Enable. Defines when ping response should be automatically sent on reception of Ping ICLC frame from LFAST master. This field can only be written when MCR[DRFEN] = 0. 0 Ping response should not be automatically sent. 1 Ping response should be automatically sent.
17–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 PNGPYLD	LFAST Slave: Defines the LFAST slaves ping reply frame payload content. This field can only be written when MCR[DRFEN] = 0. LFAST Master: Defines the expected payload of ping response frame. Used for comparison with ping frame received.

51.6.10 LFAST Rx FIFO CTS Control Register (LFAST_RFCR)

Address: 0h base + 2Ch offset = 2Ch

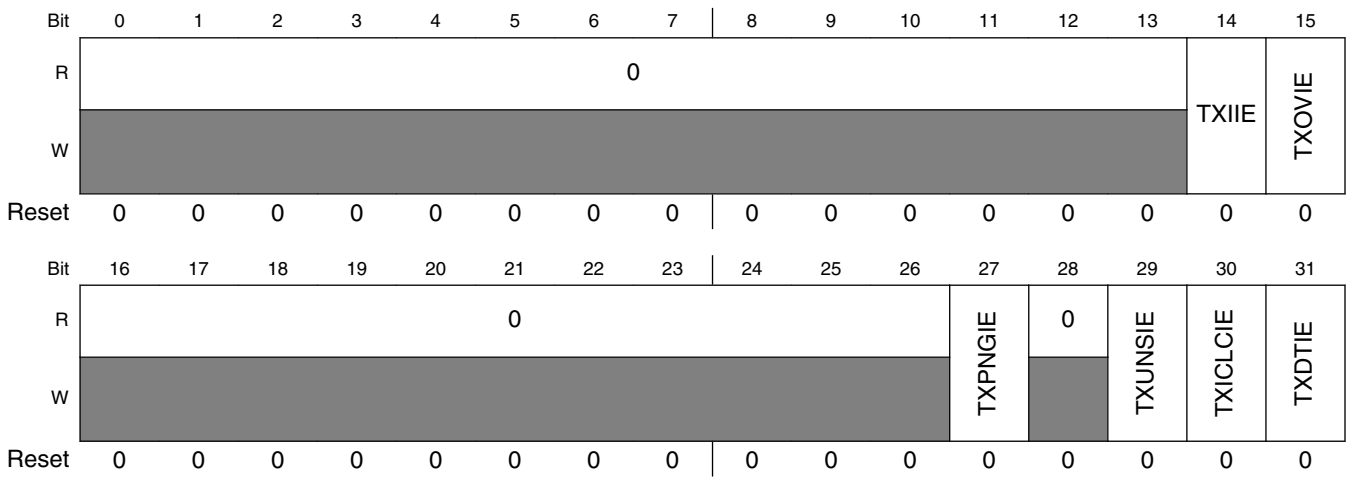


LFAST_RFCR field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 RCTSMX	Rx FIFO Maximum Threshold. Defines the condition for CTS bit of frames to be negated. This field can only be written when LFAST_MCR[DRFEN] = 0.
16–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 RCTSMN	Rx FIFO Minimum Threshold. Defines the condition for CTS bit of frames to be set. This field can only be written when LFAST_MCR[DRFEN] = 0.

51.6.11 LFAST Tx Interrupt Enable Register (LFAST_TIER)

Address: 0h base + 30h offset = 30h



LFAST_TIER field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 TXIIE	Tx Data Interface Not Enabled - (Mask) Enables or disables the interrupt. Tx Data Interface not enabled and a frame is ready to be transmitted 0 Interrupt is disabled 1 Interrupt is enabled
15 TXOVIE	Transmit Data FIFO Overflow Interrupt Enable. 0 Interrupt is disabled 1 Interrupt is enabled
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 TXPNGIE	Ping Response Frame Transmitted Interrupt Enable.

Table continues on the next page...

LFAST_TIER field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 TXUNSIE	Unsolicited Frame transmitted Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
30 TXICLCIE	ICLC Frame transmitted Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
31 TXDTIE	Data Frame transmitted Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled

51.6.12 LFAST Rx Interrupt Enable Register (LFAST_RIER)

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								RXUOIE	RXMNIE	RXMXIE	RXUFIE	RXOFIE	RXSZIE	RXICIE	RXLCEIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												RXCTSIE	RXDIE	RXUNSIE	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RIER field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 RXUOIE	Unsolicited frame register overflow. Indicates existing unsolicited frame hasn't been read and a new unsolicited frame has arrived. 0 Interrupt is disabled 1 Interrupt is enabled

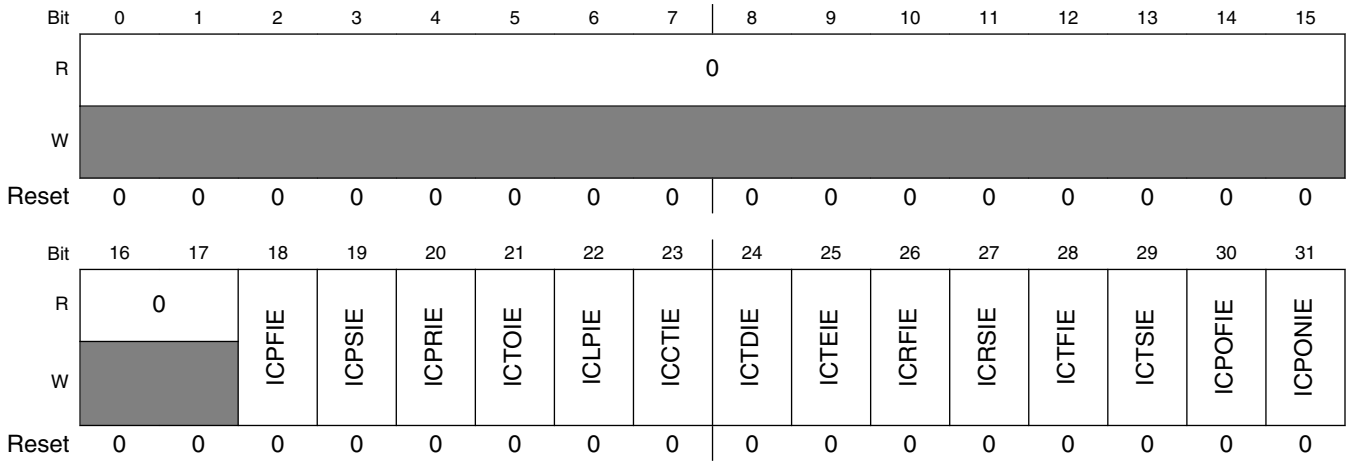
Table continues on the next page...

LFAST_RIER field descriptions (continued)

Field	Description
9 RXMNIE	Rx Data FIFO Min Threshold reached 0 Interrupt is disabled 1 Interrupt is enabled
10 RXMXIE	Rx Data FIFO Max Threshold reached 0 Interrupt is disabled 1 Interrupt is enabled
11 RXUFIE	Rx Data FIFO Underflow 0 Interrupt is disabled 1 Interrupt is enabled
12 RXOFIE	Rx Data FIFO Overflow 0 Interrupt is disabled 1 Interrupt is enabled
13 RXSZIE	Frame with unsupported frame size received. Valid frame sizes are defined in Table 51-57 0 Interrupt is disabled 1 Interrupt is enabled
14 RXICIE	Invalid ICLC code Received 0 Interrupt is disabled 1 Interrupt is enabled
15 RXLCEIE	Invalid Logical Channel Type 0 Interrupt is disabled 1 Interrupt is enabled
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 RXCTSIE	Frame with CTS bit Low Received 0 Interrupt is disabled 1 Interrupt is enabled
29 RXDIE	Data frame received 0 Interrupt is disabled 1 Interrupt is enabled
30 RXUNSIE	Unsolicited Frame received 0 Interrupt is disabled 1 Interrupt is enabled
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

51.6.13 LFAST Rx ICLC Interrupt Enable Register (LFAST_RIIER)

Address: 0h base + 38h offset = 38h



LFAST_RIIER field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 ICPFIE	Ping Frame Response failed 0 Interrupt is disabled 1 Interrupt is enabled
19 ICPSIE	Ping Frame Response successful 0 Interrupt is disabled 1 Interrupt is enabled
20 ICPRIE	ICLC frame for Ping Frame Request received 0 Interrupt is disabled 1 Interrupt is enabled
21 ICTOIE	ICLC frame for Test mode off received 0 Interrupt is disabled 1 Interrupt is enabled
22 ICLPIE	ICLC frame for Loopback On received 0 Interrupt is disabled 1 Interrupt is enabled
23 ICCTIE	ICLC frame for Clk Test mode on received 0 Interrupt is disabled 1 Interrupt is enabled
24 ICTDIE	ICLC frame for LFAST Slaves Tx Interface Disable received

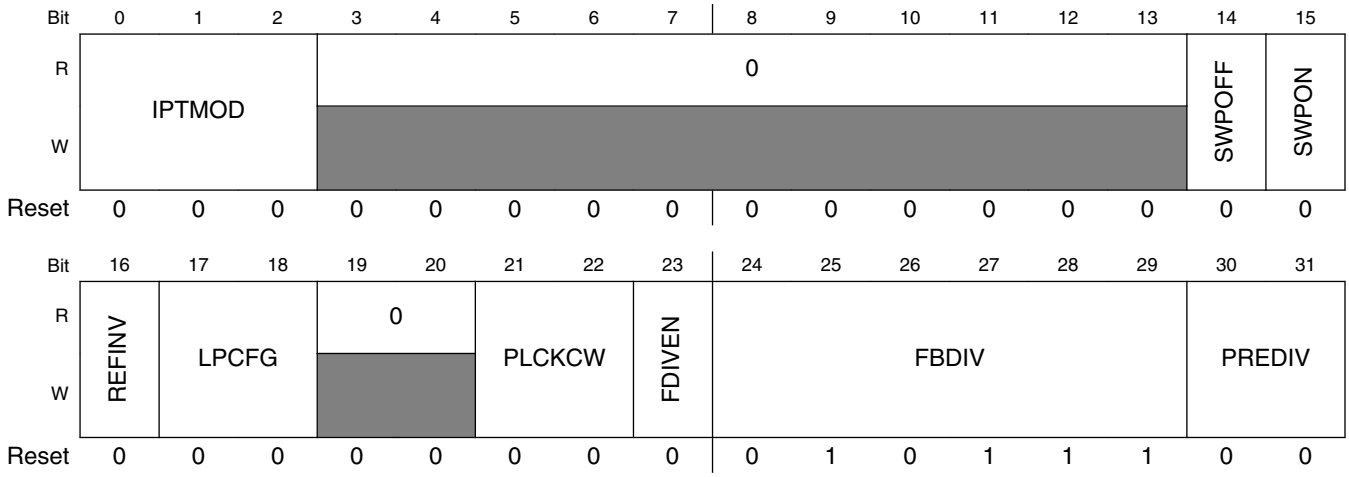
Table continues on the next page...

LFAST_RIIER field descriptions (continued)

Field	Description
	0 Interrupt is disabled 1 Interrupt is enabled
25 ICTEIE	ICLC frame for LFAST Slaves Tx Interface Enable received 0 Interrupt is disabled 1 Interrupt is enabled
26 ICRFIE	ICLC frame for LFAST Slaves Rx Interface fast mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
27 ICRSIE	ICLC frame for LFAST Slaves Rx Interface slow mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
28 ICTFIE	ICLC frame for LFAST Slaves Tx Interface fast mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
29 ICTSIE	ICLC frame for LFAST Slaves Tx Interface slow mode switch received 0 Interrupt is disabled 1 Interrupt is enabled
30 ICPOFIE	ICLC frame for PLL OFF received 0 Interrupt is disabled 1 Interrupt is enabled
31 ICPONIE	ICLC frame for PLL ON received 0 Interrupt is disabled 1 Interrupt is enabled

51.6.14 LFAST PLL Control Register (LFAST_PLLCR)

Address: 0h base + 3Ch offset = 3Ch



LFAST_PLLCR field descriptions

Field	Description
0–2 IPTMOD	Test mode programmability 000 Functional mode 001 Closed Loop 1 010 Force Vctrl 011 Charge Pump Up 100 Charge Pump Up Internal Test 101 Charge Pump Idle 110 Charge Pump Down 111 Closed Loop 2
3–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 SWPOFF	SW signal to turn OFF the PLL. Set by user software and cleared by system hardware. 0 No effect 1 PLL will be turned OFF.
15 SWPON	SW signal to turn ON the PLL. Set by user software and cleared by system hardware. 0 No effect 1 PLL will be turned ON
16 REFINV	Inverts reference clock edge to PFD. If System PLL PFD using same reference clock, enabling this feature will minimize noise injection crosstalk via the substrate. 0 Do not inverted 1 Invert

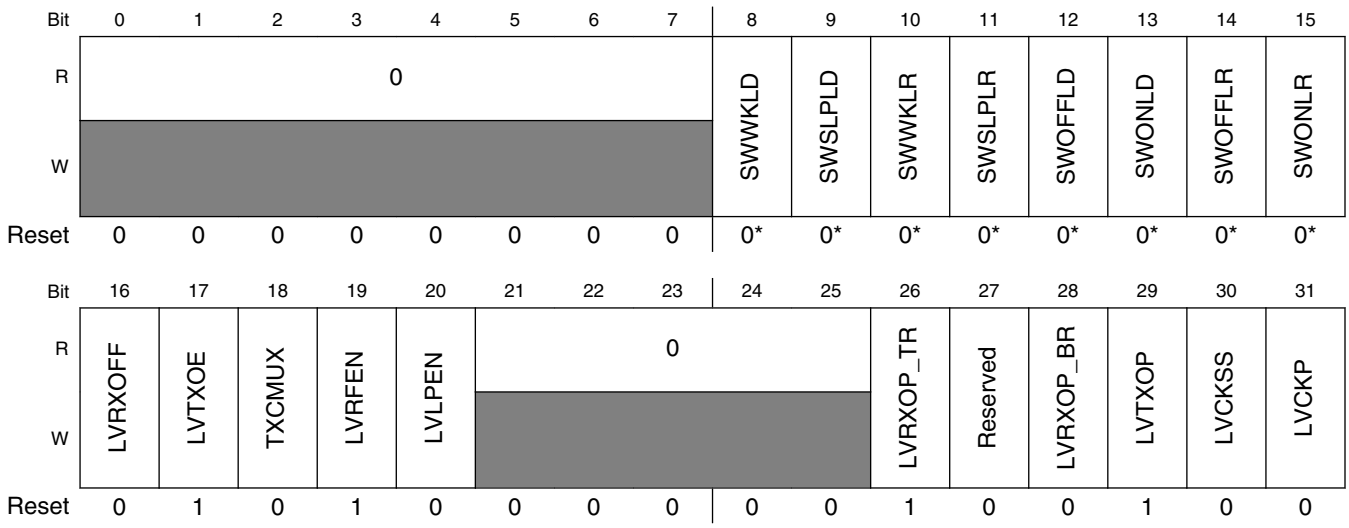
Table continues on the next page...

LFAST_PLLCR field descriptions (continued)

Field	Description
17–18 LPCFG	PLL Loop Optimization for sysclk frequency 00 1 × IBASE current 01 0.5 × IBASE current 10 1.5 × IBASE current 11 2 × IBASE current
19–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–22 PLCKCW	PLL Lock Ready Count Width. Defines the number of cycles PLL waits for before asserting lock_ready flag High 00 1040 cycles 01 520 cycles 10 320 cycles 11 200 cycles
23 FDIVEN	Enable fraction division mode in feedback divider. Enables the division of vco clock output by a factor of (FBDIV + 0.5). 0 Fraction division mode not enabled 1 Fraction division mode enabled
24–29 FBDIV	Feedback Division factor for VCO output clock. This field can only be written when LFAST_MCR[DRFEN] = 0. 00h No clock output 01h – 0Ah Reserved 0Bh Divide by 12 (11.5, if FDIVEN = 1) ... ... 1Fh Divide by 32 (31.5, if FDIVEN = 1) 20h – 3Fh Reserved
30–31 PREDIV	Division factor for PLL Reference Clock input. This field can only be written when LFAST_MCR[DRFEN] = 0. 00 Direct clock passed 01 Divide by 2 10 Divide by 3 11 Divide by 4

51.6.15 LFAST LVDS Control Register (LFAST_LCR)

Address: 0h base + 40h offset = 40h



* Notes:
- SWONLR field: Set by user software and cleared by system hardware.
- SWOFFLR field: Set by user software and cleared by system hardware.
- SWONLD field: Set by user software and cleared by system hardware.
- SWOFFLD field: Set by user software and cleared by system hardware.
- SWSLPLR field: Set by user software and cleared by system hardware.
- SWWKLR field: Set by user software and cleared by system hardware.
- SWSLPLD field: Set by user software and cleared by system hardware.
- SWWKLD field: Set by user software and cleared by system hardware.

LFAST_LCR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 SWWKLD	SW signal to take LVDS LD out of Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be taken out of sleep (provided no other source is trying to put it in sleep)
9 SWSLPLD	SW signal to put LVDS LD into Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be put in sleep

Table continues on the next page...

LFAST_LCR field descriptions (continued)

Field	Description
10 SWWKLR	SW signal to take LVDS LR out of Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be taken out of sleep (provided no other source is trying to put it in sleep)
11 SWSLPLR	SW signal to put LVDS LR into Sleep mode. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be put in sleep (provided no other source is trying to wake it up)
12 SWOFFLD	SW signal to turn OFF the LVDS LD. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be turned OFF(provided no other source is trying to turn the LD ON)
13 SWONLD	SW signal to turn ON the LVDS LD. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LD will be turned ON
14 SWOFFLR	SW signal to turn OFF the LVDS LR. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be turned OFF (provided no other source is trying to turn the LR ON)
15 SWONLR	SW signal to turn ON the LVDS LR. This field can only be written when LFAST_MCR[DRFEN] = 1. **NOTE:** Writes are not reflected in the read of this field. 0 No effect 1 LVDS LR will be turned ON
16 LVRXOFF	Indicates the value driven onto LVDS LR output when in shutdown mode
17 LVTXOE	LVDS LD output buffer enable. 0 LVDS LD output buffer enable is disabled. 1 LVDS LD output buffer enabled
18 TXCMUX	Tx and Clock Mux. The bit can be used to bring out PLL Phase 0 clock on Tx LVDS pad. 0 No effect 1 PLL Phase 0 clock will be brought out to Tx LVDS pad
19 LVRFEN	LVDS pad reference enable 0 LVDS reference pad disabled 1 LVDS reference pad enabled

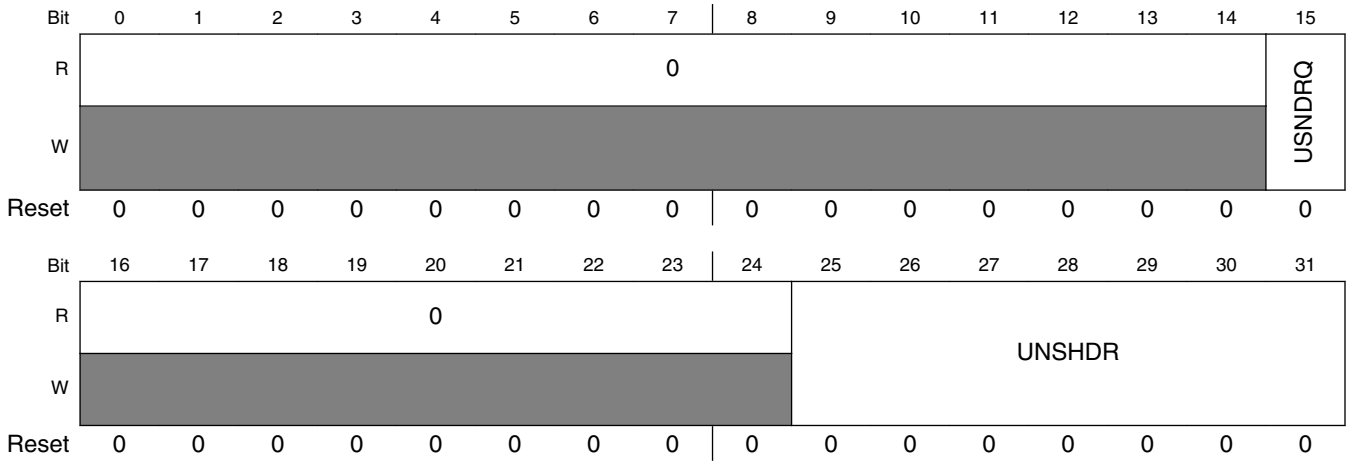
Table continues on the next page...

LFAST_LCR field descriptions (continued)

Field	Description
20 LVLPEN	Tx LVDS internal loopback enable The bit is reflected by output signal ipp_digrf_lvds_lpbk_en.The internal loopback is not intended for board level functionality, so this feature should not be implemented. 0 Tx LVDS normal mode enabled 1 Tx LVDS internal loopback mode enabled
21–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 LVRXOP_TR	Used to enable or disable the on-chip receiver termination resistor in LFAST mode (applies to LVDS pad use for LFAST only): 0 Disable on-chip LFAST receiver termination 1 Enable on-chip LFAST receiver termination
27 Reserved	This field is reserved.
28 LVRXOP_BR	Used to set the bias current for the receiver in LFAST mode. It is recommended to always write 1 to this bit when using the LFAST interface. Writing 0 will allow for a small power savings during lower baud rates (applies to LVDS pad use for LFAST only): 0 Use for LFAST receiver baud rates less than the maximum baud rate. 1 Required for LFAST receiver maximum baud rate.
29 LVTXOP	Control signal for LFASTand Micro-Second Bus selection. 0 Micro-Second Bus selection 1 LFAST Bus selection
30 LVCKSS	LVDS Clock Sync Select This bit is used to adjust the LVDS data sampling to the duty cycle of the clock. It is recommended that a value of zero is used in all cases. A value of one is reserved for specific devices/revisions with different clock timing. 0 normal clock used to sample the LVDS data 1 adjusted clock used to sample the LVDS data
31 LVCKP	LVDS clock select. This bit is used for selecting use of direct pll clock or inverted pll clock in LVDS. 0 Direct pll clock to be used inside the LVDS 1 Inverted pll clock to be used inside the LVDS

51.6.16 LFAST Unsolicited Tx Control Register (LFAST_UNSTCR)

Address: 0h base + 44h offset = 44h

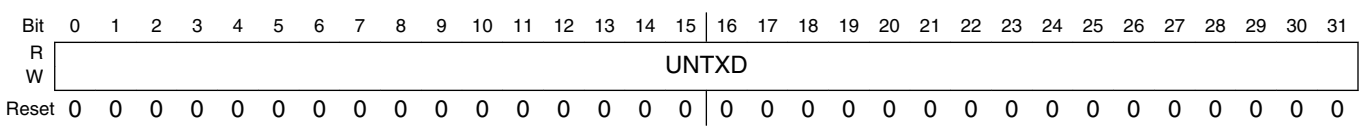


LFAST_UNSTCR field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 USNDRQ	Tx Unsolicited send request. Set by user software and cleared by system hardware. 0 No valid Unsolicited frame exists 1 Valid Unsolicited frame exists for transmission
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 UNSHDR	Tx Unsolicited message header. This field can only be written when LFAST_UNSTCR[USNDRQ] = 0

51.6.17 LFAST Unsolicited Tx Data Registers (LFAST_UNSTDR*n*)

Address: 0h base + 48h offset + (4d × i), where i=0d to 8d

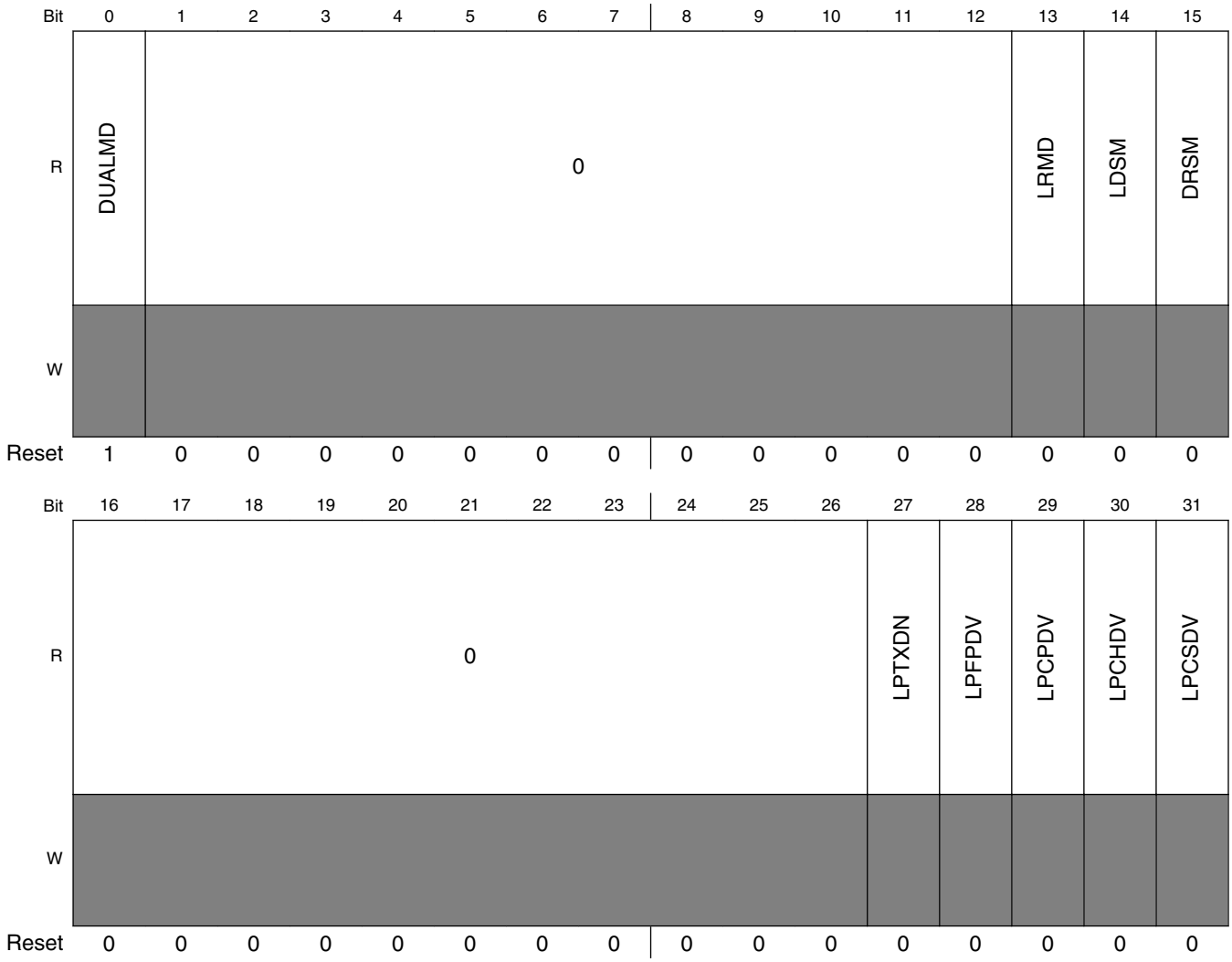


LFAST_UNSTDR*n* field descriptions

Field	Description
0–31 UNTXD	Unsolicited Transmit Data 8-0. This represents 9 registers for Unsolicited transmit data. The first bit to transmitted as part of the payload will be from UNTXD8[31], second bit from UNTXD8[30], and so on. So the last bit transmitted will be from UNTXD0[0] in case of 288 bit payload.

51.6.18 LFAST Global Status Register (LFAST_GSR)

Address: 0h base + 80h offset = 80h



LFAST_GSR field descriptions

Field	Description
0 DUALMD	Indicates the LFAST module is in Dual mode 0 LFAST Module in Slave only mode 1 LFAST Module in Dual mode
1–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 LRMD	Indicates if the Rx Controller is idle/active and that the Rx clocks are enabled. In functional mode this will always be active.

Table continues on the next page...

LFAST_GSR field descriptions (continued)

Field	Description
	0 Rx Controller is in Idle state 1 Rx Controller is active
14 LDSM	Transmit Interface Data Rate Status. Indicates the current speed rate of the Tx controller 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
15 DRSM	Receive Interface Data Rate Status. Indicates the current speed rate of the Rx controller 0 Data rate of LOW speed mode 1 Data rate of HIGH speed mode
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 LPTXDN	Auto loopback frame transmission count reached. The Count of frame is defined by LFAST_ALCR[LPFMCNT]. 0 Auto loopback frame transmission count not reached. 1 Auto loopback frame transmission count reached.
28 LPFPDV	Loopback frame pass threshold reached. 0 Pass frame threshold not reached. 1 Pass frame threshold achieved
29 LPCPDV	Valid payload received during loopback check mode. Indicates whether the Loop back frame received payload is CBh. 0 Payload received is not CBh 1 Payload received is CBh
30 LPCHDV	Valid header received during loopback check mode. Indicates whether the Loop back frame received header is 13h. 0 Header received is not 13h 1 Header received is 13h
31 LPCSDV	Valid synchronization received. 0 Valid Synchronization pattern not detected 1 Valid Synchronization pattern detected

51.6.19 LFAST Ping Status Register (LFAST_PISR)

Address: 0h base + 84h offset = 84h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								RXPNGD							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_PISR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RXPNGD	Ping Data Register. In LFAST Master mode the Ping response ICLC frame received is stored into this register.

51.6.20 LFAST Data Frame Status Register (LFAST_DFSR)

Address: 0h base + 94h offset = 94h

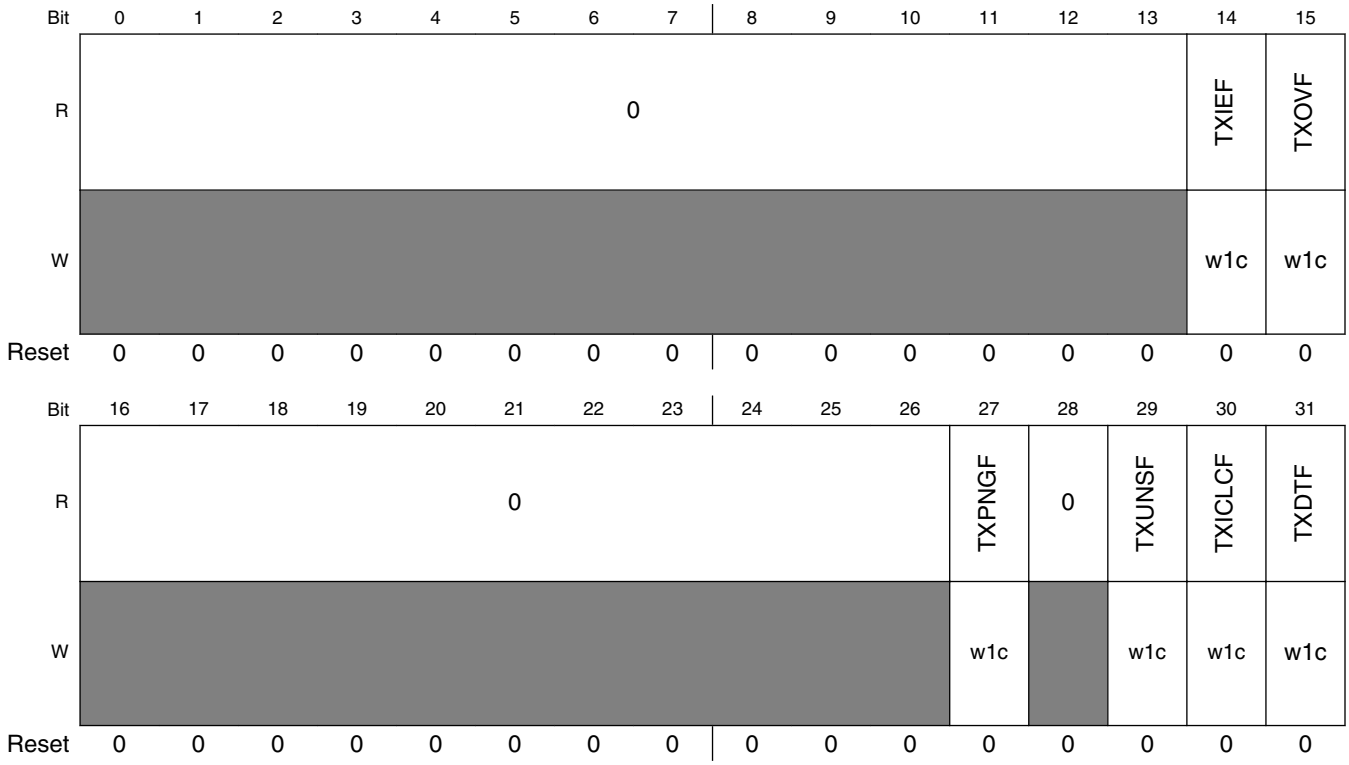
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		RXDCNT						0					RXFCNT			0		TXDCNT						0					TXFCNT		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_DFSR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 RXDCNT	Unread Rx Frame Data Count. Indicates the number of unread data stored in the Rx Data FIFO.
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 RXFCNT	Unread Rx Frame Count. Indicates the number of unread data frames stored in the Rx Data FIFO.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–23 TXDCNT	Unread Tx Frame Data Count. Indicates the number of unread data stored in the Tx Data FIFO.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 TXFCNT	Unread Tx Frame Count.Count of pending Data Frames programed by System Side Module.

51.6.21 LFAST Tx Interrupt Status Register (LFAST_TISR)

Address: 0h base + 98h offset = 98h



LFAST_TISR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 TXIEF	TxData Interface not enabled. Tx Data Interface not enabled and a frame is ready to be transmitted 0 Interrupt event has not occurred 1 Interrupt event has occurred
15 TXOVF	Transmit Data FIFO Overflow Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 TXPNGF	Ping response frame transmitted interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

LFAST_TISR field descriptions (continued)

Field	Description
29 TXUNSF	Unsolicited Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 TXICLCF	ICLC Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 TXDTF	Data Frame transmitted Interrupt 0 Interrupt event has not occurred 1 Interrupt event has occurred

51.6.22 LFAST Rx Interrupt Status Register (LFAST_RISR)

Address: 0h base + 9Ch offset = 9Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								RXUOF	RXMNF	RXMXF	RXUFF	RXOFF	RXSZF	RXICF	RXLCEF
W									w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												RXCTSF	RXDF	RXUNSF	0
W													w1c	w1c	w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RISR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

LFAST_RISR field descriptions (continued)

Field	Description
8 RXUOF	Unsolicited frame register overflow. Indicates existing unsolicited frame hasn't been read and a new unsolicited frame has arrived. 0 Interrupt event has not occurred 1 Interrupt event has occurred
9 RXMNF	Rx Data FIFO Min Threshold reached. 0 Interrupt event has not occurred 1 Interrupt event has occurred
10 RXMXF	Rx Data FIFO Max Threshold reached. 0 Interrupt event has not occurred 1 Interrupt event has occurred
11 RXUFF	Rx Data FIFO Underflow. 0 Interrupt event has not occurred 1 Interrupt event has occurred
12 RXOFF	Rx Data FIFO Overflow. 0 Interrupt event has not occurred 1 Interrupt event has occurred
13 RXSZF	Frame with unsupported frame size received. See "Frames Supported by LFAST interfaces" table for details. 0 Interrupt event has not occurred 1 Interrupt event has occurred - On reception of frame with payload size for a frame other than mentioned in the "Frames Supported by LFAST interfaces" table.
14 RXICF	Invalid ICLC code Received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
15 RXLCEF	Invalid Logical Channel Type. 0 Interrupt event has not occurred 1 Interrupt event has occurred - On reception of frame other than mentioned in the "Frames Supported by LFAST interfaces" table.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 RXCTSF	Frame with CTS bit Low Received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
29 RXDF	Data frame received. 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 RXUNSF	Unsolicited Frame received. 0 Interrupt event has not occurred 1 Interrupt event has occurred

Table continues on the next page...

LFAST_RISR field descriptions (continued)

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

51.6.23 LFAST Rx ICLC Interrupt Status Register (LFAST_RIISR)

Address: 0h base + A0h offset = A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		ICPFF	ICPSF	ICPRF	ICTOF	ICLPF	ICCTF	ICTDF	ICTEF	ICRFF	ICRSF	ICTFF	ICTSF	ICPOFF	ICPONF
W			w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_RIISR field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 ICPFF	Ping Frame Response failed 0 Interrupt event has not occurred 1 Interrupt event has occurred
19 ICPSF	Ping Frame Response successful 0 Interrupt event has not occurred 1 Interrupt event has occurred
20 ICPRF	ICLC Ping Frame Request received

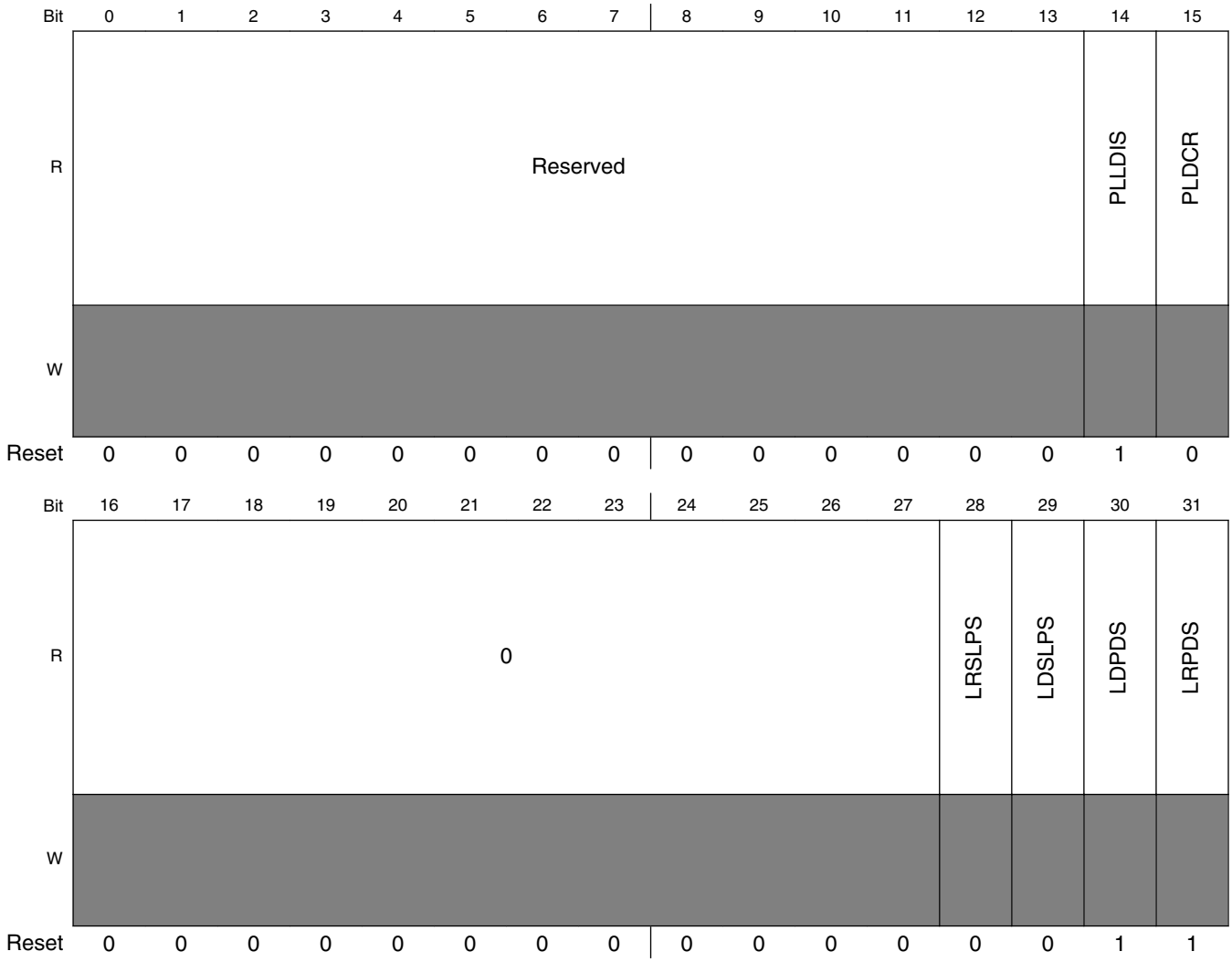
Table continues on the next page...

LFAST_RIISR field descriptions (continued)

Field	Description
	0 Interrupt event has not occurred 1 Interrupt event has occurred
21 ICTOF	ICLC frame for Test mode off received 0 Interrupt event has not occurred 1 Interrupt event has occurred
22 ICLPF	ICLC frame for Loopback On received 0 Interrupt event has not occurred 1 Interrupt event has occurred
23 ICCTF	ICLC frame for Clk Test mode received 0 Interrupt event has not occurred 1 Interrupt event has occurred
24 ICTDF	ICLC frame for LFAST Slaves Tx Interface Disable received 0 Interrupt event has not occurred 1 Interrupt event has occurred
25 ICTEF	ICLC frame for LFAST Slaves Tx Interface Enable received 0 Interrupt event has not occurred 1 Interrupt event has occurred
26 ICRFF	ICLC frame for LFAST Slaves Rx Interface fast mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
27 ICRSF	ICLC frame for LFAST Slaves Rx Interface slow mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
28 ICTFF	ICLC frame for LFAST Slaves Tx Interface fast mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
29 ICTSF	ICLC frame for LFAST Slaves Tx Interface slow mode switch received 0 Interrupt event has not occurred 1 Interrupt event has occurred
30 ICPOFF	ICLC frame for PLL OFF received 0 Interrupt event has not occurred 1 Interrupt event has occurred
31 ICPONF	ICLC frame for PLL ON received 0 Interrupt event has not occurred 1 Interrupt event has occurred

51.6.24 LFAST PLL and LVDS Status Register (LFAST_PLLLSR)

Address: 0h base + A4h offset = A4h



LFAST_PLLLSR field descriptions

Field	Description
0–13 Reserved	This field is reserved.
14 PLLDIS	PLL disable Status. When asserted, PLL is put in the power down state. 0 PLL disable signal is negated. 1 PLL disable signal is asserted.
15 PLDCR	PLL Lock Delay Counter Ready. When asserted this bit indicates that the PLL is locked after N number of reference/PLLCR[PREDIV] cycles 0 PLL Lock delay counter is not decremented to 0 1 PLL Lock delay counter is decremented to 0

Table continues on the next page...

LFAST_PLLLSR field descriptions (continued)

Field	Description
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LRSLPS	This bit indicates the real time status of LR sleep signal 0 LR power sleep signal is negated. 1 LR power sleep signal is asserted.
29 LDSLPS	This bit indicates the real time status of LD sleep signal 0 LD sleep signal is negated. 1 LD sleep signal is asserted.
30 LDPDS	This bit indicates the real time status of LD power down signal When asserted, LD is put in the power down state. 0 LD power down signal is negated. 1 LD power down signal is asserted.
31 LRPDS	This bit indicates the real time status of LR power down signal When asserted, LR is put in the power down state. 0 LR power down signal is negated. 1 LR power down signal is asserted.

51.6.25 LFAST Unsolicited Rx Status Register (LFAST_UNSRSR)

Address: 0h base + A8h offset = A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							URXDV	0					URPCNT		
W								w1c								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_UNSRSR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 URXDV	Unsolicited data valid. Indicates a valid frame exists in the Unsolicited Data registers.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 URPCNT	Rx Unsolicited payload. Indicates the number of bytes of valid Rx unsolicited Data payload present in the LFAST_UNSRDR[8:0].

51.6.26 LFAST Unsolicited Rx Data Register (LFAST_UNSRDR*n*)

Address: 0h base + ACh offset + (4d × i), where i=0d to 8d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	UNRXD																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LFAST_UNSRDR*n* field descriptions

Field	Description
0–31 UNRXD	Unsolicited Receive Data. This represents 9 registers for Unsolicited received data.It is read only register. The first bit received as part of the payload will be stored at UNRXD8[31], second bit at UNRXD8[30], and so on. So the last bit will be stored at UNRXD0[0] in case of 288 bit payload.

51.7 Functional description

51.7.1 Startup procedure

The LFAST requires a sequence of operations before it can be used for communication. The procedure to enable LFAST for proper operation is listed below.

51.7.1.1 LFAST master interface startup procedure

1. After reset the SLCR and RCDCR are programed according to the LVDS parameters in the device data sheet.
2. The PLLCR is programmed with configuration parameters for the PLL.

3. The LCR is programed with configuration parameters of the LVDS.
4. Write MCR[MSEN] = 1. Then select LFAST modes by configuring MCR[CTSEN]and MCR[DATAEN].
5. Write MCR[DRFEN] = 1 to enable the LFAST.
6. Write MCR[RXEN] = 1 and MCR[TXEN] = 1 to negate the LD powerdown, LR disable and LR powerdown signals.
7. Write ICR[ICLCPLD] = 31h to enable the Slaves Tx Interface.
8. Write ICR[SNDICLC] = 1 and ICR[ICLCSEQ] = 1.
9. Write MCR[TXARBD] = 0.
10. The ICLC transmission is confirmed by verifying one of the following:
    - ICR[SNDICLC] = 0.
    - TISR[TXICLCF] = 1.
11. Write ICR[ICLCPLD] = 00h to check the LFAST slaves status.
12. Write ICR[SNDICLC] = 1.
13. The LFAST slave status is confirmed by occurrence of one of the following:
    - LFAST slave is enabled if RIISR[ICPSF] = 1. Proceed to step 14.
    - LFAST slave is disabled if RIISR[ICPFF] = 1. The LFAST master must wait and restart from Step 7.

Speed mode change:

14. Write PLLCR[SWPON] = 1 to enable the LFAST masters PLL.
15. Write ICLC start PLL frame, ICR[ICLCPLD] = 02h.
16. Write ICR[SNDICLC] = 1.
17. The ICLC transmission is confirmed by occurrence of one of the following:
    - ICR[SNDICLC] = 0.
    - TISR[TXICLCF] = 1.
18. To change LFAST masters Tx interface speed both of the following operations are performed:

Note

(The slaves Rx interface speed should be changed first.)

- Write ICR[ICLCPLD] = 10h for Tx data fast frame.
- Write ICR[SNDICLC] = 1.

19. Both of the following operations are performed to change the LFAST masters Rx interface speed:

Note

(The slaves Tx interface speed mode should be changed first)

- Write ICR[ICLCPLD] = 80h to select Rx data fast frame.
- Write ICR[SNDICLC] = 1.

20. The ICLC transmission is confirmed by the occurrence of one of the following:
    - ICR[SNDICLC] = 0
    - TISR[TXICLCF] = 1.
21. Write SCR[TDR] = 1 to change the LFAST masters Tx interface speed.
22. Write SCR[RDR] = 1 to change the LFAST masters Rx interface speed.
23. Write ICR[ICLCPLD] = 00h to confirm the change in speed of the LFAST slave. This frame should be written after waiting for the expected delay in the start of the PLL and speed mode change delay at the LFAST slave.
24. Write ICR[SNDICLC] = 1
25. Write MCR[TXARBD] = 1, after some delay to ensure the step 24 ICLC frame is send but no other frame is sent to arbitation.
26. The LFAST slaves speed mode is confirmed by the occurrence of the following:
    - If RIISR[ICPSF] = 1. The LFAST slave is in High Speed mode.
    - If RIISR[ICPFF] = 1. The LFAST slave is in Low Speed mode.
27. If RIISR[ICPSF] = 1, then all frame arbitration is enabled by both of the following operations:

- Write ICR[ICLCSEQ] = 0.
- Write MCR[TXARBD] = 0.

28. If RIISR[ICPFF] = 1 modifying the LFAST slaves speed needs to be repeated by:
    - Write SCR[TDR] = 0 to return the Tx interface to Low Speed mode.
    - Write MCR[TXARBD] = 0, to enable frame transmission.
    - The operations from step 14 need to be executed again.

51.7.1.2 LFAST slave interface startup procedure

1. After reset the SLCR and RCDCR are programed according to the LVDS datasheet.
2. The PLLCR is programed with the configuration parameters for the PLL.
3. The LCR is programed with the configuration parameters for the LVDS.
4. Write MCR[MSEN] = 0. Then select LFAST modes by configuring MCR[CTSEN] and MCR[DATAEN].
5. Write MCR[DRFEN] = 1 to enable the LFAST.
6. The LR is enabled by writing MCR[RXEN] = 1. This negates the LR disable signaland LR's power down signal.
7. The LD enable signal needs to be asserted. This is done when one of the following are true:
    - When an ICLC frame with payload 31h is received the H/W will write RIISR[ICTEF] = 1 and MCR[TXEN] = 1.
    - MCRMCR[TXEN] = 1.
8. After a write to MCR[TXARBD] = 0, a ping frame will be sent on one of the following conditions:
    - When PICR[PNGAUTO] = 1 and an ICLC frame with payload 00h frame is received. H/W writes [ICRPF] = 1.
    - PICR[PNGREQ] = 1.

Speed mode change:

9. The PLL will start when one of the following conditions is met:

- When an ICLC frame with payload 02h is received. H/W writes RIISR[IPONF] = 1.
- Write PLLCR[SWPON] = 1.

10. The speed of the Tx interface is changed on one of the following conditions:
    - When SCR[DRMD] = 1 and an ICLC frame with payload 80h is received. H/W writes RIISR[ICTFF] = 1 and SCR[TDR] = 1.
    - SCR[TDR] = 1, when SCR[DRMD] = 0.
11. The speed of the Rx interface is changed on one of the following conditions:
    - When SCR[DRMD] = 1 and an ICLC frame with payload 10h is received. H/W writes RIISR[ICRFF] = 1 and SCR[RDR] = 1.
    - SCR[RDR] = 1, when SCR[DRMD] = 0.
12. A Ping Frame is sent on one of the following conditions:
    - When PICR[PNGAUTO] = 1 and an ICLC frame with payload 00h is received. H/W writes RIISR[ICRPF] = 1.
    - Write PICR[PNGREQ] = 1.
    - Write PICR[PNGREQ] = .

51.7.2 Line Receiver

This section describes the Line Receiver.

51.7.2.1 Introduction

The LR detects the voltage swing on the differential pair and converts it to a CMOS logic level that feeds the adaptive auto correlation block. The received data is sampled using the best of the 8 (default) or 4 (alternative setting) possible sampling edges from the high speed clock or 4 phases using the low speed clock. The sampling edge is chosen by checking which of the 8/4 Correlators provide the maximum correlation. If more than one sampling edge provides the maximum then the state machine will choose the sampling edge based on a defined selection algorithm.

After the correct sampling edge is chosen, the remaining unused sampling edges are turned off. Once the header is received the length of the frame (payload size) and logical channel definition can be obtained. The logical channel definition will determine the data

type of the payload and therefore will determine the destination for the payload. Decoding of the ICLC commands from the payload is required in order to extract the interface control, rate mode and PLL commands.

Figure 51-48 shows the block diagram of Uplink Controller.

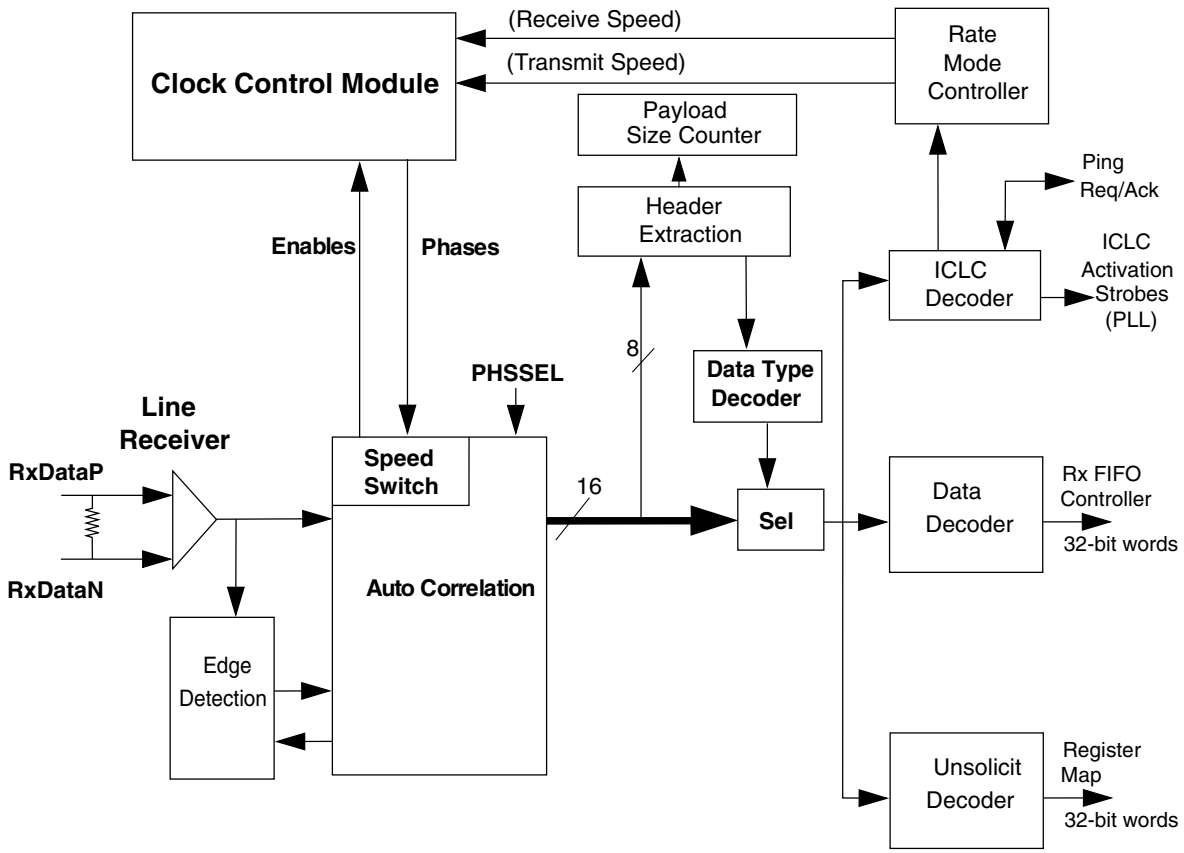


Figure 51-48. Top Level Receive Controller

51.7.2.2 Edge detection and auto-correlation

The edge detection and auto correlation are described together, as the mode setting of the auto-correlation block impacts largely on whether the edge detection circuitry is used or not. The edge detection will be performed first if required before auto-correlation.

51.7.2.2.1 Auto-Correlation modes

This section describes the Auto-Correlation modes.

51.7.2.2.1.1 Hunt Correlation mode

On reset the Receive Controller will always come up in Hunt Correlation mode. In this mode all the Correlators are enabled and the Receive Controller is always hunting for the synchronization pattern. The phase enables (either 8 or 4) to the external clock control module are always high. There is no edge detection for the first bit of the synchronization pattern. Hunt Correlation mode is considered the safest mode as the Receive Controller is always checking for the synchronization pattern, but it is the mode that consumes the most power. In Hunt Correlation mode the correlation is performed over all 16 bits of the synchronization pattern.

Figure 51-49 shows the block diagram of Hunt Correlation mode.

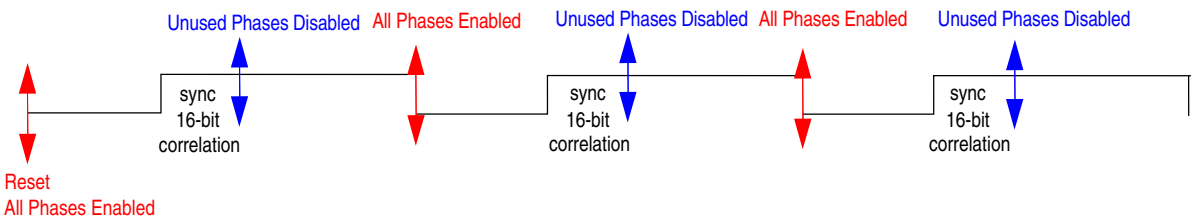


Figure 51-49. Hunt Correlation mode

51.7.2.2.2 Edge Detection

The edge detector is disabled in Hunt Correlation mode.

51.7.2.2.3 Auto correlation

The data transmission between the 2 devices LD and LR is asynchronous in nature. Hence the Receive Controller does not have the knowledge about the correct clock phase to be used for extracting the data. The task of the auto correlation (or synchronization) scheme is to estimate the best clock phase (8 or 4) for extraction of data as shown in the following figure.

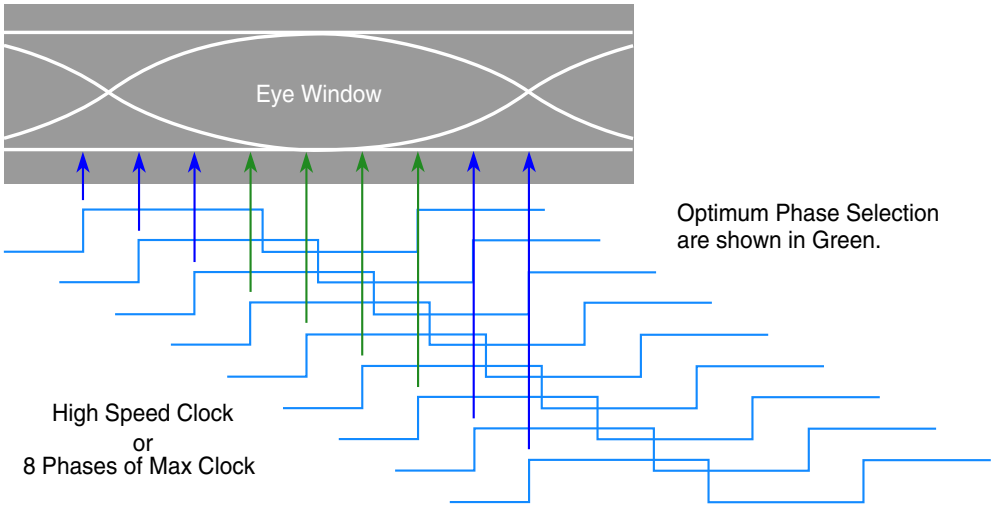


Figure 51-50. High Speed 8 Phase Selection

The objective of the auto correlation is to select the clock phase that occurs closest to the center of the eye diagram window. For 8 Phase clock alignment the worst case selection is when the clock edge is just after the LR output transition. The 8 clock phases will sample the correct value but the middle clock phases are the ideal selection. If one of the middle phases are selected then the minimum distance to the LR transition is 3 clock phases as shown in the following figure.

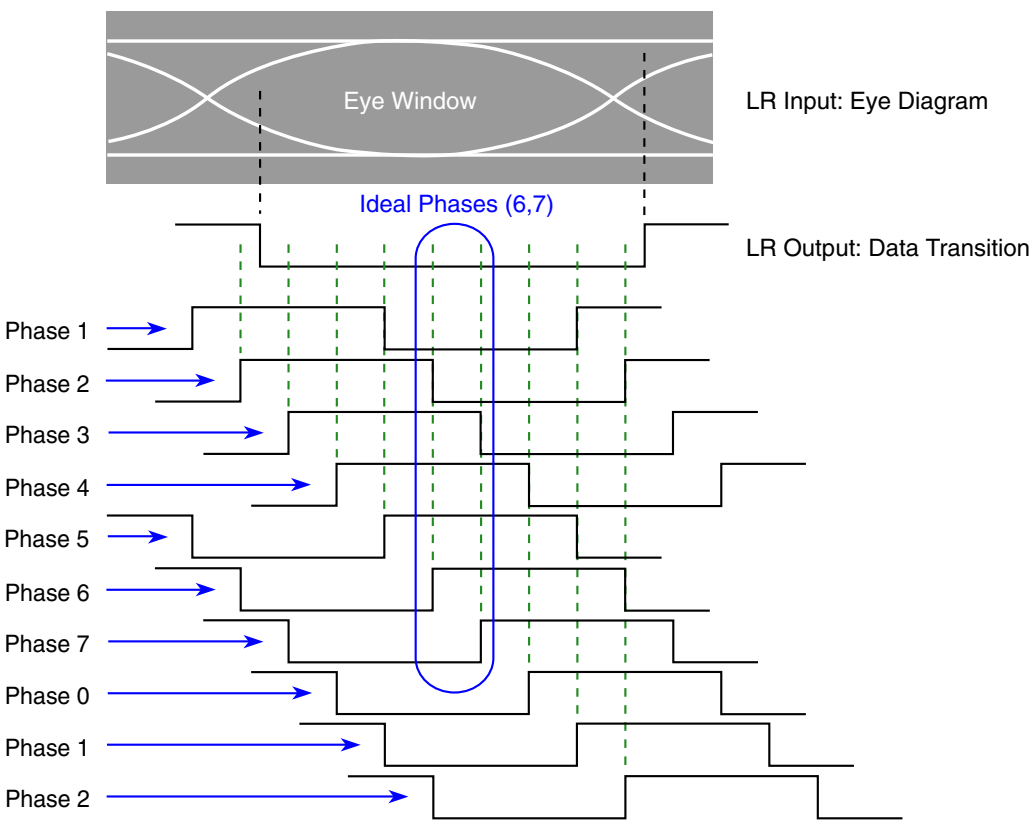


Figure 51-51. High Speed 8 Phase Clock Alignment Example

Each of the 8 high speed phases are 45 degrees separated. For 4 phases, whether high or low speed, the phases are 90 degrees separated but can have different phases enabled as shown in Figure 51-52 and Figure 51-53.

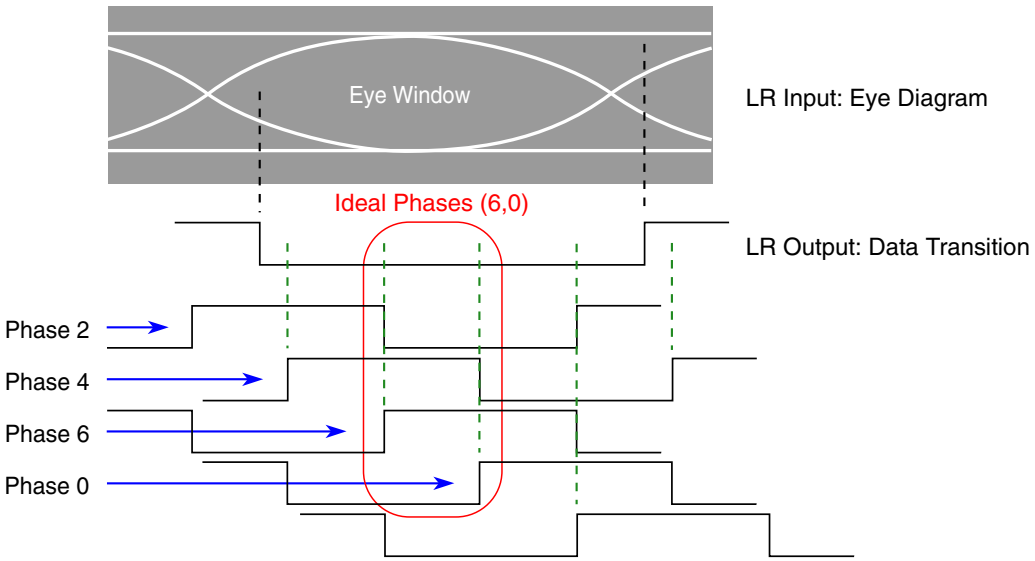


Figure 51-52. High Speed 4 Phase Clock Alignment Example

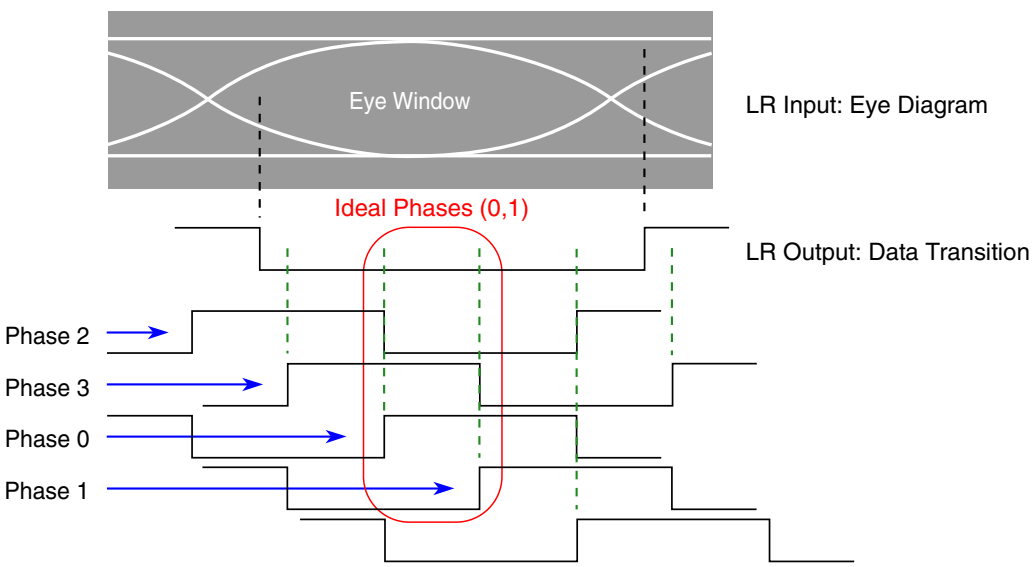


Figure 51-53. Low speed 4 phase clock alignment example

The auto correlation and selection of the correct clock phase starts after the edge detection finds a 0 to 1 transition. The high speed 8 or 4 phases from the PLL and the low speed 4 phases (generated inside the Clocking Module) are muxed inside the clocking module. The LFAST interface block will determine which phases are to be enabled and disabled. The only time the interface does not choose the phases is when the Clocking Module overrides the phase enables using COCR[SMPSEL].

The input data path can be sampled by different samplers depending on the configuration:

- High speed 8-phases: Samplers 0,1,2,3,4,5,6,7
- High speed 4-phases: Samplers 0,2,4,6
- Low Speed 4-phases: Samplers 0,1,2,3

Each sampler block samples the data path with a different clock phase from the Clocking Module, and resamples it to a intermediate phase as shown in Table 51-47, Table 51-48, and Table 51-49 before resampling it to Phase 0. The intermediate phase is used to make static timing between the initial phase and the final phase 0. The final phase, Phase 0 is chosen so all the clock trees after the sampler are clocked by the same clock.

Table 51-47. High speed 8 phase selection - sampling procedure

Samplers	InitialSample	Intermediate Sample	Final Sample
0	Phase 0	Phase 0	Phase 0
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Phase 4	Phase 2	Phase 0
5	Phase 5	Phase 2	Phase 0
6	Phase 6	Phase 4	Phase 0
7	Phase 7	Phase 4	Phase 0

Table 51-48. High speed 4 phase selection - sampling procedure

Samplers	InitialSample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0
1	Disabled	Disabled	Disabled
2	Phase 2	Phase 0	Phase 0
3	Disabled	Disabled	Disabled
4	Phase 4	Phase 2	Phase 0
5	Disabled	Disabled	Disabled
6	Phase 6	Phase 4	Phase 0
7	Disabled	Disabled	Disabled

For High speed 4 Phase selection, See Table 51-48, Samplers 1, 3, 5, 7 are disabled. In the Phase Select algorithm Phase 1 is mapped to Phase0, Phase 3 is mapped to Phase 2, Phase 5 is mapped to Phase 4, Phase 7 is mapped to Phase 6 so it simplifies the algorithm and allows the algorithm to be the same independent of 4 or 8 Phases in high speed.

Table 51-49. Low speed 4 phase selection - sampling procedure

Samplers	InitialSample Phase	Intermediate Sample Phase	Final Sample Phase
0	Phase 0	Phase 0	Phase 0

Table continues on the next page...

Table 51-49. Low speed 4 phase selection - sampling procedure (continued)

Samplers	InitialSample Phase	Intermediate Sample Phase	Final Sample Phase
1	Phase 1	Phase 0	Phase 0
2	Phase 2	Phase 0	Phase 0
3	Phase 3	Phase 0	Phase 0
4	Disabled	Disabled	Disabled
5	Disabled	Disabled	Disabled
6	Disabled	Disabled	Disabled
7	Disabled	Disabled	Disabled

For low speed 4 Phase selection, See Table 51-49, Samplers 4, 5, 6, 7 are disabled.The first four Samplers (0,1,2,3) of the low 4 phase speed selection algorithm are the same as the four samplers required for the high 8 phase speed. Therefore the same architecture can be used for both high speed and low speed with the other four data paths disabled for low speed.

51.7.2.2.4 Sampler block and phase enable and disable

There are 8 data sampler paths in total inside the auto correlation block, each data sampler has 3 sampling registers with the possibility of each register being clocked by a different phase (in example, Sampler 5 for high speed can have Phases 5, 2 and 0). Therefore, after the correct data sampler has been selected for either high or low speed, the block can turn off all the sampler paths except for one, therefore 3 out of 24 registers will remain enabled while the others are disabled, as shown in the following figure.

After correlation and the correct data sampler has been selected all the other data samplers whose initial phase does not match the select phase are disabled. The selected sampler will then keep the required phases needed enabled, this can be max 3 phases (for example, Sampler 5 has Phases 5, 2, 0) or min 1 phase (in example below, Sampler 0 has all Phase 0 content).

The end result is that the Rx Controller can disable the required number of unused sampler registers and disable any unnecessary phases from the Clocking Module by deasserting the respective phase enables.

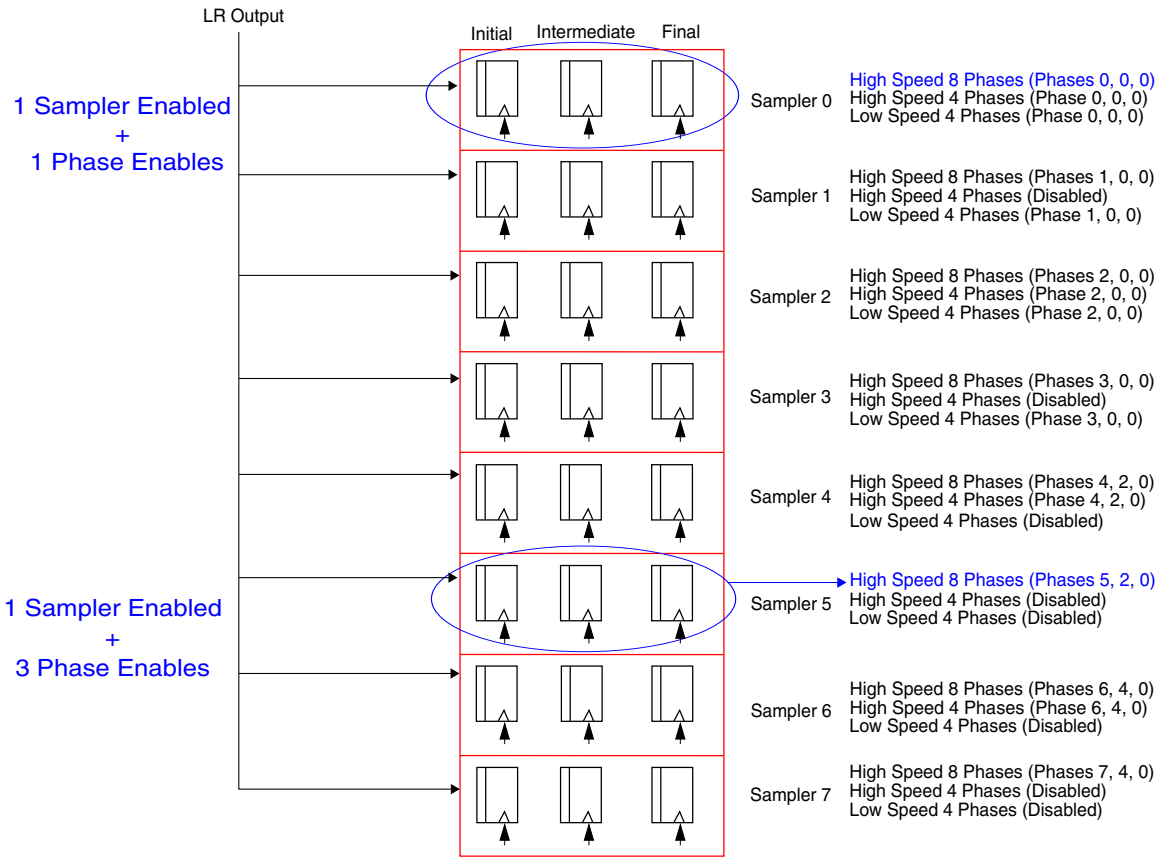


Figure 51-54. 8 Samplers, each Sampler has 3 Registers

In order to achieve the sampler and phase enable requirements each Sampler block will require the logic as shown in the following figure.

The Clock Gating Element resides inside the Clocking Module block. The interface will provide the enables to the Clocking Module and the Clocking Module will in return provide the individual clocks to the sampling registers.

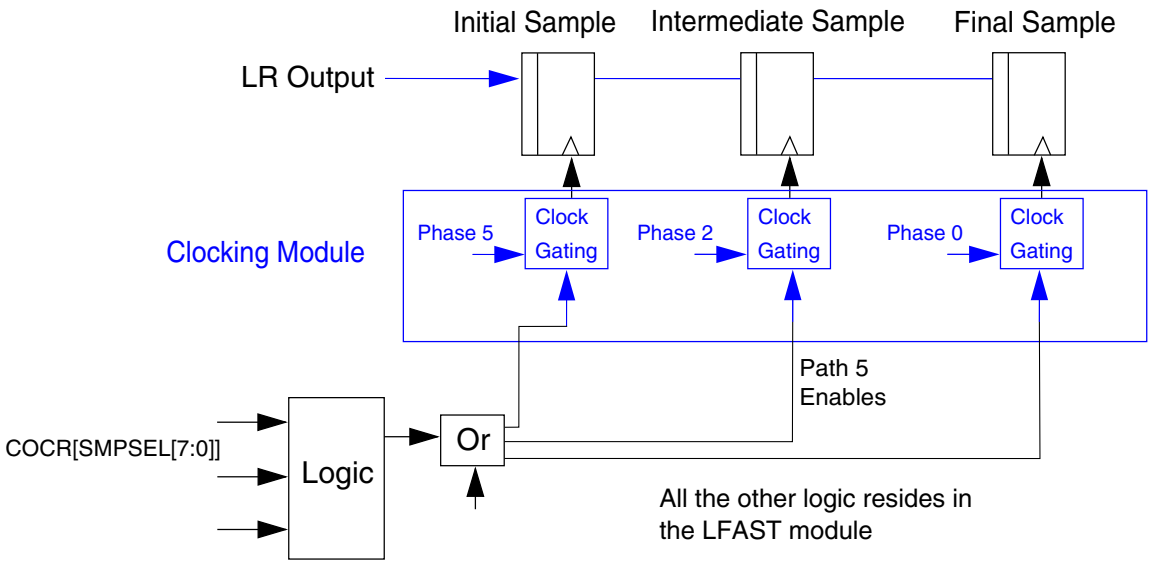


Figure 51-55. Sampler 5 Logic

51.7.2.3 Header and Payload Extraction

Once the Phases/Samplers are chosen after Synchronization and Correlation the next part of the flow is the header extraction. The Receive Controller will output a 16 bit word, bit shifted every Phase 0 clock as shown in the Figure 51-56.

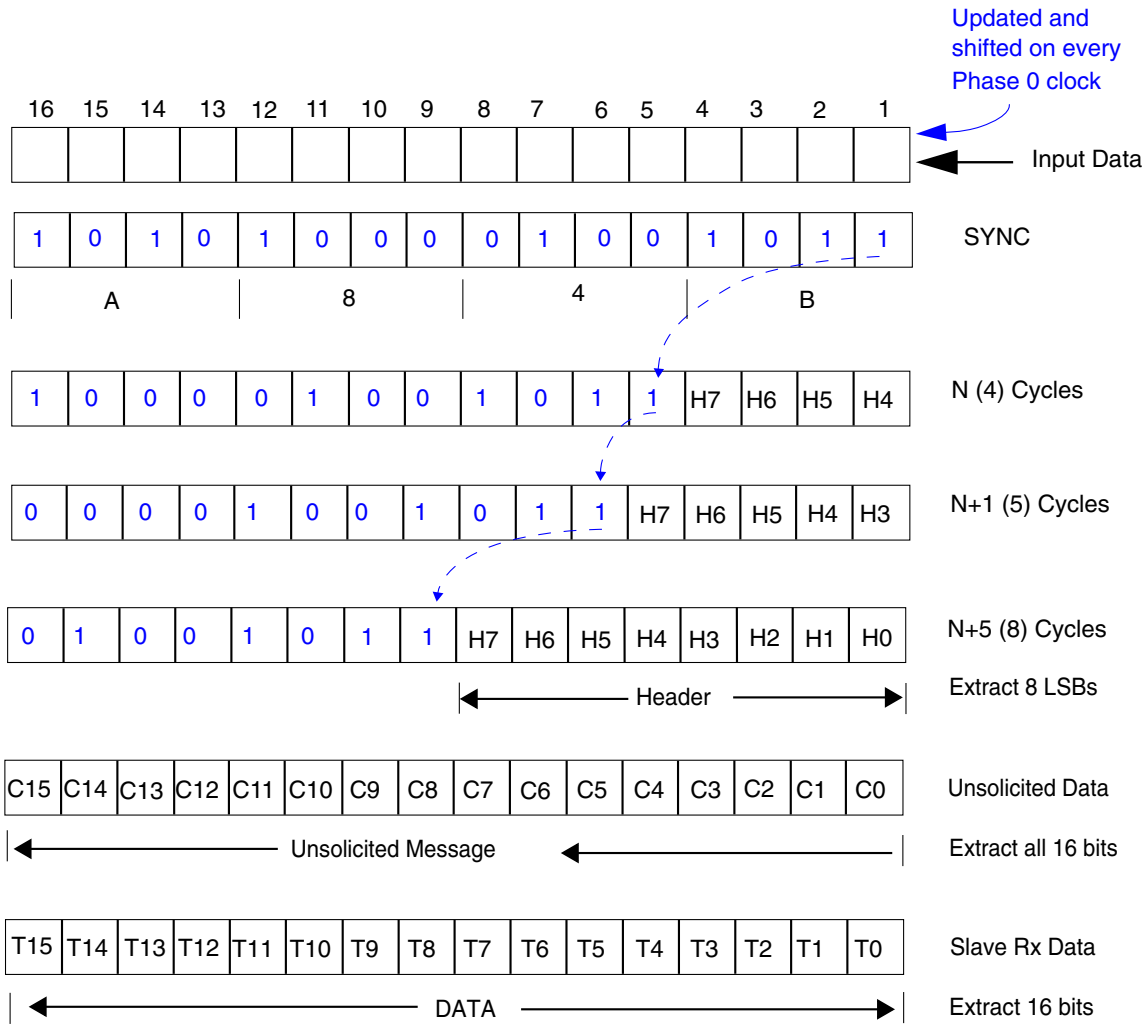


Figure 51-56. Extracting Header and Payload from the 16-bit output from Auto correlation.

51.7.3 Transmit Controller

This section describes the Transmit Controller.

51.7.3.1 Introduction

The Transmit Controller uses Rx information, status information and error control data and codes them into the appropriate frame structure for transmission to the LD. This includes the synchronization and header, in preparation for the LD, which transmits the frame to peer LFAST IC at either low or high speed. The Transmit Controller creates a frame structure that is converted to a serial format for the LD.

An arbitration block will determine which message has higher priority data, unsolicited, ICLC, CTS, ping, and so on. The data frames are extracted from the FIFO controller, the unsolicited message from the register block, the CTS messages from the Receive Controller and the ping response from the register block.

The Tx interface has two speed modes:

- Low Speed
- Fast Speed

The Transmit Controller consists of the following blocks as shown in the following figure.

- Arbitrator
- Framer
- Request Clock Control

The arbitrator will grant access to the framer from the request that has the highest priority. The framer block will extract the payload data from the granted request source and build the frame (adding synchronization or header if required). The frame is fed into a PISO (Parallel In Serial Out) and eventually sent to the LD.

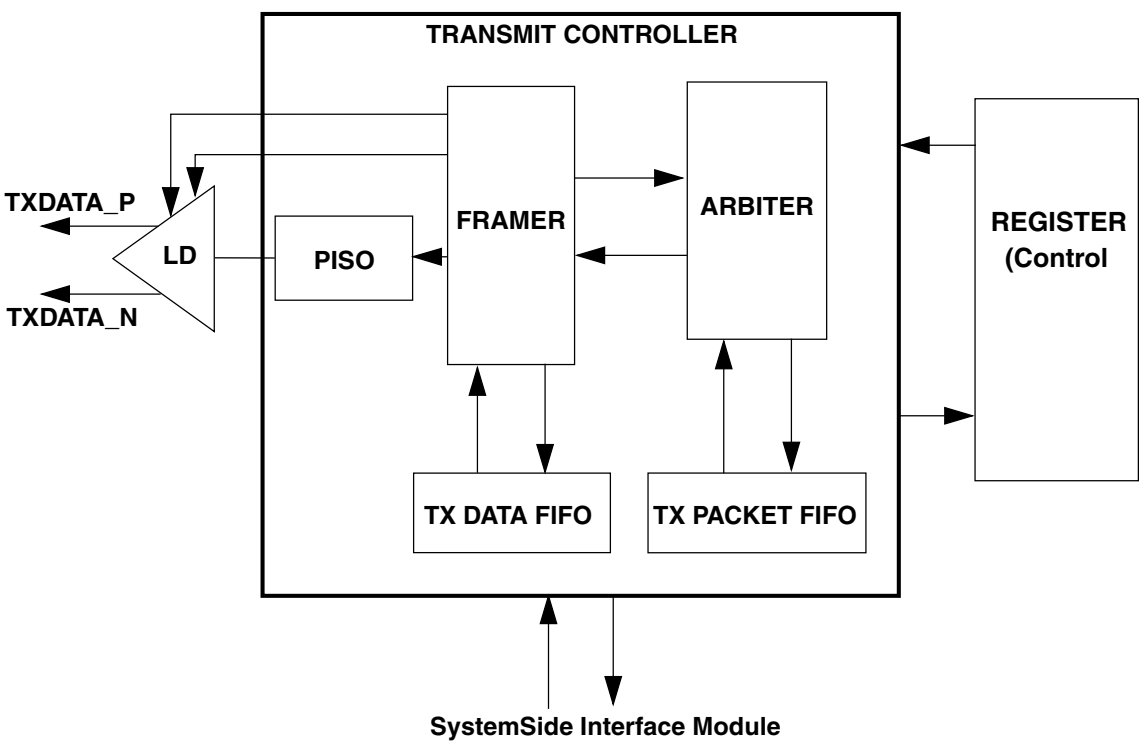


Figure 51-57. Transmit Controller Connections

51.7.3.2 Arbitration

The arbitration block prioritizes between requests from multiple sources like

- S/W programmable registers
- System side interface FIFO
- Rx interface controller

The level of priority for each request is shown in Table 51-50

Table 51-50. Priority Levels for the Transmit Controller

Request	LFAST MasterPriority	LFAST SlavePriority	Triggering Conditions *(at least one of the listed)*
LFAST interface enable	1	1	• LFAST interface enable is asserted • LFAST interface enable is negated
Tx Interface disabled	2	2	• MCR[DRFEN] = 0 • MCR[TXEN] = 0 • MCR[DRFRST] = 1 • MCR[TXARBD] = 1 • LFAST Slave: ICLC frame with payload for "Disable Rx interface" received
Tx Interface speedmode change	3	3	• SCR[TDR] is modified • LFAST Slave: ICLC frame with payload for changing Rx interface speed received
Loopback frame in Loopback mode	4	4	• TMCR[LPON] = 1Dig • RF Slave: ICLC frame with payload for loopback mode enable received **Note:** Valid only for following TMCR[LPMOD] settings: LPMOD[2:0] = 011b LPMOD[2:0] = 100b
ICLC frame request	5	-	• LFAST Master: ICR[SNDICLC] = 1 • LFAST Master: ICR[ICLCSEQ] = 1
Ping response request	-	5	• LFAST Slave: ICLC frame with payload for ping request received and PICR[PNGAUTO] = 1 • LFAST Slave: PICR[PNGREQ] = 1

Table continues on the next page...

Table 51-50. Priority Levels for the Transmit Controller (continued)

Request	LFAST MasterPriority	LFAST SlavePriority	Triggering Conditions *(at least one of the listed)*
Unsolicited frame request	6	6	• Last Frame with CTS = 1 received from the peer LFAST device • UNSTCR[USNDRQ] = 1
Data frame from Tx FIFO	7	7	• Tx FIFO contains one or more frames • Last Frame with CTS = 1 received from the peer LFAST device • MCR[DATAEN] = 1
CTSFrame	8	8	• MCR[CTSEN] = 1 • Rx FIFO reaches High/Low threshold, defined by TISR and no frame pending
Clock mode test	9	9	• TMCR[CLKTST] = 0 • LFAST Slave: ICLC frame with payload for clock test mode enable received

Once the arbitrator has granted access to a request, the Framer will commence building the frame. Any new requests for frame or data rate change will not be granted access until the framer has finished transmitting the current frame. If a data rate change request for the Tx interface is received while the Transmit controller is in the middle of sending a frame then the rate change request to the Clocking Module will be delayed. This allows for the frame to be completed before the change in speed mode. This prevents any speed mode change during the transmission of a frame. Once the speed mode request is sent to the Clocking Module by the Tx Interface Controller, it will then not allow any new requests to be processed for a specific time period defined by the bit field RCDCR[DRCNT].

51.7.3.3 Line Driver digital connections

51.7.3.3.1 Line Driver states

The LD has the following states:

- Shutdown

The LD enters the Shutdown state when the LD powerdown signal and the output buffer enable is high. In this state, the LD regulator is not supplying power and the LD outputs are connected to ground. Once LD powerdown signal is negated, a settling time is required before the LD may be used for communication.The settling time is defined by the SLCR[LWKCNT] and SLCR[HWKCNT] bitfields.

- Sleep

  The LD enters in sleep state when the signal is asserted. In this state, the LD is enabled, but held in a power-saving state. Sleep mode may be used during inter-frame gaps that are long compared to the frame durations but not long enough to allow the interface(s) or high-speed clock generators to be powered down completely. In the sleep state the LD outputs are connected to Vcm (common mode voltage). To exit from Sleep mode the LD sleep signal is negated. The LD is required to transmit a logic 0 level on the interface for a pre-defined minimum time. The minimum time is the summation of settling time of LD after negation of LD sleep signal and the wakeup time of the LR on the other side. This delay is programed in the SLCR[HSCNT] and SLCR[LSCNT] bitfields.

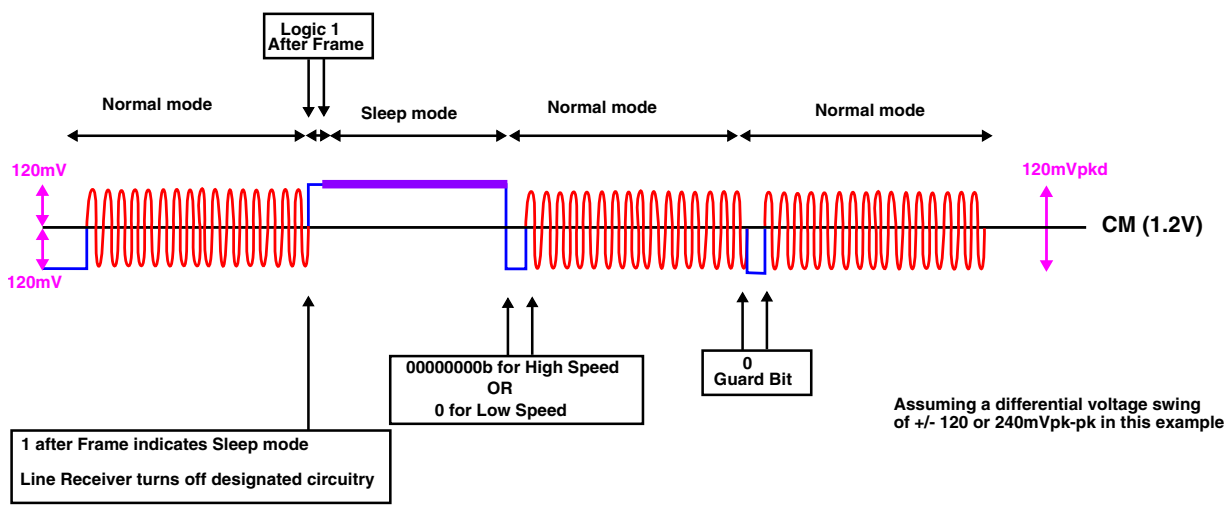


Figure 51-58. Example of Sleep mode

As shown in Figure 51-58 before every normal mode burst on the txdatap line there is one guard bit period (minimum inter-frame gap) where a logic 0 will be transmitted.

- Normal

  In the Normal state the LD is primed for transmission. The LD shall drive the interface as dictated by the input data bit that is supplied by a shift register under the control of a finite state machine. If the LD is not enabled, and the framer is activated,

and has a frame to transmit, an interrupt (TISR[TXIEF]) will be generated. The LD moves back to the shutdown state when the LFAST master sends an ICLC Rx data off command.

Table 51-51. Line Driver States

LD State	LFAST Master Triggers	LFAST Slave Triggers
Shutdown	• Programing LVDS[SWOFFLD] and LVDS[SWONLD] • Programing MCR[TXEN] and MCR[DRFEN]	• LFAST interface enable negation/assertion • Programing LVDS[SWOFFLD] and LVDS[SWONLD] • ICLC command from LFAST master • Programing MCR[TXEN] and MCR[DRFEN]
Sleep	• Programing LVDS[SWSLPLD] and LVDS[SWWKLD]	• No pending frame for transmission • Programing LVDS[SWSLPLD] and LVDS[SWWKLD]
Normal	Absence of above mentioned conditions	Absence of above mentioned conditions

51.7.4 CTS mode support

LFAST module supports flow control methods. The CTS (Clear to send) is a protocol defined method to ensure no loss of frame, due to Rx FIFO unavailability. The optimal use of bandwidth of LFAST, without losing frames, is ensured by proper programing of the Rx FIFO Lower threshold (RFCR[CTSMN]) and Higher threshold (RFCR[CTSMX]). The CTS is supported by both the LFAST Master and Slave.

CTS Transmission:

The LFAST device sends CTS information with each frame to the peer LFAST device.

If the CTS mode is enabled by write MCR[CTSEN] = 1 then:

- Sends CTS bit as 0 in LFAST frame (bit[0] of Header field) whenever the Rx FIFO reaches the higher threshold defined by the bitfield TISR[CTSMX]. This indicates that the LFAST device doesn't want the peer device to send data. The CTS bit of all frame are sent 0 untill the Rx FIFO pointer reaches the lower threshold, as described below.

- Sends CTS bit as 1 in LFAST frame (bit[0] of Header field) whenever the Rx FIFO reaches the Lower threshold defined by the bitfield TISR[CTSMN]. This indicates that the LFAST device is ready to receive data from the peer device. The CTS bit of all frame are sent 1 untill the Rx FIFO pointer reaches the Higher threshold, as described above.
- The CTS bit can be sent through any type of frame (data, unsolicited or ICLC). When there are no frames available in LFAST to send to the peer device, a CTS frame is sent. In this frame, bit[4-1] of Header field are sent as 0011b and 8-bit payload of 0.

If the CTS mode is not enabled (for example, bit MCR[CTSEN] = 0) then:

- All the frame sent will have CTS bit as 1 in LFAST frame (bit[0] of Header field).

CTS Reception:

The reception of a LFAST frame with CTS bit 0 indicates that the peer device is not ready to receive data frames. The transmission of all pending data and unsolicited frames are postponed untill a frame with CTS bit 1 is received.

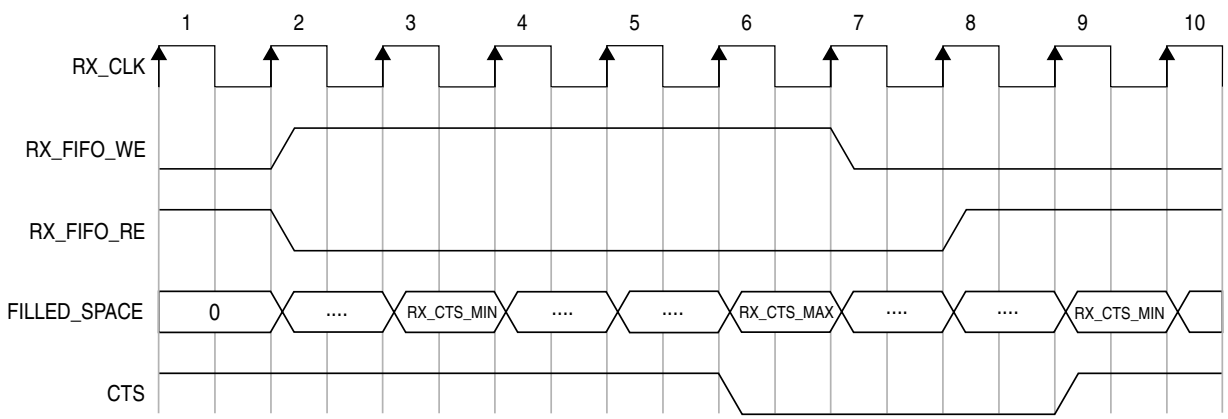


Figure 51-59. CTS generation

51.7.5 Frames supported

Table 51-52 provides the frames supported and their permissible payload sizes

Table 51-52. Frames supported by LFAST interfaces

Frame Type	LCT Code	Supported by 1. LFAST Master Tx2. LFAST Slave Rx	Supported by1. LFAST Slave Tx2. LFAST Master Rx	Payload Size (bits)
Data Frame	0100b – 1011b	YES	YES	32, 64, 96, 128, 256, 288
Unsolicited Frame	0001b	YES	YES	8, 32, 64, 96, 128, 256 and 288
ICLC Frame	0000b	YES	NO[1]	8
CTS Frame	0011b	YES	YES	8
Reserved	All others	—	—	—

1. Except the ICLC PING response frame.

If logical channel type (LCT) code other than the above mentioned are received then the interrupt flag RISR[RXLCEF] = 1. If the payload size received does not match any value in Table 51-52 then the flag RISR[RXSZF] = 1. In both these error conditions the received frame is ignored. The S/W may have to take the necessary steps to recover from such an error.

51.7.6 Frame flow

Following sections describe Data Frame Flow, ICLC Flow and Rx Unsolicited Data Flow.

51.7.6.1 Data flow

LFAST supports the transfer of data between master and slave LFAST devices.

51.7.6.1.1 Data transmit

System Side Module is the initiator of all the Tx data frame to LFAST peer device. Data frame transmit is triggered whenever the Tx Data FIFO has at least one valid frame. The MCR[DATAEN], MCR[DRFEN] and MCR[TXEN] bits, should be set to enable the transfer of Tx data.

If MCR[DATAEN] = 0, then the valid frames in the in Tx data FIFO will be ignored for transmission. The Payload Size and channel type of each frame is specified by the System Side Module interface during transfer of the frame to the LFAST interface. The frame header is stored in the Tx packet FIFO and the payload in the Tx data FIFO.Whenever the Tx FIFO has at least one frame then a data transmit request is made to the Tx arbiter of the LFAST. Tx block arbitrates the data transmit request and

schedules it depending on the priority of all the pending transmit requests. When data request is scheduled by Tx block, the required data is fetched from Tx data FIFO. The number of frames present in the Tx FIFO for transmission is indicated by the bitfield DFSR[TXFCNT].

51.7.6.1.1.1 Programing model for Tx data transmit

1. Program MCR[DATAEN] = 1, MCR[TXEN] = 1 and MCR[DRFEN] = 1 to enable the Tx path of LFAST device
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR] bits.
3. Frame present in TX FIFO (Data and Packet FIFO) are sent.
4. TISR[TXDTF] = 1 after each frame transfer
5. CTS is set depending on the push/pull mode setting defined by MCR[CTSEN].

51.7.6.1.2 Data receive

LFAST master and slave supports reception of data frame. The received frame is determined to be of data frame type by decoding channel type field of the header present in the received frame. The MCR[DATAEN], MCR[RXEN] and MCR[DRFEN] bits should be set to enable the data frame reception.When MCR[DATAEN] = 0 the received data frames will be ignored and will not be placed in the Rx data FIFO.

The Rx data frames received by Rx block are stored in the Rx FIFO. Whenever the frame is received in the Rx FIFO the System Side Module is indicated by assertion of LFAST Rx FIFO ready signal. The frame size and the Channel type is passed to the LFAST. The frame boundaries are indicated by start of frame and end of frame signals. The number of unread frames in the Rx Data FIFO are indicated by DFSR[RXFCNT]. When Rx DATA FIFO is full and cannot accommodate current frame completely, then the remaining data of the Rx frame is discarded. In this case, UNSRSR[RXOF] = 1 and an interrupt is generated if the RIER[RXOFIE] = 1.

51.7.6.2 Unsolicited flow

51.7.6.2.1 Unsolicited frame transmit flow

The S/W is the initiator for unsolicited frames to the LFAST peer device. The unsolicited frame header and payload is programed into the Unsolicited Data and Control registers. Once the Payload is programed UNSTCR[USNDRQ] is set, generating a request for unsolicited frame transfer to the Tx arbiter.

51.7.6.2.1.1 Programing model for unsolicited frame transmit

1. Program MCR[TXEN] = 1 and MCR[DRFEN] = 1 in the Mode Configuration Register (MCR) to enable the Tx path
2. Select the desired clock rate at which data should be transmitted, using ICLC transfers and appropriate programing of SCR[TDR].
3. Read the UNSTCR[USNDRQ].
   - If UNSTCR[USNDRQ] = 1 then wait for either of the following:
     - UNSTCR[USNDRQ] = 0
     - TISR[TXUNSF] = 1
   - If UNSTCR[USNDRQ] = 0 then:
     - Program the unsolicited payload in UNSTDR0–UNSTDR8, and can be written up to a payload of frame of a maximum of 288 bits.
     - Program the unsolicited frame header in UNSTCR[UNSHDR].
     - Program UNSTCR[USNDRQ] = 1.
4. UNSTCR[USNDRQ] is cleared by the Tx Block after the frame transfer.
5. TISR[TXUNSF] = 1 when the frame is transmitted.
6. CTS is set depending on the Push-Pull mode setting defined by MCR[CTSEN].

51.7.6.2.2 Unsolicited frame receive flow

Whenever an Unsolicited frame is received from the peer device, the following steps are performed:

If UNSRSR[URXDV] = 0 then:

1. The payload size field of the header is first saved into the UNSRSR[URPCNT].
2. The payload is stored in the UNSTDR0–UNSTDR8.

3. UNSRSR[URXDV] = 1 and the RISR[RXUNSF] = 1 indicating the successful reception of the frame.

If UNSRSR[URXDV] = 1 then:

1. The current unsolicited frame is ignored.
2. RISR[RXUOF] = 1.

Typical steps by the processor after RISR[RXUNSF] = 1 is as follows:

1. The processor reads UNSRSR to get the payload size of the frame.
2. It then reads the complete frame by reading UNSRDR8–UNSRDR0 for the payload size as received in the frame.
3. Then it clears the interrupt (write RISR[RXUNSF] = 1) and also the UNSRSR[URXDV] bit.

51.7.6.3 ICLC flow

The ICLC (Interface Control Logical Channel) is a separate logical channel type, which is mainly meant for implementing the data rate change in the LFAST interface and initiating the test modes.

51.7.6.3.1 ICLC data transmit flow

Whenever the processor intends to transfer an ICLC frame to the LFAST slave then the following steps are to be followed.

1. Write the ICLC frame payload in the ICR[ICLCPLD] (see Table 51-53 for payloads as defined by the standard).
2. Program ICR[SNDICLC] = 1 to initiate the ICLC frame transfer. This bit clears itself when the ICLC frame has been transmitted. The processor needs to poll this bit to make sure it is cleared before setting this bit again (even when ICR[ICLCSEQ] = 1).
3. TISR[TXICLCF] = 1 after the transfer of the ICLC frame.
4. CTS is set depending on the Push-Pull mode setting defined by MCR[CTSEN].

To determine whether the ICLC frame has been sent or not, either the ICR[SNDICLC] bit can be polled, or wait for TISR[TXICLCF] = 1.

Table 51-53. Supported ICLC Payloads

ICLC Code(hex)	ICLC Function
00	Ping Request from LFAST Master to LFAST Slave
01	Reserved
02	Start PLLIn preparation for High Speed mode
04	Stop PLL Fallback to low speed mode
08	Select TxData Slow (LFAST Slave Rx Interface)
10	Select TxData Fast (LFAST Slave Rx Interface)
20	Select RxData Slow (LFAST Slave Tx Interface)
40	Not Supported
80	Select RxData Fast (LFAST Slave Tx Interface)
31	Enable RxData Interface (LFAST Slave Tx Interface)
32	Disable RxData Interface (LFAST Slave Tx Interface)
34	Clock Test mode Send 101010… continuously using the currently configured clock rate (slow, fast);
FF	Turn payload loopback on
38	Turn Test mode (Loopback and Clock Test) off

Programing model for ICLC frame transmit:

1. Set the ICR[ICLCSEQ] bit (optional)
2. Write the ICLC frame payload in the ICR[ICLCPLD], corresponding to the desired Tx/Rx data rate change, as described in the Table 51-53.
3. Write ICR[SNDICLC] = 1 to initiate the ICLC frame transfer.
4. LFAST master will schedule the ICLC transfer. Data present in ICR[ICLCPLD] is transmitted to the LFAST slave at the old Tx data rate. LFAST slave will configure its Tx/Rx interface data rate accordingly.
5. Reset the ICR[ICLCSEQ] to allow the scheduling of other types of frames, which will be sent at the new data rate.

When ICR[ICLCSEQ] = 0, though the ICLC frame transfer still has highest arbitration priority, other frames may be scheduled if no valid ICLC frame request exist.

Note

ICLC frame transmit will not trigger the internal data rate change of LFAST Master Tx/Rx interface. LFAST master Tx/Rx interface data rate will only be changed, when corresponding SCR[TDR] and SCR[RDR] bits are appropriately configured.

Processor requires to take care that no data frames are to be transmitted while data rate change is happening or ICLC is transmitting frames by setting ICR[ICLCSEQ] bit in Table 51-53.

51.7.6.3.2 ICLC data receive flow

When the bits 4 to 1 of a receive frame are 0000b it indicates that the payload is an ICLC. ICLC payloads are always 8 bits.

The reception of an ICLC frame is indicated by an interrupt to the system. The ICLC status register RISR indicates the type of the ICLC frame received. The Rx block will decode and generate the appropriate signals. An invalid ICLC code besides the ones listed in Table 51-53 will cause RISR[RXICF] = 1.

51.7.6.3.2.1 Ping request ICLC

The LFAST slaves ICLC decoder will decode the ping request and set RIISR[ICPRF]. The S/W can also write to PICR[PNGREQ] to indicate to the Tx block that a ping response frame needs to be transmitted. The PICR[PNGAUTO] indicates whether Tx block can respond automatically to the request by the LFAST master ICLC Ping Request frame. The Tx block will arbitrate the ping response frame request and send a ping frame with ping data defined by bitfield PICR[PNGPLYD]. Once the ping response frame has been sent the Tx block will set TISR[TXPNGF] and clear PICR[PNGREQ], if set.

51.7.6.3.2.2 LFAST Slaves RxData Interface Slow/Fast ICLC

The LFAST Slaves Rx Interface has two speed modes supported: slow (Low speed) and fast (High speed). The ICLC command will write RIISR[ICRSF] = 1 (Low speed) or RIISR[ICRFF] = 1 (High speed).

51.7.6.3.2.3 LFAST Slaves TxData Interface Slow/Fast ICLC

The LFAST slaves Tx Interface Controller has two speed modes: slow (Low speed), and fast (High speed). The ICLC command will write RIISR[ICTSF] = 1 (Low speed) or RIISR[ICTFF] = 1 (High speed).

51.7.6.3.2.4 Enable/Disable LFAST Slaves TxData Interface ICLC

The ICLC commands, Enable RxData Interface and Disable RxData Interface are decoded and the appropriate enable/disable signal sent to the Tx block Controller. These ICLC command writes RIISR[ICTEF] = 1 and RIISR[ICTDF] = 1. The ICLC Tx enable command will write MCR[TXEN] = 1.

No frames can be sent on the LFAST Slave Tx interface until the either LFAST master has enabled it via an ICLC frame or S/W programs MCR[TXEN] = 1.

51.7.6.3.2.5 Clock Test mode ICLC

This ICLC command is decoded, and then is used to write TMCR[CLKTST] = 1. This indicates to the Tx interface to output an alternating pattern of 1 and 0 at the currently configured clock rate. The Rx interface generates RIISR[ICCTF] = 1 on reception of this ICLC command.

The exit from this mode happens on reception of Test mode off ICLC command.

51.7.6.3.2.6 Loopback payload on ICLC

This ICLC command is decoded and TMCR[LPON] = 1 to indicate to the Tx Block that the received payload is to be sent back. The exit from this mode happens on reception of Test mode off ICLC command. The Rx interface causes RIISR[ICLPF] = 1 on reception of this ICLC command.

51.7.6.3.2.7 Test mode off ICLC

This ICLC command is decoded and TMCR[LPON] and TMCR[CLKTST] are cleared to indicate to the Tx block to exit from loopback mode or clock test mode.

Another option is for the payload loopback option to remain enabled until negated on the toggling of LFAST interface enable.

51.7.6.3.2.8 Ping response ICLC

The LFAST master considers any ICLC frame payload as ping response data. The LFAST masters Rx block will store the received ping data in PISR[RXPNGD] and match it with PICR[PNGPYLD]. RIISR[ICPSF] = 1 if the Ping received matches with the one stored in PICR[PNGPYLD], or RIISR[ICPFF] = 1.

51.7.6.4 CTS flow

51.7.6.4.1 CTS Tx flow

The CTS frame, if enabled by the S/W, indicates the status of the Receive Data FIFO. CTS frame is triggered whenever Rx Data FIFO reaches either the High/Low threshold and no data, ICLC or unsolicited frame is ready for transmission.

Programing model for CTS frame transmit:

- Program RFCR[CTSMX] and RFCR[CTSMN].
- Enable the CTS frame Transmission by programming MCR[CTSEN] = 1.
- Program MCR[TXEN] = 1 and MCR[DRFEN] = 1 to enable the Tx path.
- The CTS bit sent with a valid frame (for example, data, ICLC and unsolicited) is 1 until the Rx Data FIFO reaches High Threshold.
- When the Rx Data FIFO reaches higher threshold and no valid frame (data, ICLC and unsolicited) is pending for transmission, then the CTS frame is triggered, with CTS bit of header = 0 and the status bit RISR[RXMXF] = 1.
- When the Rx Data FIFO reaches Low Threshold, due to system side reads and no valid frame (data, ICLC and unsolicited) is pending for transmission, then the CTS frame is triggered, with CTS bit of header = 1, and status bit RISR[RXMNF] = 1.

51.7.6.4.2 CTS Rx flow

Whenever a CTS frame header or any other frame with valid header is received with CTS bit = 0, then RISR[RXCTSF] = 1. The Tx Interface arbitration for unsolicited frame and data frame transmit request is turned off on reception of frame with CTS bit 0. The arbitration for unsolicited frame and data frame is enabled on reception of frame with CTS bit 1. Frame with LCT type CTS are not stored in the LFAST.

51.7.7 Test and Debug Support

The test and debug interface helps to debug the LFAST module. These signals can be brought out for ease of validation.

51.7.7.1 Loopback Test mode

The loopback function allows to verify the correct operation of the physical interface and the basic checks for the LFAST module without a peer device.

There are certain prerequisites before entering the Loopback mode:

- LD and LR should be turned on.
- Tx and Rx mode should be enabled.
- Interrupts needed for S/W should be enabled.
- Both Tx and Rx interface should be in same speed mode.
- For Automatic Test mode TMCR[LPFRMTH] should be programed.

The LFAST module supports four loopback modes, defined by TMCR[LPMOD].

Table 51-54. Loopback modes

LPMOD[2:0]	Mode Selected
000	Rx Loopback (default)
001	Rx LVDS Loopback
010	Tx Loopback without Automatic frame generation
011	Tx Loopback with Automatic frame generation
100	Tx LVDS Loopback (external) with Automatic frame generation

The TMCR[LPON] bit controls the state of the loopback function, and the default setting is 0 (off). This can be written by S/W, or in the case of the LFAST slave, the TMCR[LPON] bit can be modified by the Rx interface controller on decoding of a valid ICLC loopback on or Test mode off commands. The LPMOD should not be changed when the LPON bit is set.

In all loopback modes, except Tx loopback without automatic frame generation, the decoding of frame and error flags is stopped. Only decoding of following is done:

- ICLC Turn Test mode Off frame.

- CTS bit of all frames.
- Automatic Loopback frame decoding (only in Automatic frame generation mode).

51.7.7.1.1 Rx Loopback mode

For Rx loopback mode the output of the LR is passed to the LD with manipulation via the Rx and Tx Interface controllers. CTS bit (Bit 0) of the header is guaranteed to be asserted when the incoming data from the other device is looped back. In Rx Loopback mode the Rx Interface controller operates in normal mode, decoding the frames (header, payloads). This allows the LFAST Master to control the Loopback mode on and off by sending ICLC commands.

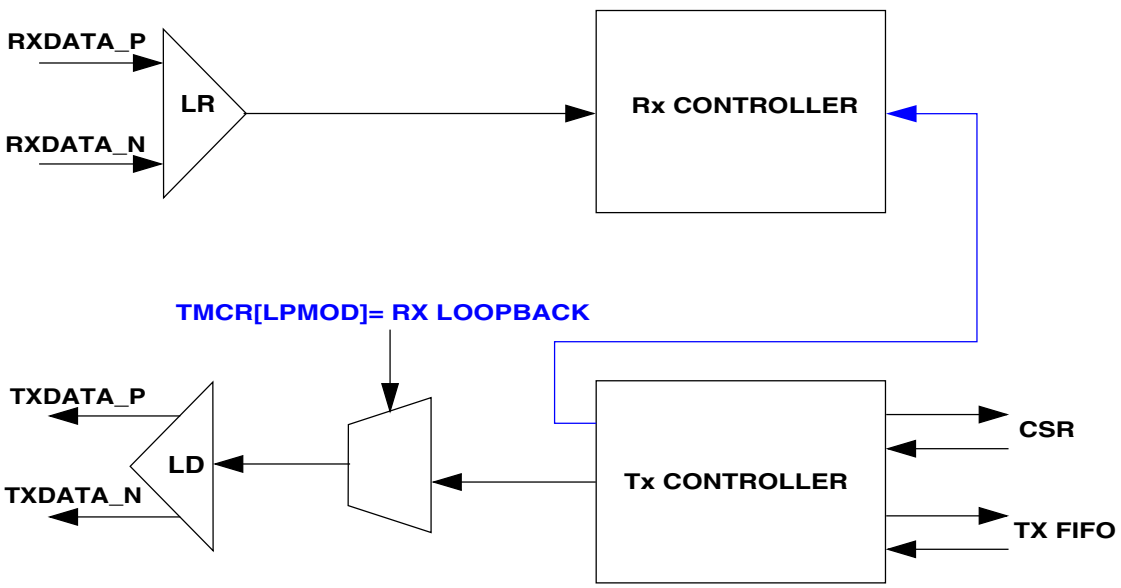


Figure 51-60. Rx LoopBack mode

Entry to Rx Loopback mode:

1. S/W programs TMCR[LPMOD] = 000b.
2. Loopback can be turned on by either of the following methods:
   - S/W programs TMCR[LPON] = 1.
   - For Slave Only: - Reception of ICLC Frame with Payload FFh (Loopback mode on), from LFAST Master

Exit from Rx Loopback mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0
- For LFAST Slave: - Transmission of ICLC Frame with payload 38h (Test mode off), from LFAST Master

The peer LFAST device is required to maintain at least 2-bit IFG between two Loopback frames in this mode.

51.7.7.1.2 Rx LVDS LoopBack mode

This loopback mode is provided to verify and characterize the LVDS pads. In this loopback mode the data received by LFAST on Rx LVDS input is loopback to Tx LVDS output, bypassing LFAST. For LVDS loopback the output of the LR is passed to the LD via "Rx LVDS LoopBack Mux". Bit 0 of the header cannot be guaranteed to be asserted when the incoming data from the other device is looped back. In this loopback, the Rx Interface Controller operates in normal mode, decoding the frames (header, payloads) but the Tx Interface Controller is ignored.

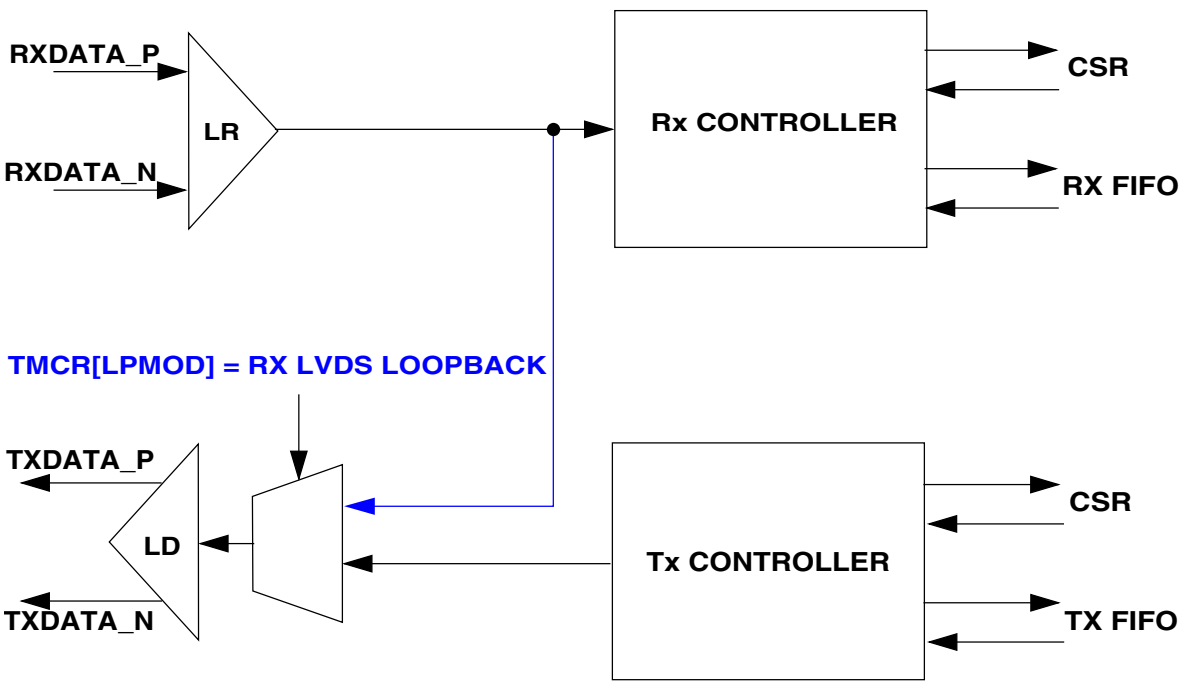


Figure 51-61. Rx LVDS LoopBack mode

Entry to Rx LVDS Loopback mode:

1. S/W programs TMCR[LPMOD] = 001b.
2. Loopback can be turned ON by either of the following methods:

- S/W writes TMCR[LPON] = 1.
- For Slave Only: - Reception of ICLC Frame with Payload = FFh (Loopback mode ON), from LFAST Master.

Exit from Rx LVDS Loopback mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave:
  - Transmission of ICLC frame with payload 38h (Test mode off), from LFAST Master.

51.7.7.1.3 Tx loopback mode without automatic frame generation

This loopback mode is provided to verify the LFAST functionality, if LVDS pads are not functional. In this loopback mode the data transmitted by LFAST on Tx LVDS output is loopbacked internally on Rx LVDS input, bypassing LVDS pads.

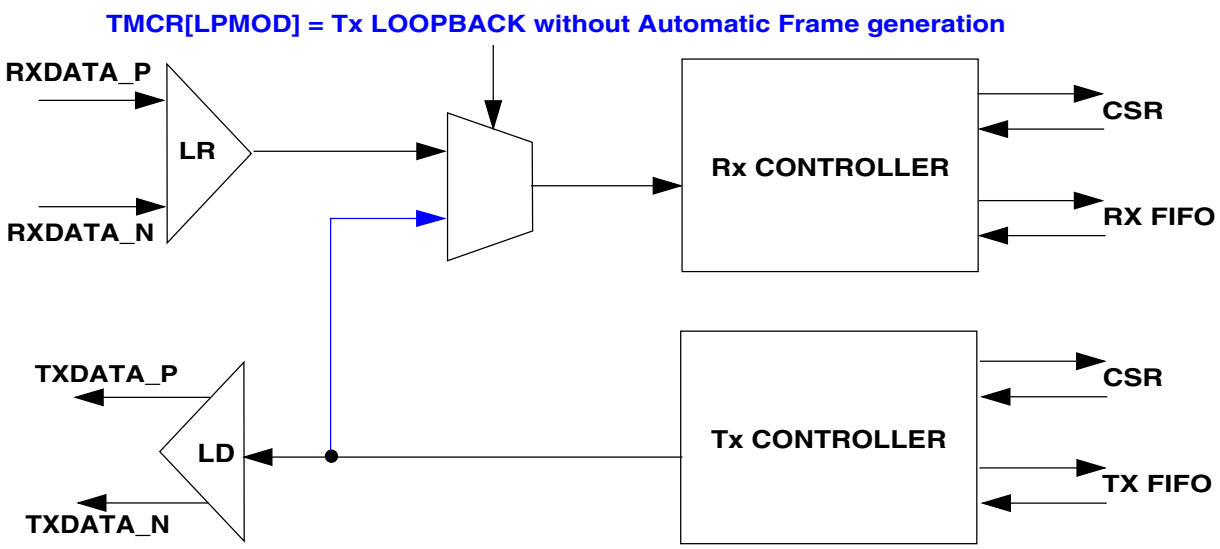


Figure 51-62. Tx LoopBack mode without automatic frame generation

Entry to Tx Loopback without Automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 010b.
2. Loopback can be turned on by either of the following methods:
   - S/W programs TMCR[LPON] = 1.
   - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx Loopback without Automatic frame generation mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 38h (Test mode off), from Tx interface (using unsolicited frame).

All frames and error flags are decoded in this mode.

51.7.7.1.4 Automatic frame generation

The LFAST module supports two Loopback modes where pre-defined frames are generated automatically by the Tx Interface and Rx Interface indicates its successful reception by dedicated signals and status registers. These modes are defined for BIST like check for the LFAST, with minimal S/W intervention.

The Control register used for these modes are:

- ALCR[LPCNTEN] defines whether fixed number of auto loopback frames to be transmitted
- ALCR[LPFMCNT] defines the number of loopback frames to be transmitted if ALCR[LPCNTEN] = 1.

The Status signals and register used for these modes are:

- GSR[LPCSDV]: Valid Synchronization received.
- GSR[LPCHDV]: Frame with Header of 13h received.
- GSR[LPCPDV]: Frame with Payload of CBh received.
- GSR[LPTXDN]: Number of Auto Loopback frame transmitted with valid Synchronization, Header (13h) and Payload (CBh) is equal to ALCR[LPFMCNT].

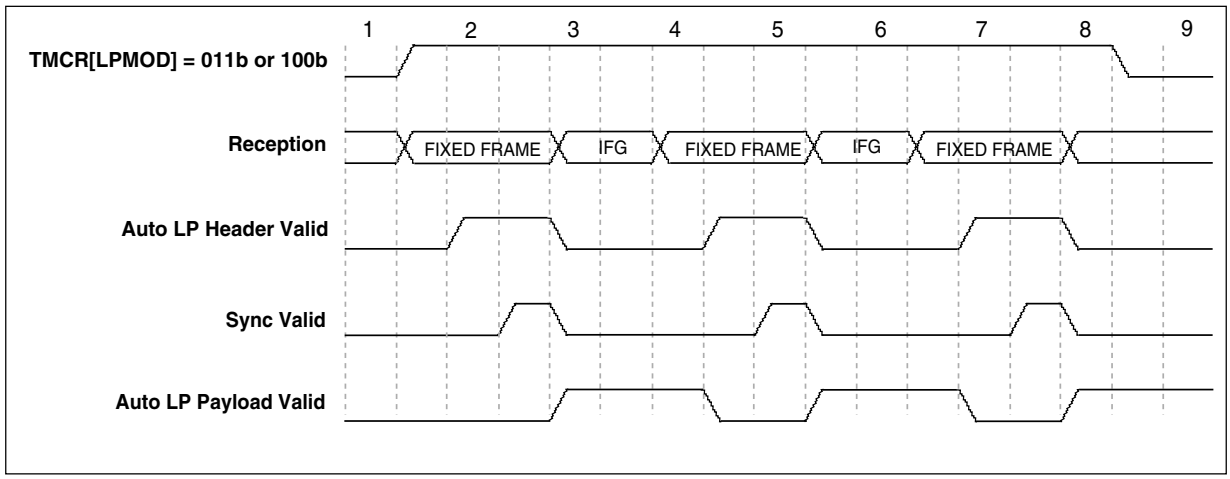


Figure 51-63. Automatic Loopback Test status signal timings

The two Loopback with automatic frame generation modes are:

1. Tx Loopback with Automatic frame generation
2. Tx LVDS (external) with Automatic frame generation

51.7.7.1.4.1 Tx loopback mode with automatic frame generation

When TMCR[LPMOD] = 011b (Tx Loopback mode with Automatic frame generation) the frames are generated by the Tx Interface. This mode helps to validate the Tx and Rx Path with minimal intervention from the S/W. The frames generated have fixed header of 13h and payload of CBh. The successful reception of this frame is indicated by the status signals and registers.

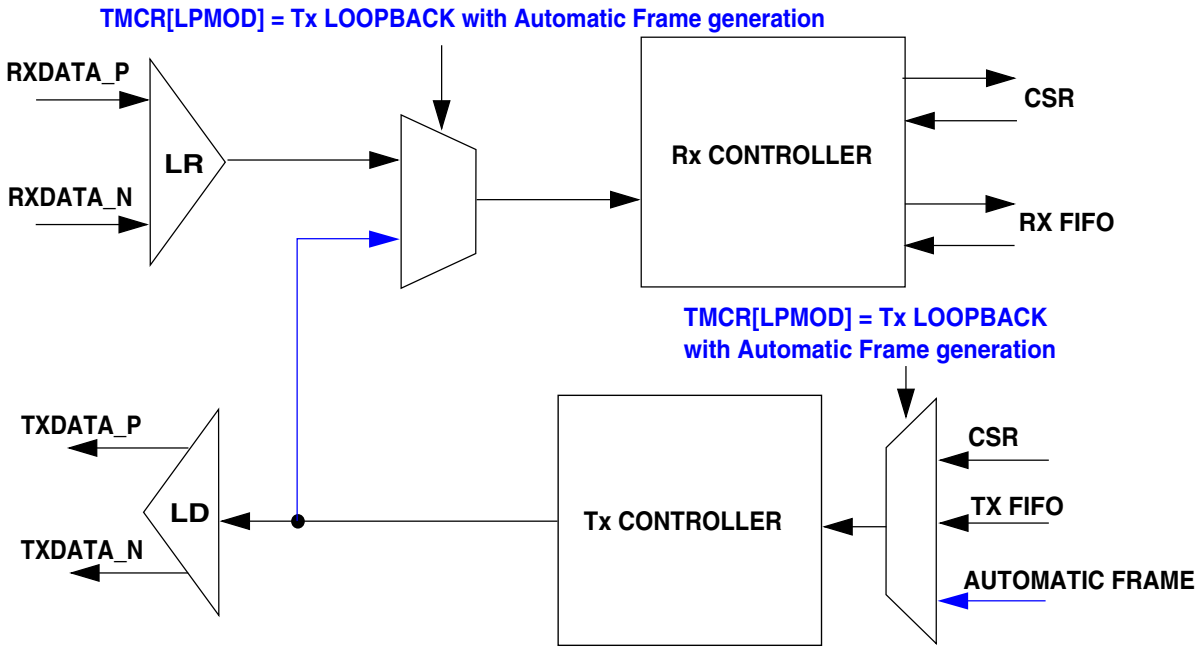


Figure 51-64. Tx LoopBack mode with automatic frame generation

Entry to Tx Loopback with automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 011b.
2. Loopback can be turned on by either of the following methods:
   - S/W programs TMCR[LPON] = 1.
   - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx Loopback with Automatic frame generation mode can be done by any of the following methods:

- S/W programs TMCR[LPON] = 0.

51.7.7.1.4.2 Tx LVDS loopback (external) mode with automatic frame generation

In this mode the LD/Tx Controller is used to test the LR/Rx Controller. The idea is to use a test frame (predefined) from the Tx Controller out through the LD, loopback completed on the DUT-board (LD connected to LR via a transmission line), back into the LR, synchronized and correlated in the Rx Controller and compared to what was sent in the Downlink controller.

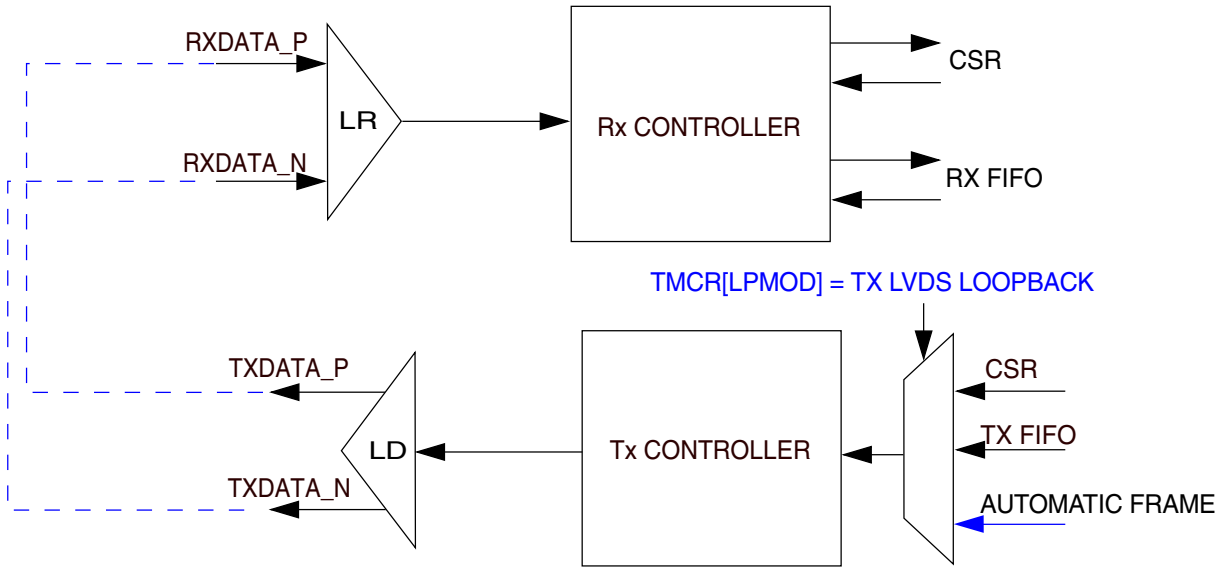


Figure 51-65. Tx LVDS loopback (external) mode with automatic frame generation

Entry to Tx LVDS loopback with automatic frame generation mode:

1. S/W programs TMCR[LPMOD] = 100b.
2. Loopback can be turned on by either of the following methods:
   - S/W programs TMCR[LPON] = 1.
   - For Slave Only: Reception of ICLC frame with payload FFh (Loopback mode on), from LFAST Master

Exit from Tx LVDS loopback with automatic frame generation mode can be done by the following method:

- S/W programs TMCR[LPON] = 0.

51.7.7.2 Clock test mode

The bit TMCR[CLKTST] enables or disables the Clock Test mode of the LFAST module. In this mode the LFAST sends fixed pattern out on the LD at the current configured RxData clock rate. It is a RWM bitfield and the default setting is 0 (off). This bit can be set under one of the following conditions:

- S/W programs TMCR[CLKTST].

The Tx Controller will send out a pattern of alternating 1 and 0 (pattern 101010...). This provides a divide by 2 test clock of the current Tx clock.

The Clock Test mode can be cancelled by either of the following methods:

- S/W programs TMCR[CLKTST] = 0.
- For LFAST Slave: Reception of ICLC frame with payload 38h (Test mode off) from LFAST Master

In clock test mode the Tx Controller does not output any synchronization pattern or header, just a pattern of alternating 1's and 0's.

This mode doesn't affect the functionality of the Rx Interface.

51.7.8 Interrupts

The LFAST module generates five interrupts to the processor:

- Transmit complete interrupt
- Transmit exception interrupt
- Receive complete interrupt
- Receive exception interrupt
- ICLC recieve interrupt

Each interrupt is asserted when any one of their status conditions is met.

51.7.8.1 Transmit complete interrupt

This interrupt indicates a transfer compeletion of a frame. This interrupt is asserted whenever any of the following bits are set in the TISR and also the corresponding mask bits are set in the TIER.

- TXDTF
- TXICLCF
- TXUNSF
- TXPNGF

Each of these bits can be individually maskable. So, those bits that are not required to generate an interrupt to the processor can be masked. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the TISR.

51.7.8.2 Transmit exception interrupt

This interrupt indicates occurrence of an exception condition in the Tx block. This interrupt is asserted whenever any of the following bits are set in the TISR along with the corresponding mask bits in the TIER.

- TXOVF
- TXIEF

Each of these bits can be individually masked. So, those bits that are not required to generate an interrupt to the processor can be masked. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the TISR.

Table 51-55. Recommended transmit exception handling

Exception	Recommendation
TXOVF	• MCR[TXEN] = 0 • The system level interfaces transmit is reseted and enabled again • For Slave: After reception of ICLC Ping Response Request the MCR[TXEN] may be set • For Master: The MCR[TXEN] should be set and an ICLC frame Ping Response Request should be sent
TXIEF	• The MCR[TXEN] should be set

51.7.8.3 Receive complete interrupt

This interrupt indicates a reception of a frame. This interrupt is asserted whenever any of the following bits is set in the RISR along with the corresponding interrupt enable bit is also set in the RIER.

- RXCTSF
- RXDF
- RXUNSF

Each of these bits are individually maskable. So, the bits that are not required to generate an interrupt to the processor can be masked by clearing its bit in the RIER. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the RISR.

51.7.8.4 Receive exception interrupt

This interrupt indicates occurrence of an exception condition in Rx block. This interrupt is asserted whenever any of the following bits is set in the RISR and corresponding enable mask bit is also set in the RIER.

- RXLCEF
- RXICF
- RXSZF
- RXUOF
- RXUFF
- RXMXF
- RXMNF
- RXOFF

Each of these bits are individually maskable. So, those bits that are not required to generate an interrupt to the processor can be masked by clearing the appropriate bit in the RIER. For details on the cause of assertion of each bit refer to the RISR description. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in the RISR.

Table 51-56. Recommended receive exception handling mechanism

Exception	Recommendation
RXLCEF	The MCR[RXEN] may be set and cleared, as this exception can result in loss of subsequent packet
RXICF	No action required
RXSZF	The MCR[RXEN] may be set and cleared, as this exception can result in loss of subsequent packet
RXOFF	• The MCR[RXEN] may be cleared • The system level interfaces receive may be reset and enabled, as current transfer may be corrupted
RXUFF	No action required
RXMXF	The system level interfaces receive should be enabled, if not enabled
RXMNF	No action required
RXUNSF	No action required

51.7.8.5 ICLC receive interrupt

This interrupt indicates reception of a valid ICLC frame. This interrupt is asserted whenever any of the following bits is set in the RIISR along with the corresponding enable bit mask bit is set in the RIIER.

- ICPONF
- ICPOFF
- ICTSF
- ICTFF
- ICRSF
- ICRFF
- ICTEF
- ICTDF
- ICCTF
- ICLPF
- ICTOF
- ICPRF
- ICPSF
- ICPFF

Each of these bits are individually maskable. So those bits that are not required to generate an interrupt to the processor can be masked. For details on the cause of assertion of each bit, refer RIISR. Once the interrupt is asserted, it can be negated by clearing the corresponding flag bit in RIISR.

51.8 Packet memory

The LFAST stores packet frames for transmission, and reads packet frames after reception. The transmitter has its own dedicated buffer and the receiver has it own dedicated buffer, and they are not shared between each other.

Table 51-57. Frames Supported by LFAST interfaces

Frame Type	Tx Buffer(in bits)	Rx Buffer(in bits)	Memory Type
Data Frame	38 × 32 max 6 packets	38 × 32 max 6 packets	FIFO
Unsolicited Frame	9 × 32 max 1 packet	9 × 32 max 1 packet	Registers UNSTD[8-0], UNSRDR[8-0]
ICLC Frame	1 × 7 max 1 packet[1]	1 × 8 max 1 packet[2]	Registers ICR, PISR
CTS Frame	N/A	N/A	N/A

1. Only for MCR[MSEN] = 1
2. Only for ping response data when MCR[MSEN] = 1

51.9 Resets

The various blocks in LFAST are reset as described in the table below.

Table 51-58. Recommended receive exception handling mechanism

Reset	Blocks Reset
Asynchronous Hardware reset	Polarity: Active Low • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Register Space
DRFRST bit of Mode Configuration Register (MCR)	Polarity: Active High • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Register Space
DRFEN bit of Mode Configuration Register (MCR)	Polarity: Active Low • Clock control module (CCM) • LFAST Domain Logic (Rx and Tx block) • System Side Module Interface FIFOs • LFAST Status Registers • SCR[RDR] and SCR[TDR] • TMCR[CLKTST] and TMCR[LPON] • ICR[ICLCSEQ] and ICR[SNDICLC] • PICR[PNGREQ] • UNSTCR[USNDRQ]

51.10 Clocks

The LFAST mainly works on three clocks:

- System Bus Clock used to program the registers.
- Protocol clock, which is either High Speed PLL clock or Low Speed clock depending on the speed mode, used for LFAST protocol operation.
- System Side Module clock, which is synchronous to the System Bus clock.

51.10.1 Clocking strategy

The following figure shows the clock domain in which each module functions. The PLL provides eight phases of the high speed clock to the Clock Muxing portion of the Clock Control Module. The Clock Module will then generate four phases of slow speed clock using both the edges of lfast_sysclk, muxes high speed and low speed clocks and provides a muxed clock to the Sampler Module.

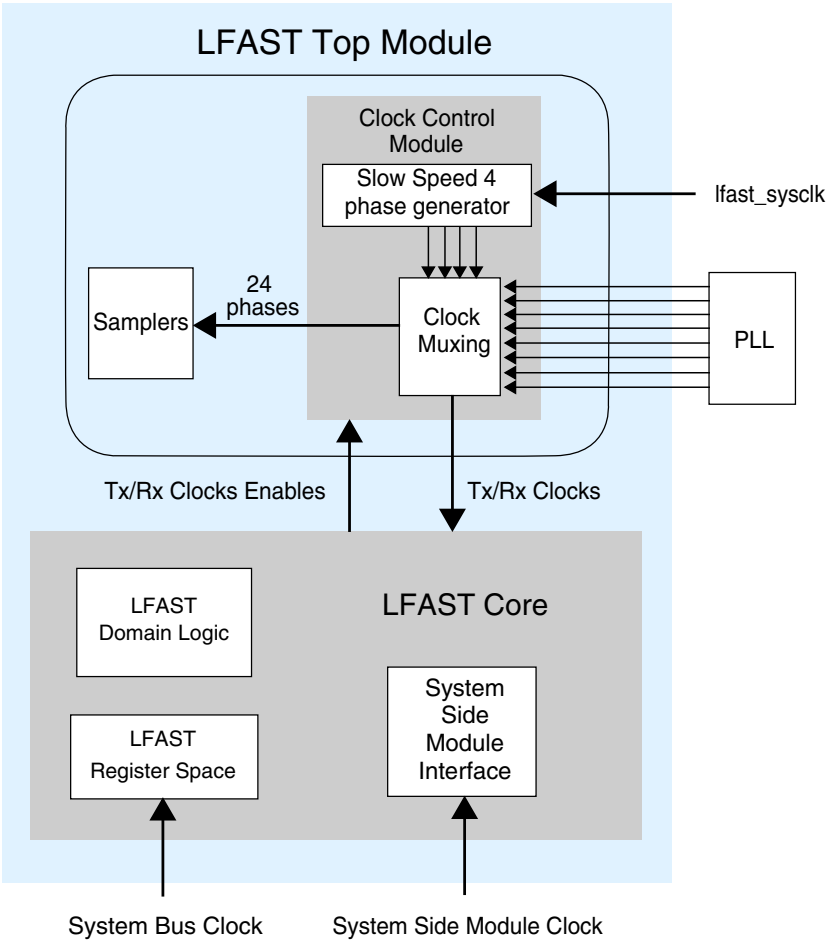


Figure 51-66. LFAST module clock domains

51.10.2 Slow speed clock

51.10.2.1 External muxing

The Clock Control Module will receive lfast_sysclk from the clocking subsystem as shown in Figure 51-67, Figure 51-68 and Figure 51-69.

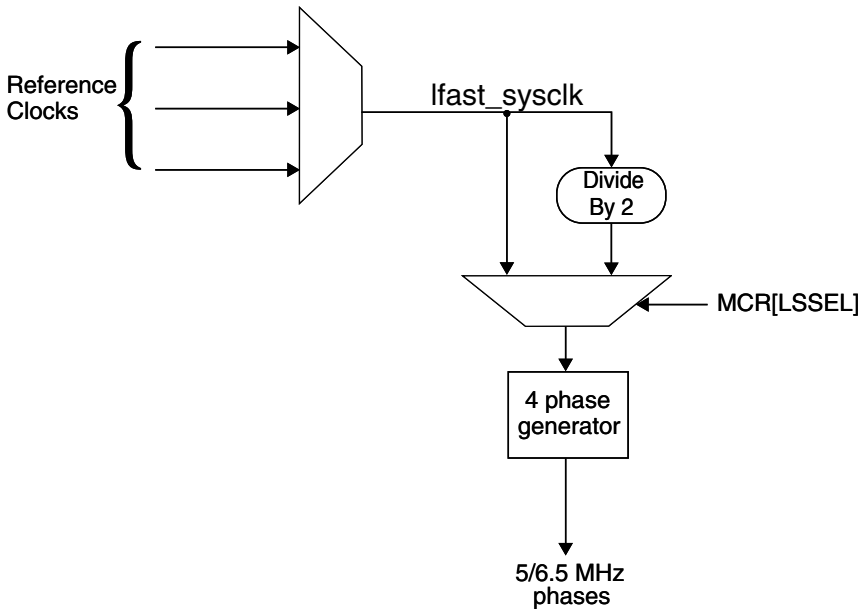


Figure 51-67. External clock muxing of lfast_sysclk

All the reference clocks will be muxed first to provide lfast_sysclk to the module and then either div/2 or direct muxed clock will be used to generate 4 slow speed phases in the Clock Control Module of LFAST as shown in Figure 51-67.

The lfast_sysclk could be either 20/26 MHz or 10/13 MHz. When lfast_sysclk is 20/26 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/4 when MCR[LSSEL] = 1. If lfast_sysclk is 20/26 MHz it needs to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in Figure 51-68 (selected clock path shown in green).

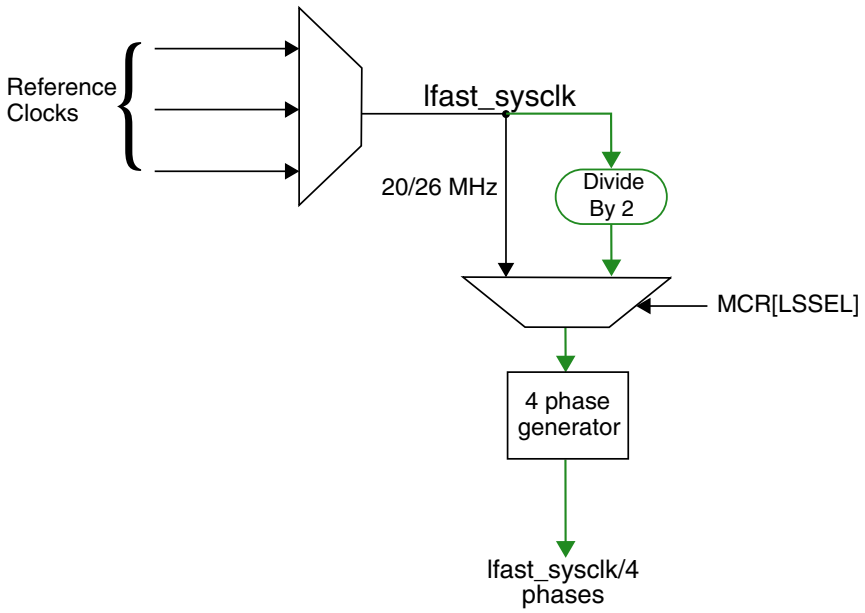


Figure 51-68. External clock muxing of lfast_sysclk of 20/26 MHz

In this case, lfast_sysclk is 10/13 MHz frequency, the clock control module will generate 4 phases of slow speed clock of frequency lfast_sysclk/2 when MCR[LSSEL] = 0. If lfast_sysclk is 10/13 MHz then it does not need to be first divided by 2 before using it for 4 phase generation in LFAST Clock Module, which generates 4 phases of half of the input frequency as shown in Figure 51-69 (selected clock path selected shown in magenta).

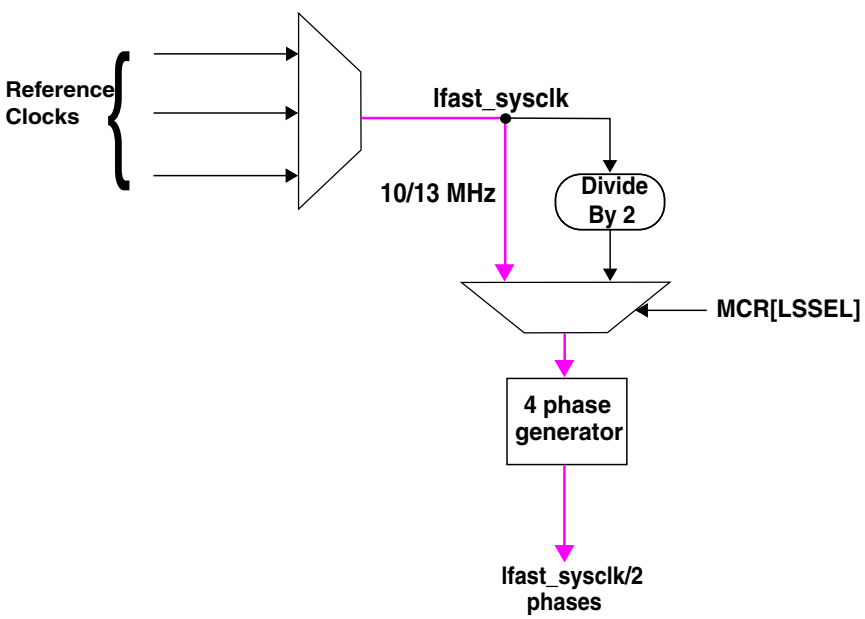


Figure 51-69. External clock muxing of lfast_sysclk of 10 MHz

51.10.2.2 Slow Speed 4 phase generator

Clock control module will generate 4 phases of slow speed clock of frequency of 10/13 MHz. For instance, if lfast_sysclk is 10 MHz then 4 phases of 5 MHz will be generated. The following figure shows 4 phases getting generated using lfast_sysclk.

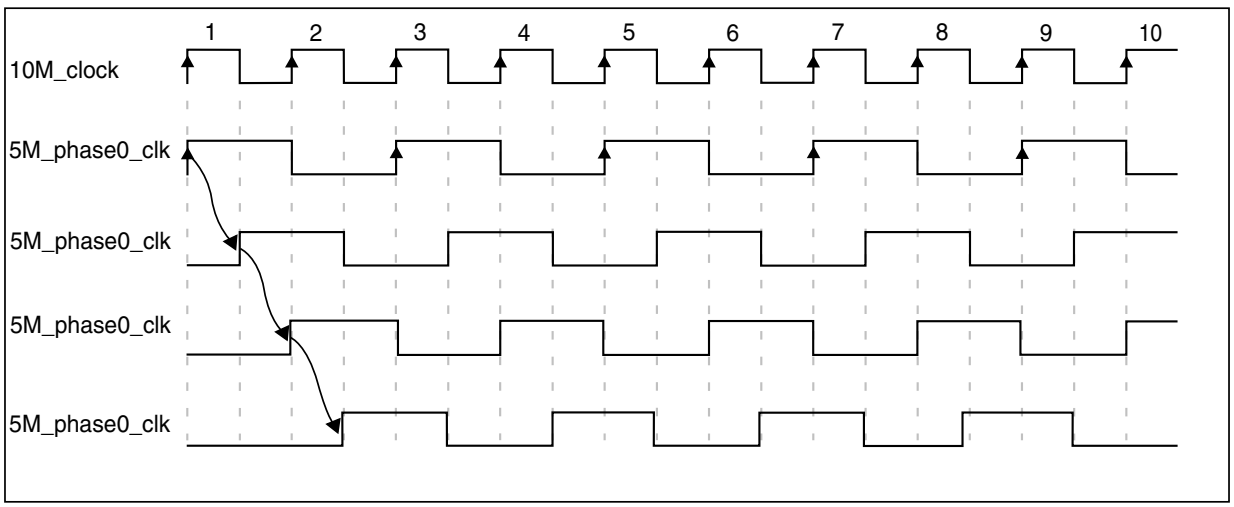


Figure 51-70. Slow speed 4 phases generation

51.10.3 Rx Controller Clocks

51.10.4 Clocking Module Requirements for High Speed Phases

The high speed phases are obtained from the PLL via the Clocking Module as shown in the following figure.

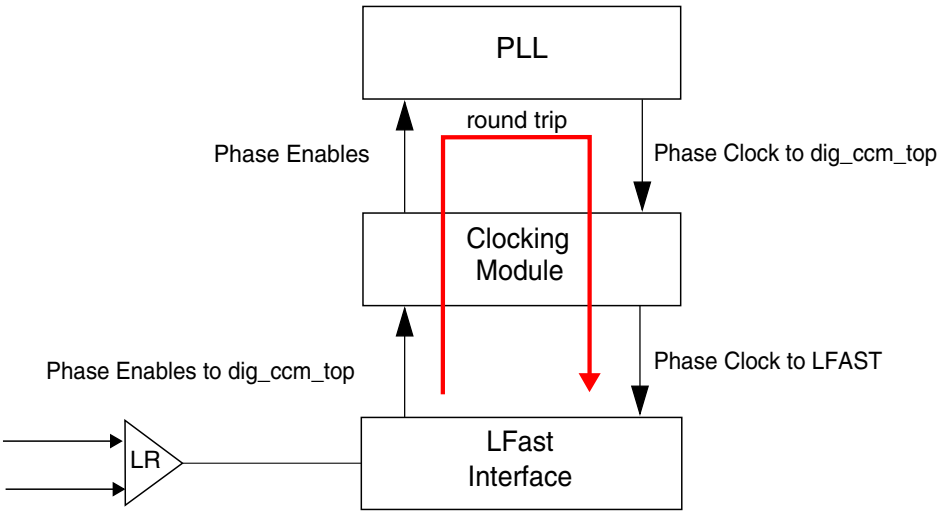


Figure 51-71. Clock enables and clock paths

The LFAST block will know which of the phases will be required for the different modes of operations. Therefore the LFAST block will provide enables and speed mode switches to the Clocking Module. The Clocking Module will use these signals to control the enables to the clock phases to reduce the power consumption.

The 8 phases of clocks signal are fed to the Clocking Module and muxed with the low speed phases. Depending on enables and data rate speed modes, the selected clocks are passed to the Sampler block and interface block.

COCR[PHSSEL] is connected to the Clocking Module, which selects whether 8 or 4 phases are selected for High Speed mode. It is only valid for high speed.

The routing of the 8 high speed phases to the interface is critical. Each clock phase will need to have the same routing track length from the PLL to the interface of Clocking Module. Each of the 8 high speed phases are separated by 45 degrees. This separation needs to be maintained from the phase generator to the interface.

51.10.5 Clock module requirements for low speed phases

51.10.5.1 Low speed

The low speed clock phases are generated in Clocking Module. These four phases are required to sample the data correctly at Low Speed mode. The LFAST block will provide the enable for the phases. The clock source for the low speed phases is SYSCLK. The Clocking Module block will need to generate the 4 phases of low speed clock 90 degrees apart.

Using the Low Speed mode on the interface allows the polyphase generation logic to be turned off and the requirement of high-speed-freq × 8 MHz clock from the PLL is not needed.

Table 51-59. Rx clocks summary

Rx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

51.10.6 Tx Controller Clocks

51.10.6.1 Clocking Module Requirements for Speed Phases

The low speed phase 0 clock is sourced from the lfast_sysclk and the high speed phase 0 clock is sourced from the PLL. See the following table.

Table 51-60. Tx clocks summary

Tx Speed mode	Source
Low Speed	lfast_sysclk
High Speed	PLL

51.10.6.2 Transmit Clock Muxing

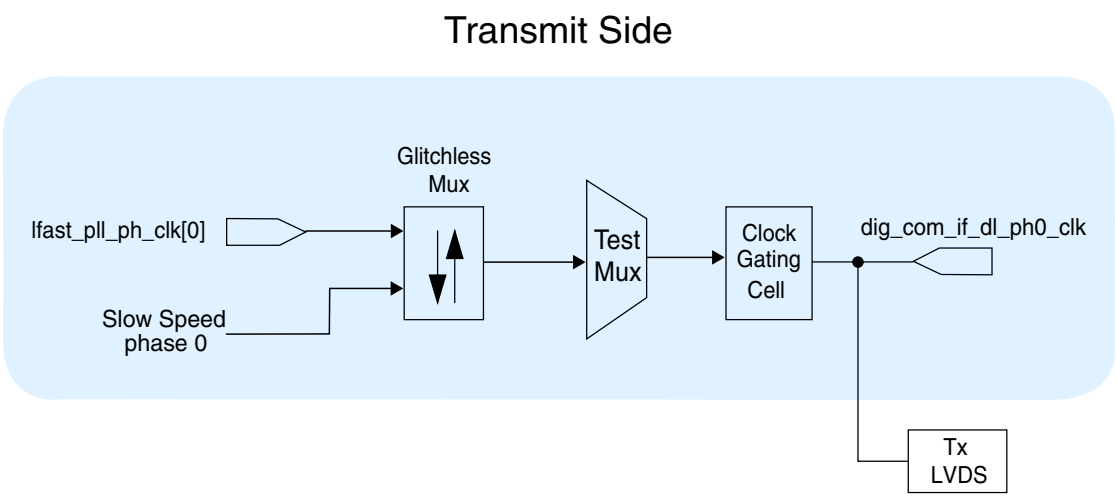


Figure 51-72. Transmit Clocks Muxing

51.10.6.3 Tx Request Clock Control

The Tx phase 0 clock is enabled when all the resets are negated.

Chapter 52
Fast Ethernet Controller (FEC)

52.1 Fast Ethernet Controller (FEC) chip-specific module information

52.1.1 FEC interface selection

On this chip, the FEC interface depends on the settings of the RCR (in the FEC) and SIUL2_SCR0 (in the SIUL2), as shown in the following table.

Table 52-1. FEC interface selection

RCR[MII_MODE]	SIUL2_SCR0[FEC_MODE]	Interface used by the FEC
0	0 or 1	7-wire
1	0	RMII
	1	MII-Lite[1]

1. MII-Lite supported only on BGA packages. Full MII is not supported on this chip.

52.2 Introduction

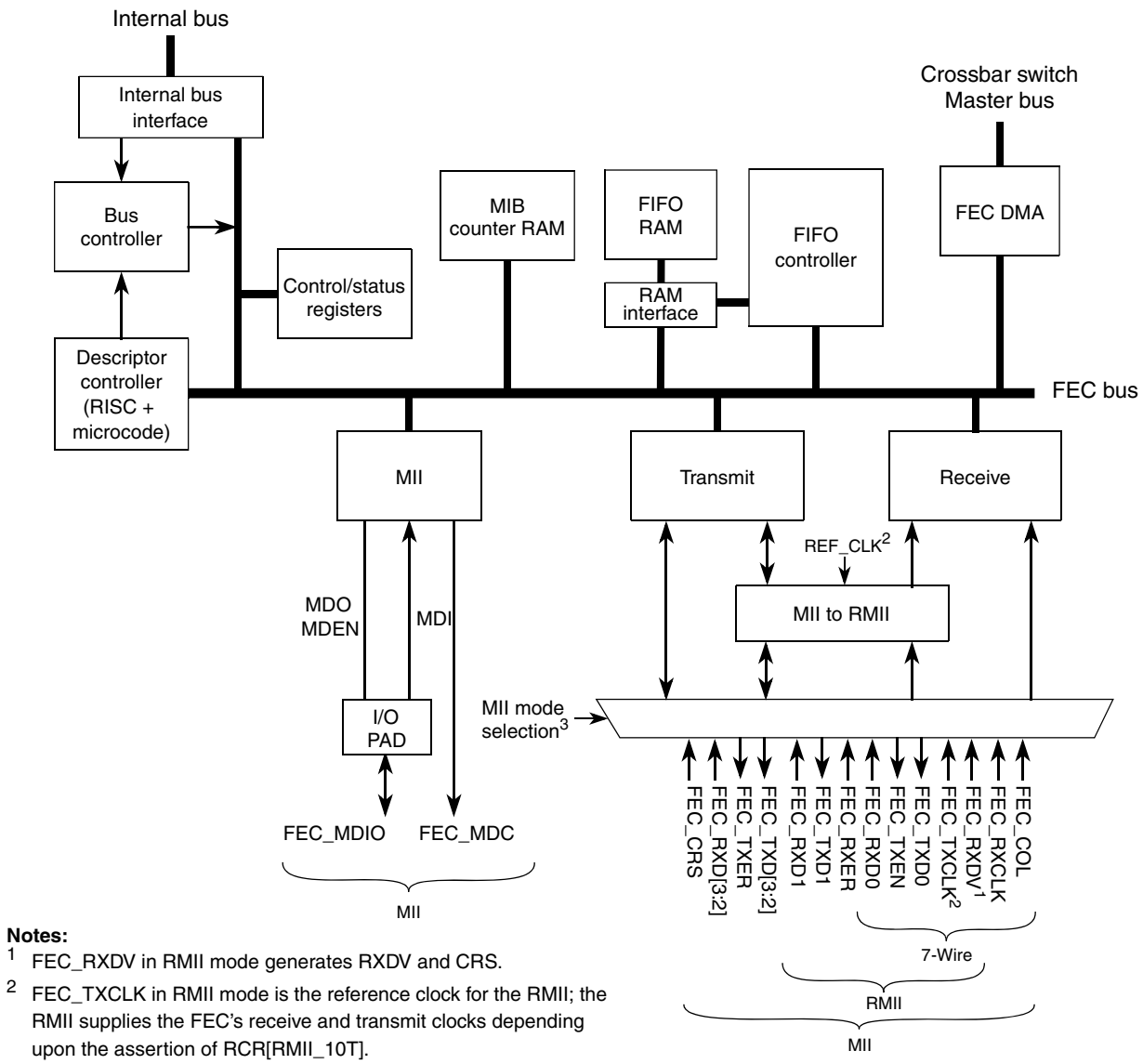
The Fast Ethernet Controller (FEC) is a communication controller that supports 10 and 100 Mbps Ethernet/IEEE 802.3 networks. An external transceiver interface and transceiver function are required to complete the interface to the media. The FEC supports five different standard MAC-PHY (physical) interfaces for connection to an external Ethernet transceiver. The FEC supports the 10/100 Mbps MII, 10/100 Mbps MII-Lite, 10/100 Mbps reduced MII, and the 10 Mbps-only 7-wire interface.

52.2.1 Block diagram

Figure 52-1 shows the block diagram of the FEC. The FEC is implemented with a combination of hardware and microcode. The off-chip (Ethernet) interfaces are compliant with industry and IEEE 802.3 standards.

NOTE

When MII-Lite is used on this chip, the FEC_TXER, FEC_RXER, FEC_COL, and FEC_CRS signals shown in Figure 52-1 are not used.



Notes:

1. FEC_RXDV in RMII mode generates RXDV and CRS.
2. FEC_TXCLK in RMII mode is the reference clock for the RMII; the RMII supplies the FEC's receive and transmit clocks depending upon the assertion of RCR[RMII_10T].
3. Interface selection is chip-specific; see the chapter that describes how modules are configured and connected.

Figure 52-1. FEC block diagram

The following table provides a brief description of the blocks in the previous figure.

Table 52-2. FEC sub-block descriptions

Block	Description
Control and status registers (CSR)	Provides global control (Ethernet reset and enable) and interrupt managing registers
Descriptor controller	A RISC-based controller providing these functions in the FEC: • Initialization (those internal registers not initialized by you or hardware) • High-level control of the DMA channels (initiating DMA transfers) • Interpreting buffer descriptors • Address recognition for receive frames • Random number generation for transmit collision backoff timer
DMA block	Provides multiple channels allowing transmit data, transmit descriptor, receive data and receive descriptor accesses to all run independently **Note:** In this document, "DMA" refers to the DMA block within the FEC, and is not related to any other DMA controllers that may be present on the chip.
Media independent interface (MII)	Provides a serial channel for control/status communication with the external physical layer device (transceiver). This serial channel consists of: • The management data clock (FEC_MDC) • The management data input/output lines (FEC_MDIO) of the MII interface
Message information block (MIB)	Maintains counters for a variety of network events and statistics. It is not necessary for operation of the FEC, but provides valuable counters for network management. The counters supported are: • The RMON (RFC 1757) Ethernet Statistics group • Some of the IEEE 802.3 counters
RAM	Serves as the focal point of all data flow in the Fast Ethernet controller and divides into transmit and receive FIFOs. The FIFO boundaries are programmable using the FRSR register. User data flows to/from the DMA block from/to the receive/transmit FIFOs. Transmit data flows from the transmit FIFO into the transmit block. Receive data flows from the receive block into the receive FIFO.
Transmit and receive blocks	Provide the Ethernet MAC functionality (with some assist from microcode)

52.2.2 Features

The FEC incorporates the following features:

- Support for the following Ethernet physical interfaces:[1]
  - 100-Mbps IEEE 802.3 MII
  - 10-Mbps IEEE 802.3 MII

1. For interface selection options, see the chip-specific FEC information.

    - 100-Mbps IEEE 802.3 MII-Lite
    - 10-Mbps IEEE 802.3 MII-Lite
    - 100-Mbps reduced media independent interface (RMII)
    - 10-Mbps reduced media independent interface (RMII)
    - 10-Mbps 7-wire interface (industry standard)
- IEEE 802.3 full duplex flow control
- Programmable max frame length supports IEEE 802.1 VLAN tags and priority
- Support for full-duplex operation (200 Mbps throughput) with a minimum internal bus clock rate of 50 MHz
- Support for half-duplex operation (100 Mbps throughput) with a minimum internal bus clock rate of 25 MHz
- Retransmission from transmit FIFO following a collision (no processor bus utilization)
- Automatic internal flushing of the receive FIFO for runts (collision fragments) and address recognition rejects (no processor bus utilization)
- Address recognition
    - Frames with broadcast address may be always accepted or always rejected
    - Exact match for single 48-bit individual (unicast) address
    - Hash (64-bit hash) check of individual (unicast) addresses
    - Hash (64-bit hash) check of group (multicast) addresses
    - Promiscuous mode

52.2.3 Clocking requirements

The FEC system clock frequency must be equal to or greater than the FEC protocol reference clock frequency (for example, 25 MHz system frequency for 100 Mbps protocol frequency). This requirement is true regardless of the reference selected by the application.

52.3 Modes of operation

This section describes the FEC operating modes.

52.3.1 Full and half duplex operation

Full duplex mode is for use on point-to-point links between switches or end node to switch. Half duplex mode works in connections between an end node and a repeater or between repeaters. TCR[FDEN] controls duplex mode selection.

When configured for full duplex mode, flow control may be enabled. For more details, see:

- The RFC_PAUSE and TFC_PAUSE fields in the TCR
- The RCR[FCE] field
- Full duplex flow control

52.3.2 Interface options

The following interface options are supported[2]. A detailed discussion of the interface configurations is provided in Network interface options.

52.3.2.1 10 Mbps and 100 Mbps MII interface

The IEEE 802.3 standard defines the media independent interface (MII) for 10/100 Mbps operation. The MAC-PHY interface may be configured to operate in MII mode by programming RCR[MII_MODE]=1.

The operation speed is determined by the FEC_TXCLK and FEC_RXCLK pins driven by the external transceiver. The transceiver auto-negotiates the speed or software controls it via the serial management interface (FEC_MDC/FEC_MDIO pins) to the transceiver. Refer to the MMFR and MSCR register descriptions, as well as the section on the MII, for a description of how to read and write registers in the transceiver via this interface.

2. For interface selection options, see the chip-specific FEC information.

52.3.2.2 10 Mbps and 100 Mbps MII-Lite interface

MII-Lite refers to MII without the following signals:

- FEC_COL
- FEC_CRS
- FEC_RXER
- FEC_TXER

The MII-Lite connection is for PHYs that do not have all the MII pins and signals that are available for MII.

The following table provides additional information on the behavior of these signals when MII-Lite is used.

Table 52-3. MII signals not used in MII-Lite

Signal	Information
FEC_COL	This signal is asserted by the PHY when there has been a collision. With full-duplex links, this would always be low, so this signal's absence implies either that the link in full-duplex or that collisions are handled internally within the PHY. The signal is only asserted high on simultaneous Rx and Tx on half-duplex links. As this must be handled by the PHY if the signal is not available, it can be safely pulled low.
FEC_CRS	Carrier sense is asserted high by the PHY to indicate that receive medium is non-idle. As in the case with the COL signal, this never occurs with full-duplex links. Again, it is recommended that this signal be pulled low using an external pulldown.
FEC_RXER	This signal is asserted high by the PHY to signal an error to the MAC. However, in these circumstances the PHY will deliberately corrupt the CRC of the incoming frame before passing to the MAC. This input to the FEC can therefore be pulled low and any RXER can be detected by a CRC error in the received frame where the CR bit is set in the receive buffer descriptor (RxBD).
FEC_TXER	This FEC output can be left unconfigured and unused. The signal was originally included for the test case in some PHY designs where the MAC layer in the FEC can demand that the PHY corrupt data during transmission. However, it is readily possible to do this within the MAC by setting the ABC (Append Bad CRC) bit in the transmit buffer descriptor (TxBD) for the frame being sent.

52.3.2.3 10 Mbps and 100 Mbps RMII interface

The reduced media independent interface (RMII) is a low-cost alternative to the IEEE802.3 MII standard. This interface provides the functionality of the MII interface on a total of 8 pins instead of 18. The RMII interface for 10/100 Ethernet MAC-PHY interface was defined by an industry consortium and is not currently included in the IEEE 802.3 standard. This functionality is selected as described in the chip configuration chapter, and is reflected in the read-only RCR[RMII_MODE] bit. The RCR[RMII_10T] bit determines the speed of operation. The reference clock for RMII is always 50 MHz, but this clock can be divided by 10 within the RCR register to support 10 Mbps operation. The PHY must be configured accordingly.

52.3.2.4 10 Mbps 7-wire interface operation

The FEC supports 7-wire interface used by many 10 Mbps Ethernet transceivers. The RCR[MII_MODE] bit controls this functionality. If this bit is cleared, MII mode is disabled and the 10 Mbps 7-wire mode is enabled.

52.3.3 Address recognition options

The address options supported are:

- Promiscuous
- Broadcast reject
- Individual address (hash or exact match)
- Multicast hash match

Address recognition options are discussed in detail in Ethernet address recognition.

52.3.4 Internal loopback

Internal loopback mode is selected via RCR[LOOP]. Loopback mode is discussed in detail in MII internal and external loopback.

52.4 External signal description

The following table describes the various FEC signals, as well as indicating which signals work in available modes.

Table 52-4. FEC signal descriptions

Signal name	MII	7-wire	RMII	Description
FEC_COL	X[1, 1]	X	—	Asserted upon detection of a collision and remains asserted while the collision persists. This signal is not defined for full-duplex mode.
FEC_CRS	X[1]	—	—	Carrier sense. When asserted, indicates transmit or receive medium is not idle. In RMII mode, this signal is present on the FEC_RXDV pin.
FEC_MDC	X	—	X	Output clock provides a timing reference to the PHY for data transfers on the FEC_MDIO signal.

Table continues on the next page...

Table 52-4. FEC signal descriptions (continued)

Signal name	MII	7-wire	RMII	Description
FEC_MDIO	X	—	X	Transfers control information between the external PHY and the media-access controller. Data is synchronous to FEC_MDC. This signal is an input after reset. When the FEC operates in 10Mbps 7-wire interface mode, this signal should be connected to VSS.
FEC_RXCLK	X	X	—	Provides a timing reference for FEC_RXDV, FEC_RXD[3:0], and FEC_RXER.
FEC_RXDV	X	X	X	Asserting the FEC_RXDV input indicates PHY has valid nibbles present on the MII. FEC_RXDV must remain asserted from the first recovered nibble of the frame through to the last nibble. Assertion of FEC_RXDV must start no later than the SFD and exclude any EOF. In RMII mode, this pin also generates the CRS signal.
FEC_RXD0	X	X	X	This pin contains the Ethernet input data transferred from PHY to the media-access controller when FEC_RXDV is asserted.
FEC_RXD1	X	—	X	This pin contains the Ethernet input data transferred from PHY to the media access controller when FEC_RXDV is asserted.
FEC_RXD[3:2]	X	—	—	These pins contain the Ethernet input data transferred from PHY to the media access controller when FEC_RXDV is asserted.
FEC_RXER	X[1]	—	X	When asserted with FEC_RXDV, indicates PHY detects an error in the current frame. When FEC_RXDV is not asserted, FEC_RXER has no effect.
FEC_TXCLK	X	X	X	Input clock which provides a timing reference for FEC_TXEN, FEC_TXD[3:0] and FEC_TXER. In RMII mode, this signal is the reference clock for receive, transmit, and the control interface.
FEC_TXD0	X	X	X	The serial output Ethernet data and only valid during the assertion of FEC_TXEN.
FEC_TXD1	X	—	X	This pin contains the serial output Ethernet data and valid only during assertion of FEC_TXEN.
FEC_TXD[3:2]	X	—	—	These pins contain the serial output Ethernet data and valid only during assertion of FEC_TXEN.
FEC_TXEN	X	X	X	Indicates when valid nibbles are present on the MII. This signal is asserted with the first nibble of a preamble and is negated before the first FEC_TXCLK following the final nibble of the frame.
FEC_TXER	X[1]	—	—	When asserted for one or more clock cycles while FEC_TXEN is also asserted, PHY sends one or more illegal symbols. FEC_TXER has no effect at 10 Mbps or when FEC_TXEN is negated.
Note: Interface selection is chip-specific; see the chapter that describes how modules are configured and connected.				

1. This signal is not available in the MII-Lite interface.

52.5 Memory map and register definition

The FEC is programmed by a combination of control/status registers (CSRs) and buffer descriptors. The CSRs control operation modes and extract global status information. The descriptors pass data buffers and related buffer information between the hardware and software.

Each FEC implementation requires a 1 KB memory map space, which is divided into two sections of 512 bytes each for:

- Control/status registers
- Event/statistic counters held in the MIB block

The following table defines the top-level memory map.

Table 52-5. FEC memory map

Address offset	Function
000h–1FFh	Control/status registers
200h–3FFh	MIB block counters

The following table shows the FEC register map. Any address offset not explicitly mentioned in this table is reserved. Do not modify the content of reserved fields. All accesses to and from the registers must be via 32-bit accesses. There is no support for accesses other than 32-bit. Accessing reserved or unimplemented register spaces in the FEC will not return transfer errors.

The MIB counters (shown in the following table) define the locations in the MIB RAM space where hardware-maintained counters reside. The counters are divided into two groups:

- RMON counters include the Ethernet statistics counters defined in RFC 1757
- Truncated frames counter (only frames with lengths up to 2047 bytes are supported)

The transmit and receive RMON counters are independent, which ensures accurate network statistics when operating in full duplex mode.

The included IEEE counters support the mandatory and recommended counter packages defined in Section 5 of ANSI/IEEE Std. 802.3 (1998 edition). The FEC supports IEEE Basic Package objects, but these do not require counters in the MIB block. In addition, some of the recommended package objects supported do not require MIB counters. Counters for transmit and receive full duplex flow control frames are also included.

FEC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4	Interrupt Event Register (FEC_EIR)	32	w1c	0000_0000h	52.5.1/2233
8	Interrupt Mask Register (FEC_EIMR)	32	R/W	0000_0000h	52.5.2/2235
10	Receive Descriptor Active Register (FEC_RDAR)	32	R/W	0000_0000h	52.5.3/2237
14	Transmit Descriptor Active Register (FEC_TDAR)	32	R/W	0000_0000h	52.5.4/2238
24	Ethernet Control Register (FEC_ECR)	32	R/W	F000_0000h	52.5.5/2239
40	MII Management Frame Register (FEC_MMFR)	32	R/W	Undefined	52.5.6/2240
44	MII Speed Control Register (FEC_MSCR)	32	R/W	0000_0000h	52.5.7/2241
64	MIB Control Register (FEC_MIBC)	32	R/W	C000_0000h	52.5.8/2242
84	Receive Control Register (FEC_RCR)	32	R/W	05EE_0001h	52.5.9/2244
C4	Transmit Control Register (FEC_TCR)	32	R/W	0000_0000h	52.5.10/ 2246
E4	Physical Address Low Register (FEC_PALR)	32	R/W	See section	52.5.11/ 2247
E8	Physical Address High Register and Type Field (FEC_PAUR)	32	R/W	See section	52.5.12/ 2248
EC	Opcode/Pause Duration (FEC_OPD)	32	R/W	See section	52.5.13/ 2248
118	Descriptor Individual Upper Address Register (FEC_IAUR)	32	R/W	See section	52.5.14/ 2249
11C	Descriptor Individual Lower Address Register (FEC_IALR)	32	R/W	See section	52.5.15/ 2249
120	Descriptor Group Upper Address Register (FEC_GAUR)	32	R/W	See section	52.5.16/ 2250
124	Descriptor Group Lower Address Register (FEC_GALR)	32	R/W	See section	52.5.17/ 2250
144	Transmit FIFO Watermark (FEC_TFWR)	32	R/W	0000_0000h	52.5.18/ 2251
14C	FIFO Receive Bound Register (FEC_FRBR)	32	R	0000_0600h	52.5.19/ 2252
150	FIFO Receive Start Register (FEC_FRSR)	32	R/W	0000_0500h	52.5.20/ 2252
180	Receive Descriptor Ring Start Register (FEC_ERDSR)	32	R/W	See section	52.5.21/ 2253
184	Transmit Buffer Descriptor Ring Start Register (FEC_ETDSR)	32	R/W	See section	52.5.22/ 2254
188	Receive Buffer Size Register (FEC_EMRBR)	32	R/W	See section	52.5.23/ 2254
200	Count of frames not counted correctly (FEC_RMON_T_DROP)	32	R/W	Undefined	52.5.24/ 2255
204	RMON Tx packet count (FEC_RMON_T_PACKETS)	32	R/W	Undefined	52.5.25/ 2256
208	RMON Tx broadcast packets (FEC_RMON_T_BC_PKT)	32	R/W	Undefined	52.5.26/ 2256

Table continues on the next page...

FEC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20C	RMON Tx multicast packets (FEC_RMON_T_MC_PKT)	32	R/W	Undefined	52.5.27/ 2257
210	RMON Tx packets with CRC/align error (FEC_RMON_T_CRC_ALIGN)	32	R/W	Undefined	52.5.28/ 2257
214	RMON Tx packets < 64 bytes, good CRC (FEC_RMON_T_UNDERSIZE)	32	R/W	Undefined	52.5.29/ 2258
218	RMON Tx packets > MAX_FL bytes, good CRC (FEC_RMON_T_OVERSIZE)	32	R/W	Undefined	52.5.30/ 2258
21C	RMON Tx packets < 64 bytes, bad CRC (FEC_RMON_T_FRAG)	32	R/W	Undefined	52.5.31/ 2259
220	RMON Tx packets > MAX_FL bytes, bad CRC (FEC_RMON_T_JAB)	32	R/W	Undefined	52.5.32/ 2259
224	RMON Tx collision count (FEC_RMON_T_COL)	32	R/W	Undefined	52.5.33/ 2260
228	RMON Tx 64 byte packets (FEC_RMON_T_P64)	32	R/W	Undefined	52.5.34/ 2260
22C	RMON Tx 65 to 127 byte packets (FEC_RMON_T_P65TO127)	32	R/W	Undefined	52.5.35/ 2261
230	RMON Tx 128 to 255 byte packets (FEC_RMON_T_P128TO255)	32	R/W	Undefined	52.5.36/ 2261
234	RMON Tx 256 to 511 byte packets (FEC_RMON_T_P256TO511)	32	R/W	Undefined	52.5.37/ 2262
238	RMON Tx 512 to 1023 byte packets (FEC_RMON_T_P512TO1023)	32	R/W	Undefined	52.5.38/ 2262
23C	RMON Tx 1024 to 2047 byte packets (FEC_RMON_T_P1024TO2047)	32	R/W	Undefined	52.5.39/ 2263
240	RMON Tx packets with > 2048 bytes (FEC_RMON_T_P_GTE2048)	32	R/W	Undefined	52.5.40/ 2263
244	RMON Tx Octets (FEC_RMON_T_OCTETS)	32	R/W	Undefined	52.5.41/ 2264
248	Count of transmitted frames not counted correctly (FEC_IEEE_T_DROP)	32	R/W	Undefined	52.5.42/ 2264
24C	Frames transmitted OK (FEC_IEEE_T_FRAME_OK)	32	R/W	Undefined	52.5.43/ 2265
250	Frames transmitted with single collision (FEC_IEEE_T_1COL)	32	R/W	Undefined	52.5.44/ 2265
254	Frames transmitted with multiple collisions (FEC_IEEE_T_MCOL)	32	R/W	Undefined	52.5.45/ 2266
258	Frames transmitted after deferral delay (FEC_IEEE_T_DEF)	32	R/W	Undefined	52.5.46/ 2266
25C	Frames transmitted with late collision (FEC_IEEE_T_LCOL)	32	R/W	Undefined	52.5.47/ 2267
260	Frames transmitted with excessive collisions (FEC_IEEE_T_EXCOL)	32	R/W	Undefined	52.5.48/ 2267

Table continues on the next page...

FEC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
264	Frames transmitted with Tx FIFO underrun (FEC_IEEE_T_MACERR)	32	R/W	Undefined	52.5.49/ 2268
268	Frames transmitted with carrier sense error (FEC_IEEE_T_CSERR)	32	R/W	Undefined	52.5.50/ 2268
26C	Frames transmitted with SQE error (FEC_IEEE_T_SQE)	32	R/W	Undefined	52.5.51/ 2269
270	Flow control pause frames transmitted (FEC_IEEE_T_FDXFC)	32	R/W	Undefined	52.5.52/ 2269
274	Octet count for frames transmitted without error (FEC_IEEE_T_OCTETS_OK)	32	R/W	Undefined	52.5.53/ 2270
280	Count of received frames not counted correctly (FEC_RMON_R_DROP)	32	R/W	Undefined	52.5.54/ 2270
284	RMON Rx packet count (FEC_RMON_R_PACKETS)	32	R/W	Undefined	52.5.55/ 2271
288	RMON Rx broadcast packets (FEC_RMON_R_BC_PKT)	32	R/W	Undefined	52.5.56/ 2271
28C	RMON Rx multicast packets (FEC_RMON_R_MC_PKT)	32	R/W	Undefined	52.5.57/ 2272
290	RMON Rx packets with CRC/Align error (FEC_RMON_R_CRC_ALIGN)	32	R/W	Undefined	52.5.58/ 2272
294	RMON Rx packets < 64 bytes, good CRC (FEC_RMON_R_UNDERSIZE)	32	R/W	Undefined	52.5.59/ 2273
298	RMON Rx packets > MAX_FL bytes, good CRC (FEC_RMON_R_OVERSIZE)	32	R/W	Undefined	52.5.60/ 2273
29C	RMON Rx packets < 64 bytes, bad CRC (FEC_RMON_R_FRAG)	32	R/W	Undefined	52.5.61/ 2274
2A0	RMON Rx packets > MAX_FL bytes, bad CRC (FEC_RMON_R_JAB)	32	R/W	Undefined	52.5.62/ 2274
2A4	Reserved (FEC_RMON_R_RESVD_0)	32	R/W	Undefined	52.5.63/ 2275
2A8	RMON Rx 64 byte packets (FEC_RMON_R_P64)	32	R/W	Undefined	52.5.64/ 2275
2AC	RMON Rx 65 to 127 byte packets (FEC_RMON_R_P65TO127)	32	R/W	Undefined	52.5.65/ 2276
2B0	RMON Rx 128 to 255 byte packets (FEC_RMON_R_P128TO255)	32	R/W	Undefined	52.5.66/ 2276
2B4	RMON Rx 256 to 511 byte packets (FEC_RMON_R_P256TO511)	32	R/W	Undefined	52.5.67/ 2277
2B8	RMON Rx 512 to 1023 byte packets (FEC_RMON_R_P512TO1023)	32	R/W	Undefined	52.5.68/ 2277
2BC	RMON Rx 1024 to 2047 byte packets (FEC_RMON_R_P1024TO2047)	32	R/W	Undefined	52.5.69/ 2278
2C0	RMON Rx packets with > 2048 bytes (FEC_RMON_R_P_GTE2048)	32	R/W	Undefined	52.5.70/ 2278

Table continues on the next page...

FEC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2C4	RMON Rx octets (FEC_RMON_R_OCTETS)	32	R/W	Undefined	52.5.71/ 2279
2C8	Count of received frames not counted correctly (FEC_IEEE_R_DROP)	32	R/W	Undefined	52.5.72/ 2279
2CC	Frames received OK (FEC_IEEE_R_FRAME_OK)	32	R/W	Undefined	52.5.73/ 2280
2D0	Frames received with CRC error (FEC_IEEE_R_CRC)	32	R/W	Undefined	52.5.74/ 2280
2D4	Frames received with alignment error (FEC_IEEE_R_ALIGN)	32	R/W	Undefined	52.5.75/ 2281
2D8	Receive FIFO overflow count (FEC_IEEE_R_MACERR)	32	R/W	Undefined	52.5.76/ 2281
2DC	Flow control pause frames received (FEC_IEEE_R_FDXFC)	32	R/W	Undefined	52.5.77/ 2282
2E0	Octet count for frames received without error (FEC_IEEE_R_OCTETS_OK)	32	R/W	Undefined	52.5.78/ 2282

52.5.1 Interrupt Event Register (FEC_EIR)

When an event occurs that sets a bit in EIR, an interrupt occurs if the corresponding bit in the interrupt mask register (EIMR) is also set. Writing a 1 to an EIR bit clears it; writing 0 has no effect. This register is cleared upon hardware reset.

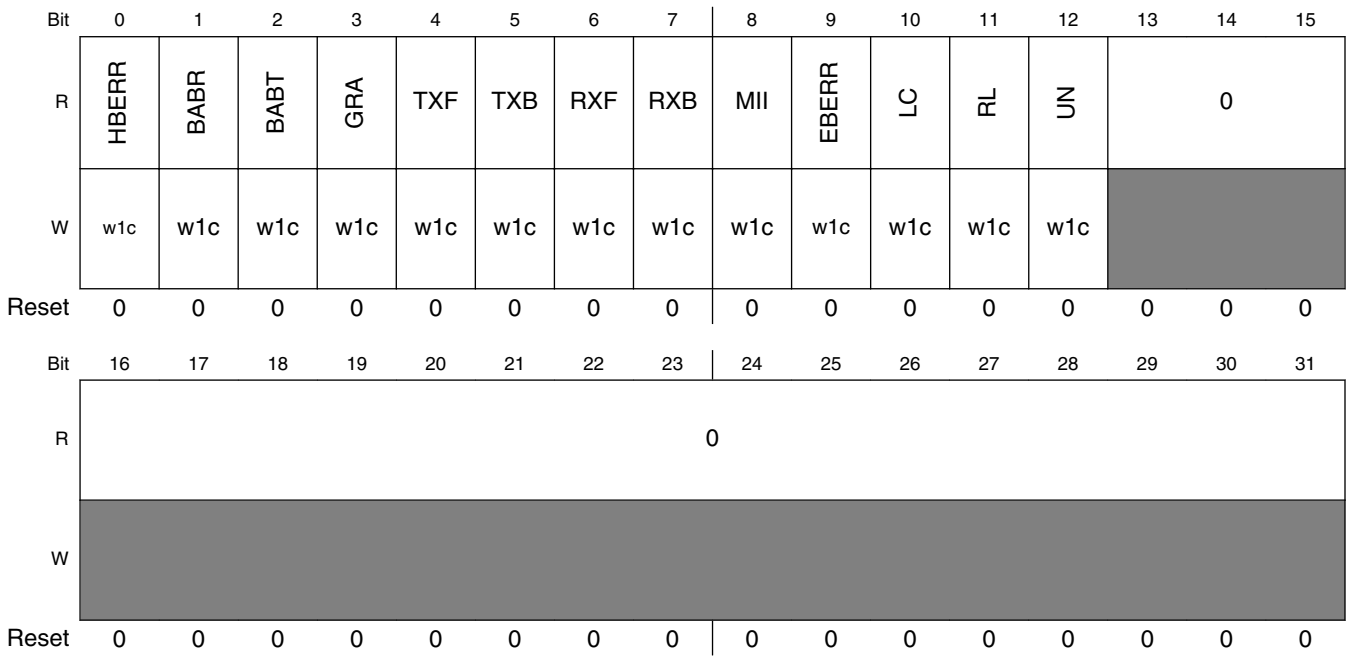
These interrupts can be divided into operational interrupts, transceiver/network error interrupts, and internal error interrupts. Interrupts which may occur in normal operation are GRA, TXF, TXB, RXF, RXB, and MII. Interrupts resulting from errors/problems detected in the network or transceiver are HBERR, BABR, BABT, LC, and RL. Interrupts resulting from internal errors are HBERR and UN.

Some of the error interrupts are independently counted in the MIB block counters:

- HBERR - IEEE_T_SQE
- BABR - RMON_R_OVERSIZE (good CRC), RMON_R_JAB (bad CRC)
- BABT - RMON_T_OVERSIZE (good CRC), RMON_T_JAB (bad CRC)
- LATE_COL - IEEE_T_LCOL
- COL_RETRY_LIM - IEEE_T_EXCOL
- XFIFO_UN - IEEE_T_MACERR

Software may choose to mask off these interrupts because these errors are visible to network management via the MIB counters.

Address: 0h base + 4h offset = 4h



FEC_EIR field descriptions

Field	Description
0 HBERR	Heartbeat error. Indicates TCR[HBC] is set and that the COL input was not asserted within the heartbeat window following a transmission.
1 BABR	Babbling receive error. Indicates a frame was received with length in excess of RCR[MAX_FL] bytes.
2 BABT	Babbling transmit error. Indicates the transmitted frame length exceeds RCR[MAX_FL] bytes. Usually this condition is caused by a frame that is too long is placed into the transmit data buffer(s). Truncation does not occur.
3 GRA	Graceful stop complete. Indicates the graceful stop is complete. During graceful stop the transmitter is placed into a pause state after completion of the frame currently being transmitted. This bit is set by one of these conditions: • A graceful stop initiated by the setting of the TCR[GTS] bit is now complete. • A graceful stop initiated by the setting of the TCR[TFC_PAUSE] bit is now complete. • A graceful stop initiated by the reception of a valid full duplex flow control pause frame is now complete. See Full duplex flow control.
4 TXF	Transmit frame interrupt. Indicates a frame has been transmitted and the last corresponding buffer descriptor has been updated.
5 TXB	Transmit buffer interrupt. Indicates a transmit buffer descriptor has been updated.
6 RXF	Receive frame interrupt. Indicates a frame has been received and the last corresponding buffer descriptor has been updated.
7 RXB	Receive buffer interrupt. Indicates a receive buffer descriptor not the last in the frame has been updated.
8 MII	MII interrupt. Indicates the MII has completed the data transfer requested.
9 EBERR	Ethernet bus error. Indicates a system bus error occurred when a DMA transaction is underway. When the EBERR bit is set, ECR[ETHER_EN] is cleared, halting frame processing by the FEC. When this occurs, software needs to ensure that the FIFO controller and DMA also soft reset.

Table continues on the next page...

FEC_EIR field descriptions (continued)

Field	Description
10 LC	Late collision. Indicates a collision occurred beyond the collision window (slot time) in half duplex mode. The frame truncates with a bad CRC and the remainder of the frame is discarded.
11 RL	Collision retry limit. Indicates a collision occurred on each of 16 successive attempts to transmit the frame. The frame is discarded without being transmitted and transmission of the next frame commences. This error can only occur in half duplex mode.
12 UN	Transmit FIFO underrun. Indicates the transmit FIFO became empty before the complete frame was transmitted. A bad CRC is appended to the frame fragment and the remainder of the frame is discarded.
13–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

52.5.2 Interrupt Mask Register (FEC_EIMR)

The EIMR controls which interrupt events are allowed to generate actual interrupts. A hardware reset clears this register. If the corresponding bits in the EIR and EIMR registers are set, an interrupt is generated. The interrupt signal remains asserted until a 1 is written to the EIR bit (write 1 to clear) or a 0 is written to the EIMR bit.

Each bit in this register corresponds to an interrupt source defined by the EIR. The corresponding EIMR bit determines whether an interrupt condition can generate an interrupt. At every processor clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR bit reflects the state of the interrupt signal even if the corresponding EIMR bit is set.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HBERR	BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN	0		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_EIMR field descriptions

Field	Description
0 HBERR	Heartbeat error interrupt mask.

Table continues on the next page...

FEC_EIMR field descriptions (continued)

Field	Description
	0 The interrupt source is masked. 1 The interrupt source is not masked.
1 BABR	Babbling receive error interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
2 BABT	Babbling transmit error interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
3 GRA	Graceful stop complete interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
4 TXF	Transmit frame interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
5 TXB	Transmit buffer interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
6 RXF	Receive frame interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
7 RXB	Receive buffer interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
8 MII	MII interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
9 EBERR	Ethernet bus error interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
10 LC	Late collision interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
11 RL	Collision retry limit interrupt mask. 0 The interrupt source is masked. 1 The interrupt source is not masked.
12 UN	Transmit FIFO underrun interrupt mask.

Table continues on the next page...

FEC_EIMR field descriptions (continued)

Field	Description
	0 The interrupt source is masked. 1 The interrupt source is not masked.
13–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

52.5.3 Receive Descriptor Active Register (FEC_RDAR)

RDAR is a command register, written by the user, indicating the receive descriptor ring is updated (the driver produced empty receive buffers with the empty bit set).

When the register is written, the RDAR bit is set. This is independent of the data actually written by the user. When set, the FEC polls the receive descriptor ring and processes receive frames (if ECR[ETHER_EN] is also set). After the FEC polls a receive descriptor whose empty bit is not set, FEC clears the RDAR bit and ceases receive descriptor ring polling until the user sets the bit again, signifying that additional descriptors are placed into the receive descriptor ring.

The RDAR is cleared at reset and when ECR[ETHER_EN] is cleared.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							RDAR	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_RDAR field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 RDAR	Set to 1 when this register is written, regardless of the value written. Cleared by the FEC when no additional empty descriptors remain in the receive ring. Also cleared when ECR[ETHER_EN] is cleared.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

52.5.4 Transmit Descriptor Active Register (FEC_TDAR)

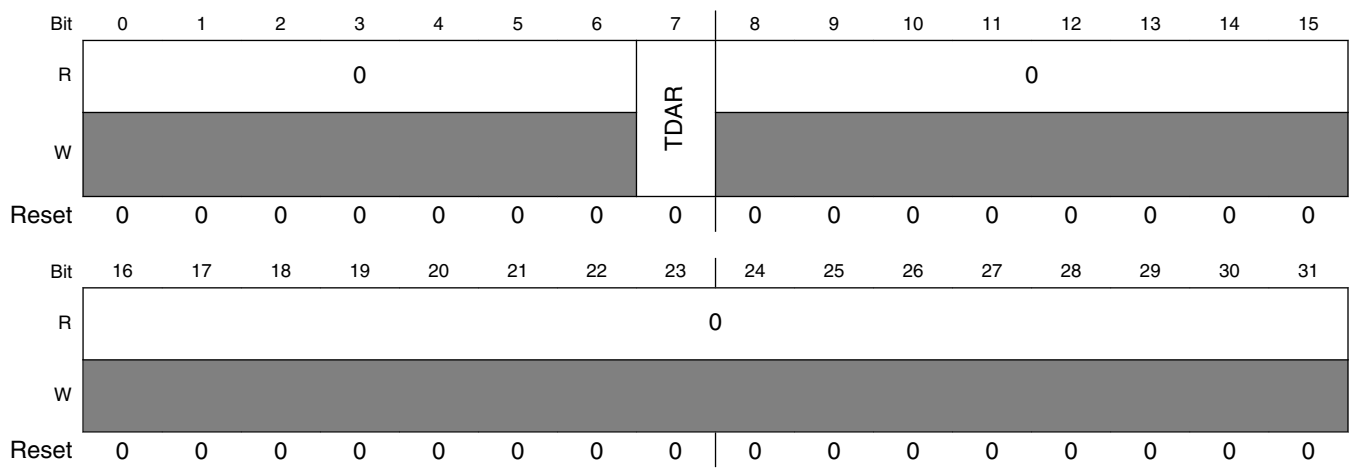
The TDAR is a command register which the user writes to indicate the transmit descriptor ring is updated (transmit buffers have been produced by the driver with the ready bit set in the buffer descriptor).

When the register is written, the TDAR field is set. This value is independent of the data actually written by the user. When set, the FEC polls the transmit descriptor ring and processes transmit frames (provided ECR[ETHER_EN] is also set). After the FEC polls a transmit descriptor that is a ready bit not set, FEC clears the TDAR field and ceases transmit descriptor ring polling until the user sets the bit again, signifying additional descriptors are placed into the transmit descriptor ring.

The TDAR field is cleared when any of the following events occur:

- Chip reset
- When ECR[ETHER_EN] is cleared
- When ECR[RESET] is set

Address: 0h base + 14h offset = 14h



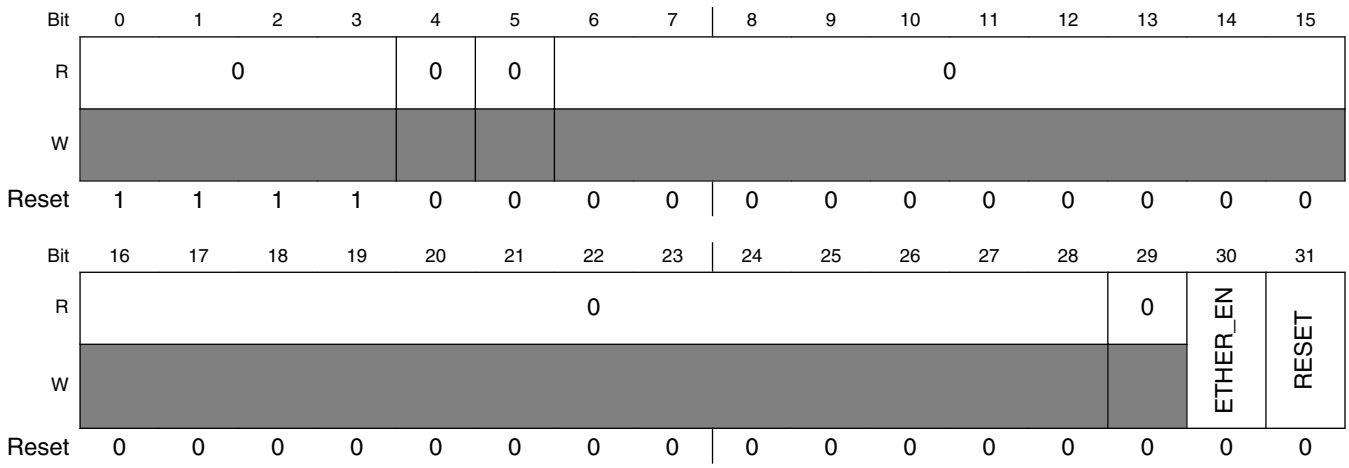
FEC_TDAR field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 TDAR	Set to 1 when this register is written, regardless of the value written. Cleared by the FEC device when no additional ready descriptors remain in the transmit ring. Also cleared when ECR[ETHER_EN] is cleared.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

52.5.5 Ethernet Control Register (FEC_ECR)

ECR is a read/write user register, though hardware may alter fields in this register as well. The ECR enables/disables the FEC.

Address: 0h base + 24h offset = 24h



FEC_ECR field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 ETHER_EN	When this bit is set, FEC is enabled, and reception and transmission are possible. When this bit is cleared, reception immediately stops and transmission stops after a bad CRC is appended to any currently transmitted frame. The buffer descriptor(s) for an aborted transmit frame are not updated after clearing this bit. When ETHER_EN is cleared, the DMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers. Hardware alters the ETHER_EN bit under the following conditions: • ECR[RESET] is set by software, in which case ETHER_EN is cleared. • An error condition causes the EIR[EBERR] bit to set, in which case ETHER_EN is cleared.
31 RESET	When this bit is set, the equivalent of a hardware reset is performed but it is local to the FEC. ECR[ETHER_EN] is cleared and all other FEC registers take their reset values. Also, any transmission/reception currently in progress is abruptly aborted. This bit is automatically cleared by hardware during the reset sequence. The reset sequence takes approximately eight internal bus clock cycles after this bit is set.

52.5.6 MII Management Frame Register (FEC_MMFR)

The MMFR is used to communicate with the attached MII compatible PHY device(s), providing read/write access to their MII registers. Performing a write to the MMFR causes a management frame to be sourced unless the MSCR is programmed to 0. If MSCR is cleared while MMFR is written and then MSCR is written with a non-zero value, an MII frame is generated with the data previously written to the MMFR. This allows MMFR and MSCR to be programmed in either order if MSCR is currently zero.

The MMFR does not reset to any predefined value.

To perform a read or write operation on the MII Management Interface, write the MMFR. To generate a valid read or write management frame, ST field must be written with a 01 pattern, and the TA field must be written with a 10. If other patterns are written to these fields, a frame is generated, but does not comply with the IEEE 802.3 MII definition.

To generate an IEEE 802.3-compliant MII Management Interface write frame (write to a PHY register), write {01 01 PHYAD REGAD 10 DATA} to the MMFR. Writing this pattern causes the control logic to shift out the data in the MMFR following a preamble generated by the control state machine. During this time, contents of the MMFR register are altered as the contents are serially shifted and are unpredictable if read by the user. After the write management frame operation completes, the MII interrupt is generated. At this time, contents of the MMFR match the original value written.

To generate an MII management interface read frame (read a PHY register), write {01 10 PHYAD REGAD 10 XXXX} to the MMFR (the content of the DATA field is unimportant). Writing this pattern causes the control logic to shift out the data in the MMFR following a preamble generated by the control state machine. During this time, contents of the MMFR register are altered as the contents are serially shifted and are unpredictable if read by the user. After the read management frame operation completes, the MII interrupt is generated. At this time, the contents of the MMFR register match the original value written except for the DATA field whose contents are replaced by the value read from the PHY register.

If the MMFR register is written while frame generation is in progress, the frame contents are altered. Software must use the MII interrupt to avoid writing to the MMFR while frame generation is in progress.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ST		OP		PA					RA					TA		DATA															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- x = Undefined at reset.

FEC_MMFR field descriptions

Field	Description
0–1 ST	Start of frame delimiter. These bits must be programmed to 0b01 for a valid MII management frame.
2–3 OP	Operation code. 00 Write frame operation, but not MII compliant. 01 Write frame operation for a valid MII management frame. 10 Read frame operation for a valid MII management frame. 11 Read frame operation, but not MII compliant.
4–8 PA	PHY address. This field specifies one of up to 32 attached PHY devices.
9–13 RA	Register address. This field specifies one of up to 32 registers within the specified PHY device.
14–15 TA	Turn around. This field must be programmed to 10 to generate a valid MII management frame.
16–31 DATA	Management frame data. This is the field for data to be written to or read from the PHY register.

52.5.7 MII Speed Control Register (FEC_MSCR)

The MSCR:

- Provides control of the MII clock (FEC_MDC pin) frequency
- Allows a preamble drop on the MII management frame

The MII_SPEED field must be programmed with a value to provide an FEC_MDC frequency of less than or equal to 2.5 MHz to be compliant with the IEEE 802.3 MII specification. The MII_SPEED must be set to a non-zero value to source a read or write management frame. After the management frame is complete, the MSCR register may optionally be set to 0 to turn off the FEC_MDC. The FEC_MDC generated has a 50% duty cycle except when MII_SPEED changes during operation (change takes effect following a rising or falling edge of FEC_MDC).

If the internal bus clock is 25 MHz, programming this register to 0000_0005h results in an FEC_MDC as stated the equation below.

$$25\,\mathrm{MHz} \times \frac{1}{5 \times 2} = 2.5\,\mathrm{MHz}$$

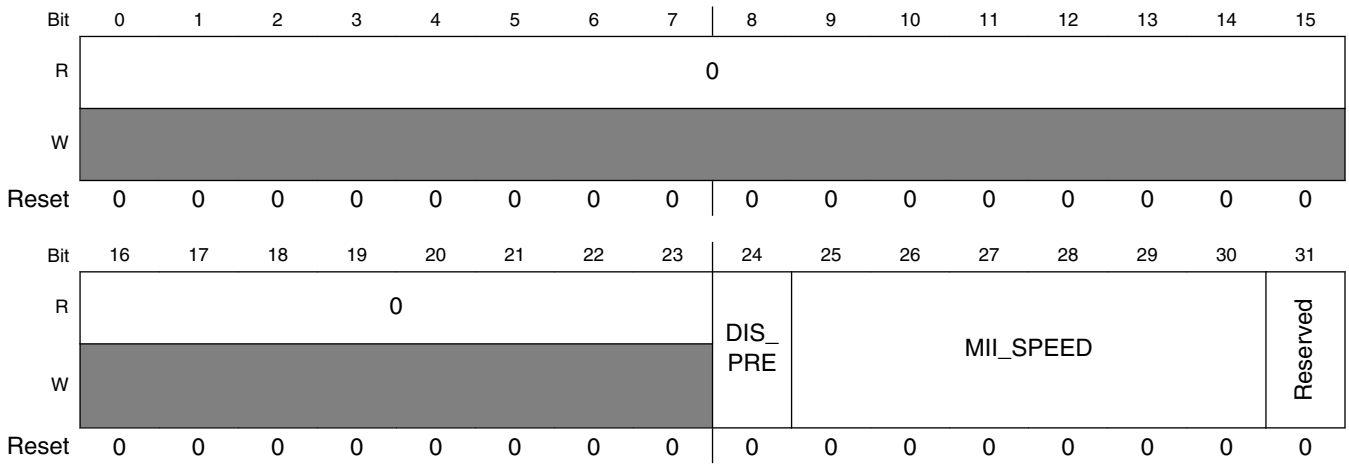
Figure 52-9. FEC_MDC example

Table 52-14 shows optimum values for MII_SPEED as a function of internal bus clock frequency.

Table 52-14. Programming examples for MSCR

Internal FEC clock frequency	MSCR[MII_SPEED]	FEC_MDC frequency
25 MHz	5h	2.50 MHz
33 MHz	7h	2.36 MHz
40 MHz	8h	2.50 MHz
50 MHz	Ah	2.50 MHz
66 MHz	Eh	2.36 MHz
80 MHz	10h	2.50 MHz
83 MHz	10h	2.59 MHz

Address: 0h base + 44h offset = 44h



FEC_MSCR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 DIS_PRE	Setting this bit causes the preamble (32 ones) not to be prepended to the MII management frame. The MII standard allows the preamble to be dropped if the attached PHY device(s) does not require it.
25–30 MII_SPEED	Controls the frequency of the MII management interface clock (FEC_MDC) relative to the internal bus clock. A value of 0 in this field turns off the FEC_MDC and leaves it in low voltage state. Any non-zero value results in the FEC_MDC frequency of 1/(MII_SPEED x 2) of the internal bus frequency.
31 Reserved	This field is reserved. Reserved

52.5.8 MIB Control Register (FEC_MIBC)

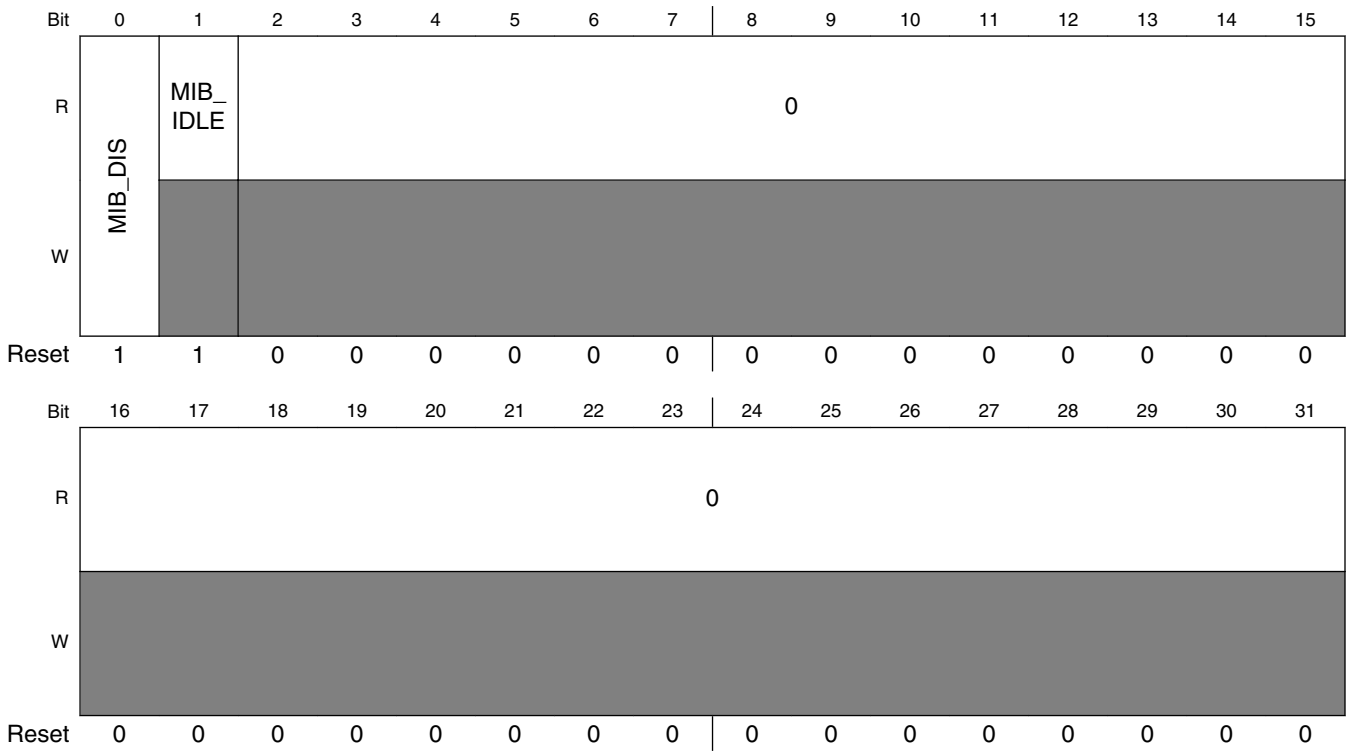
The MIBC is a read/write register controlling and observing the state of the MIB block. User software accesses this register if there is a need to disable the MIB block operation. For example, to clear all MIB counters in RAM:

1. Disable the MIB block

2. Clear all the MIB RAM locations
3. Enable the MIB block

The MIB_DIS bit is reset to 1. See the memory map for the locations of the MIB counters.

Address: 0h base + 64h offset = 64h



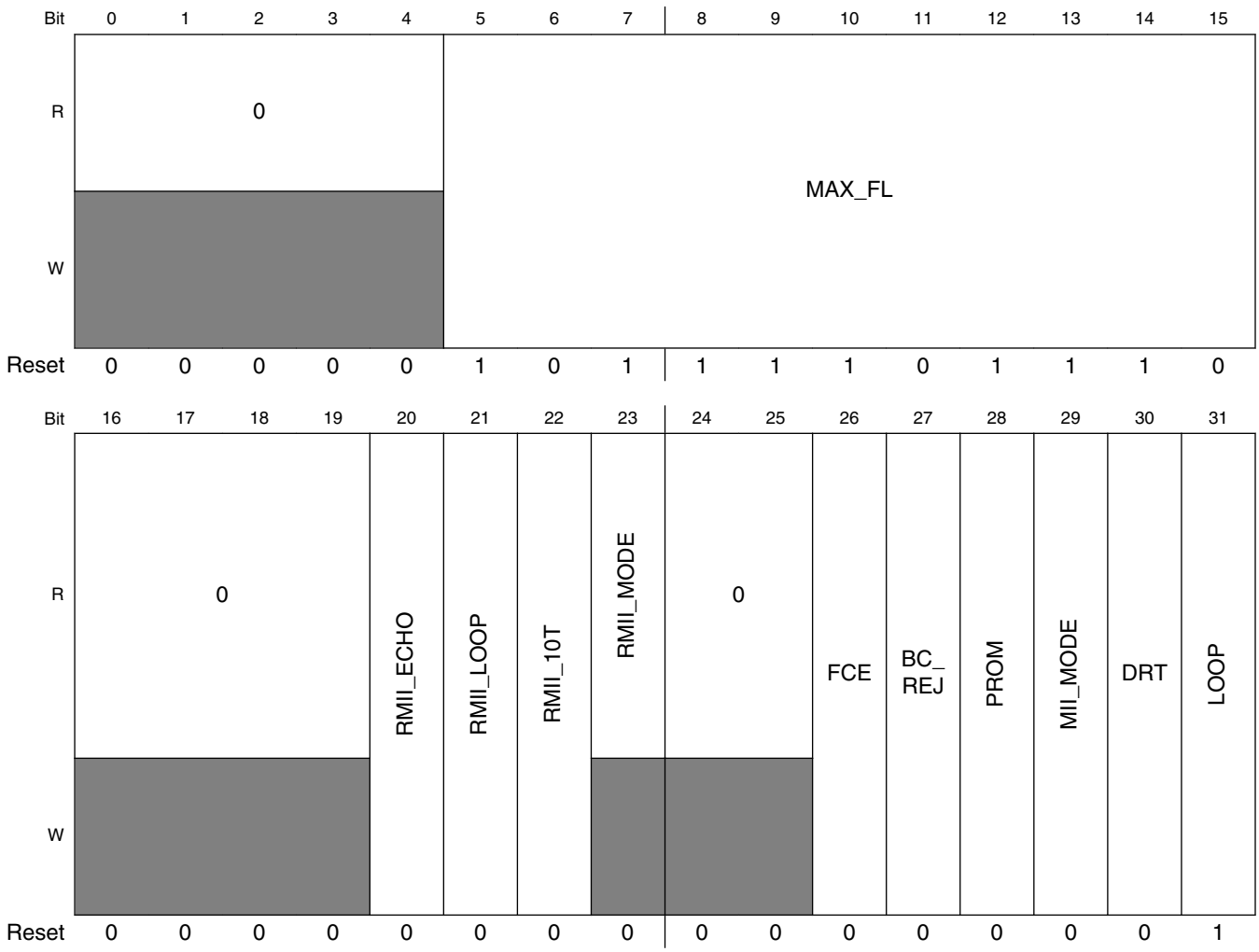
FEC_MIBC field descriptions

Field	Description
0 MIB_DIS	If MIB_DIS = 1, the MIB logic halts and not update any MIB counters.
1 MIB_IDLE	If MIB_IDLE = 1, the MIB block is not currently updating any MIB counters.
2–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

52.5.9 Receive Control Register (FEC_RCR)

RCR controls the operational mode of the receive block and must be written only when ECR[ETHER_EN] is cleared (initialization time).

Address: 0h base + 84h offset = 84h



FEC_RCR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 MAX_FL	Maximum frame length. Length is measured starting at DA and includes the CRC at the end of the frame. Transmit frames longer than MAX_FL causes the BABT interrupt to occur. Receive frames longer than MAX_FL causes the BABR interrupt to occur and sets the LG bit in the end of frame receive buffer descriptor.

Table continues on the next page...

FEC_RCR field descriptions (continued)

Field	Description
	The recommended default value to be programmed is 1518 or 1522 if VLAN tags are supported.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 RMII_ECHO	RMII Echo. Enables RMII echo mode. RMII 2-bit receive data is processed through the RMII receive logic to the FEC and also routes through the RMII's transmit logic to become RMII 2-bit transmit data out. 0 Normal operation. 1 RMII echo mode enabled. **NOTE:** When RMII_ECHO is set, proper operation is guaranteed only when RMII_MODE = 1, RMII_LOOP = 0 and LOOP = 0.
21 RMII_LOOP	RMII loopback. Enables RMII loopback mode. Causes the MII transmit outputs from the Ethernet controller to loop back to the Ethernet controllers's MII receive inputs through the RMII transmit/receive logic. 0 Normal operation. 1 RMII loopback mode enabled. **NOTE:** When RMII_LOOP is set, proper operation is guaranteed only when RMII_MODE = 1, RMII_ECHO = 0, LOOP = 0, and TCR *n* [FDEN] = 1.
22 RMII_10T	RMII 10-Base T. Enables 10Mbps mode of the RMII. Determines the clock frequency of the clock source to the FEC logic to support 10/100Mbps operations. 0 100 Mbps operation. The 50 MHz RMII reference clock on FEC_TXCLK is sent to the RMII, while a divided-by-2 version (25 MHz) is sent to the FEC. 1 10 Mbps operation. The 50 MHz RMII reference clock on FEC_TXCLK is divided by 10 (5 MHz) and sent to the RMII, while a divided-by-20 version (2.5 MHz) is sent to the FEC.
23 RMII_MODE	RMII Mode. Indicates if the FEC is in RMII or MII/7-wire mode. See the chip configuration chapter for more details. **NOTE:** Interface selection is chip-specific; see the chapter that describes how modules are configured and connected. 0 FEC is in true MII mode. 1 FEC is in RMII mode.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 FCE	Flow control enable. If asserted, the receiver detects PAUSE frames. Upon PAUSE frame detection, the transmitter stops transmitting data frames for a given duration.
27 BC_REJ	Broadcast frame reject. If asserted, frames with DA (destination address) equal to FFFF_FFFF_FFFF are rejected unless the PROM bit is set. If BC_REJ and PROM are set, frames with broadcast DA are accepted and the M (MISS) is set in the receive buffer descriptor.
28 PROM	Promiscuous mode. All frames are accepted regardless of address matching.
29 MII_MODE	Media independent interface mode. Selects the external interface mode for transmit and receive blocks. **NOTE:** Interface selection is chip-specific; see the chapter that describes how modules are configured and connected. 0 7-wire mode (used only for serial 10 Mbps) 1 MII or RMII mode as indicated by the RMII_MODE bit

Table continues on the next page...

FEC_RCR field descriptions (continued)

Field	Description
30 DRT	Disable receive on transmit. 0 Receive path operates independently of transmit (use for full duplex or to monitor transmit activity in half duplex mode). 1 Disable reception of frames while transmitting (normally used for half duplex mode).
31 LOOP	Internal loopback. If set, transmitted frames are looped back internal to the device and transmit output signals are not asserted. The internal bus clock substitutes for the FEC_TXCLK when LOOP is asserted. DRT must be set to 0 when setting LOOP.

52.5.10 Transmit Control Register (FEC_TCR)

The TCR configures the transmit block.

Address: 0h base + C4h offset = C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											RFC_PAUSE	TFC_PAUSE	FDEN	HBC	GTS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_TCR field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 RFC_PAUSE	Receive frame control pause. This read-only status bit is asserted when a full duplex flow control pause frame is received and the transmitter pauses for the duration defined in this pause frame. This bit automatically clears when the pause duration is complete.
28 TFC_PAUSE	Transmit frame control pause. Transmits a PAUSE frame when asserted. When this bit is set, the MAC stops transmission of data frames after the current transmission is complete. At this time, GRA interrupt in the EIR is asserted. With transmission of data frames stopped, MAC transmits a MAC Control PAUSE frame. Next, the MAC clears the TFC_PAUSE bit and resumes transmitting data frames. If the transmitter pauses due to user assertion of GTS or reception of a PAUSE frame, the MAC may continue transmitting a MAC Control PAUSE frame.
29 FDEN	Full duplex enable. If set, frames transmit independent of carrier sense and collision inputs. This field must be modified only when ECR[ETHER_EN] = 0.
30 HBC	Heartbeat control. If set, the heartbeat check performs following end of transmission and the HB bit in the status register is set if the collision input does not assert within the heartbeat window. This field must be modified only when ECR[ETHER_EN] = 0.
31 GTS	Graceful transmit stop. When this bit is set, MAC stops transmission after any frame currently transmitted is complete and GRA interrupt in the EIR is asserted. If frame transmission is not currently underway, the GRA interrupt is asserted immediately. After transmission finishes, clear GTS to restart. The next frame in the transmit FIFO is then transmitted. If an early collision occurs during transmission when GTS is set, transmission stops after the collision. The frame is transmitted again after GTS is cleared. There may be old frames in the transmit FIFO that transmit when GTS is reasserted. To avoid this situation, program ECR[ETHER_EN] = 0 following the GRA interrupt.

52.5.11 Physical Address Low Register (FEC_PALR)

PALR contains the lower 32 bits (bytes 0, 1, 2, 3) of the 48-bit address used in the address recognition process to compare with the DA (destination address) field of receive frames with an individual DA. In addition, this register is used in bytes 0 through 3 of the 6-byte source address field when transmitting PAUSE frames.

This register is not reset and you must initialize it.

Address: 0h base + E4h offset = E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PADDR1																															
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:
- u = Unaffected by reset.

FEC_PALR field descriptions

Field	Description
0–31 PADDR1	Bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8), and 3 (bits 7:0) of the 6-byte individual address are used for exact match and the source address field in PAUSE frames.

52.5.12 Physical Address High Register and Type Field (FEC_PAUR)

PAUR contains the upper 16 bits (bytes 4 and 5) of the 48-bit address used in the address recognition process to compare with the DA (destination address) field of receive frames with an individual DA. In addition, this register is used in bytes 4 and 5 of the 6-byte Source Address field when transmitting PAUSE frames. Bits 15:0 of PAUR contain a constant type field (8808h) for transmission of PAUSE frames. The upper 16 bits of this register are not reset and you must initialize them.

Address: 0h base + E8h offset = E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PADDR2																TYPE															
W																																
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

* Notes:
- u = Unaffected by reset.

FEC_PAUR field descriptions

Field	Description
0–15 PADDR2	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address used for exact match, and the source address field in PAUSE frames.
16–31 TYPE	Type field in PAUSE frames. These 16 read-only bits are a constant value of 8808h.

52.5.13 Opcode/Pause Duration (FEC_OPD)

The OPD contains the 16-bit opcode and 16-bit pause duration fields used in transmission of a PAUSE frame. The opcode field is a constant value, 0001h. When another node detects a PAUSE frame, that node pauses transmission for the duration specified in the pause duration field.

The lower 16 bits of this register are not reset and you must initialize them.

Address: 0h base + ECh offset = ECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OPCODE																PAUSE_DUR															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_OPD field descriptions

Field	Description
0–15 OPCODE	Opcode field used in PAUSE frames. These read-only bits are a constant, 0001h.
16–31 PAUSE_DUR	Pause Duration field used in PAUSE frames.

52.5.14 Descriptor Individual Upper Address Register (FEC_IAUR)

IAUR contains the upper 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the destination address (DA) field of receive frames with an individual DA.

This register is not reset and you must initialize it.

Address: 0h base + 118h offset = 118h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IADDR1																															
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:
- u = Unaffected by reset.

FEC_IAUR field descriptions

Field	Description
0–31 IADDR1	The upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR1 contains hash index bit 63. Bit 0 of IADDR1 contains hash index bit 32.

52.5.15 Descriptor Individual Lower Address Register (FEC_IALR)

IALR contains the lower 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the DA field of receive frames with an individual DA.

This register is not reset and you must initialize it.

Address: 0h base + 11Ch offset = 11Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IADDR2																															
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:
- u = Unaffected by reset.

FEC_IALR field descriptions

Field	Description
0–31 IADDR2	The lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR2 contains hash index bit 31. Bit 0 of IADDR2 contains hash index bit 0.

52.5.16 Descriptor Group Upper Address Register (FEC_GAUR)

GAUR contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address.

This register is not reset and you must initialize it.

Address: 0h base + 120h offset = 120h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	GADDR1																															
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:
- u = Unaffected by reset.

FEC_GAUR field descriptions

Field	Description
0–31 GADDR1	The GADDR1 register contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR1 contains hash index bit 63. Bit 0 of GADDR1 contains hash index bit 32.

52.5.17 Descriptor Group Lower Address Register (FEC_GALR)

GALR contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address.

This register is not reset and you must initialize it.

Address: 0h base + 124h offset = 124h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	GADDR2																															
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:
- u = Unaffected by reset.

FEC_GALR field descriptions

Field	Description
0–31 GADDR2	The GADDR2 register contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR2 contains hash index bit 31. Bit 0 of GADDR2 contains hash index bit 0.

52.5.18 Transmit FIFO Watermark (FEC_TFWR)

The TFWR controls the amount of data required in the transmit FIFO before transmission of a frame can begin. This allows you to minimize transmit latency (TFWR = 00 or 01) or allow for larger bus access latency (TFWR = 11) due to contention for the system bus. Setting the watermark to a high value minimizes the risk of transmit FIFO underrun due to contention for the system bus. The byte counts associated with the TFWR field may need to be modified to match a given system requirement (worst case bus access latency by the transmit data DMA channel).

Address: 0h base + 144h offset = 144h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0														TFWR	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FEC_TFWR field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 TFWR	Number of bytes written to transmit FIFO before transmission of a frame begins 00 64 bytes written

Table continues on the next page...

FEC_TFWR field descriptions (continued)

Field	Description
	01 64 bytes written 10 128 bytes written 11 192 bytes written

52.5.19 FIFO Receive Bound Register (FEC_FRBR)

FRBR indicates the upper address bound of the FIFO RAM. Drivers can use this value, along with the FRSR, to appropriately divide the available FIFO RAM between the transmit and receive data paths.

Address: 0h base + 14Ch offset = 14Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					1	R_BOUND								0	
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

FEC_FRBR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
22–29 R_BOUND	Highest valid FIFO RAM address.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

52.5.20 FIFO Receive Start Register (FEC_FRSR)

FRSR indicates the starting address of the receive FIFO. FRSR marks the boundary between the transmit and receive FIFOs. The transmit FIFO uses addresses from the start of the FIFO to the location four bytes before the address programmed into the FRSR. The receive FIFO uses addresses from FRSR to FRBR inclusive.

Hardware initializes the FRSR at reset. FRSR only needs to be written to change the default value.

Address: 0h base + 150h offset = 150h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					1	R_FSTART								0	
W																
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0

FEC_FRSR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
22–29 R_FSTART	Address of first receive FIFO location. Acts as delimiter between receive and transmit FIFOs. For proper operation, ensure that R_FSTART is set to 48h or greater.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

52.5.21 Receive Descriptor Ring Start Register (FEC_ERDSR)

ERDSR points to the start of the circular receive buffer descriptor queue in external memory. This pointer must be 32-bit aligned; however, it is recommended it be made 128-bit aligned (evenly divisible by 16). You should write zeros to bits 1 and 0. Hardware ignores non-zero values in these two bit positions.

This register is undefined at reset and must be initialized prior to operation.

Address: 0h base + 180h offset = 180h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	R_DES_START																														0	
W																																
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:
- u = Unaffected by reset.

FEC_ERDSR field descriptions

Field	Description
0–29 R_DES_START	Pointer to start of receive buffer descriptor queue.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

52.5.22 Transmit Buffer Descriptor Ring Start Register (FEC_ETDSR)

ETSDR provides a pointer to the start of the circular transmit buffer descriptor queue in external memory. This pointer must be 32-bit aligned; however, it is recommended it be made 128-bit aligned (evenly divisible by 16). You should write zeros to bits 1 and 0. Hardware ignores non-zero values in these two bit positions.

This register is undefined at reset and must be initialized prior to operation.

Address: 0h base + 184h offset = 184h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	X_DES_START																														0	
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:
- u = Unaffected by reset.

FEC_ETDSR field descriptions

Field	Description
0–29 X_DES_START	Pointer to start of transmit buffer descriptor queue.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

52.5.23 Receive Buffer Size Register (FEC_EMRBR)

The EMRBR dictates the maximum size of all receive buffers. This value should take into consideration that the receive CRC is always written into the last receive buffer. To allow one maximum size frame per buffer, EMRBR must be set to RCR[MAX_FL] or larger. To properly align the buffer, EMRBR must be evenly divisible by 16. To ensure this, bits 3-0 are forced low.

To minimize bus utilization (descriptor fetches), it is recommended that EMRBR be greater than or equal to 256 bytes.

The EMRBR is undefined at reset and must be initialized by the user.

Address: 0h base + 188h offset = 188h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																R_BUF_SIZE												0			
W																																
Reset	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*	u*

* Notes:
- u = Unaffected by reset.

FEC_EMRBR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–27 R_BUF_SIZE	Maximum size of receive buffer size in bytes. To minimize bus utilization (descriptor fetches), set this field to 256 bytes (10h) or larger. 10h: 256 + 15 bytes (minimum size recommended) 11h: 272 + 15 bytes ... 7Fh: 2032 + 15 bytes. The FEC writes up to 2047 bytes in the receive buffer. If data larger than 2047 is received, the FEC truncates it and shows 7FFh in the receive descriptor
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

52.5.24 Count of frames not counted correctly (FEC_RMON_T_DROP)

See Using the RMON and IEEE registers.

Address: 0h base + 200h offset = 200h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_DROP																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_DROP field descriptions

Field	Description
0–31 RMON_T_DROP	Count of frames not counted correctly

52.5.25 RMON Tx packet count (FEC_RMON_T_PACKETS)

See Using the RMON and IEEE registers.

Address: 0h base + 204h offset = 204h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_PACKETS																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_PACKETS field descriptions

Field	Description
0–31 RMON_T_ PACKETS	RMON Tx packet count

52.5.26 RMON Tx broadcast packets (FEC_RMON_T_BC_PKT)

See Using the RMON and IEEE registers.

Address: 0h base + 208h offset = 208h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_BC_PKT																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_BC_PKT field descriptions

Field	Description
0–31 RMON_T_BC_ PKT	RMON Tx broadcast packets

52.5.27 RMON Tx multicast packets (FEC_RMON_T_MC_PKT)

See Using the RMON and IEEE registers.

Address: 0h base + 20Ch offset = 20Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_MC_PKT																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_MC_PKT field descriptions

Field	Description
0–31 RMON_T_MC_ PKT	RMON Tx multicast packets

52.5.28 RMON Tx packets with CRC/align error (FEC_RMON_T_CRC_ALIGN)

See Using the RMON and IEEE registers.

Address: 0h base + 210h offset = 210h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_CRC_ALIGN																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_CRC_ALIGN field descriptions

Field	Description
0–31 RMON_T_CRC_ ALIGN	RMON Tx packets with CRC/align error

52.5.29 RMON Tx packets < 64 bytes, good CRC (FEC_RMON_T_UNDERSIZE)

See Using the RMON and IEEE registers.

Address: 0h base + 214h offset = 214h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_UNDERSIZE																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_UNDERSIZE field descriptions

Field	Description
0–31 RMON_T_ UNDERSIZE	RMON Tx packets < 64 bytes, good CRC

52.5.30 RMON Tx packets > MAX_FL bytes, good CRC (FEC_RMON_T_OVERSIZE)

See Using the RMON and IEEE registers.

Address: 0h base + 218h offset = 218h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_OVERSIZE																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_OVERSIZE field descriptions

Field	Description
0–31 RMON_T_ OVERSIZE	RMON Tx packets > MAX_FL bytes, good CRC

52.5.31 RMON Tx packets < 64 bytes, bad CRC (FEC_RMON_T_FRAG)

See Using the RMON and IEEE registers.

Address: 0h base + 21Ch offset = 21Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_FRAG																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_FRAG field descriptions

Field	Description
0–31 RMON_T_FRAG	RMON Tx packets < 64 bytes, bad CRC

52.5.32 RMON Tx packets > MAX_FL bytes, bad CRC (FEC_RMON_T_JAB)

See Using the RMON and IEEE registers.

Address: 0h base + 220h offset = 220h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_JAB																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_JAB field descriptions

Field	Description
0–31 RMON_T_JAB	RMON Tx packets > MAX_FL bytes, bad CRC

52.5.33 RMON Tx collision count (FEC_RMON_T_COL)

See Using the RMON and IEEE registers.

Address: 0h base + 224h offset = 224h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_COL																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_COL field descriptions

Field	Description
0–31 RMON_T_COL	RMON Tx collision count

52.5.34 RMON Tx 64 byte packets (FEC_RMON_T_P64)

See Using the RMON and IEEE registers.

Address: 0h base + 228h offset = 228h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_P64																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_P64 field descriptions

Field	Description
0–31 RMON_T_P64	RMON Tx 64 byte packets

52.5.35 RMON Tx 65 to 127 byte packets (FEC_RMON_T_P65TO127)

See Using the RMON and IEEE registers.

Address: 0h base + 22Ch offset = 22Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_P65TO127																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_P65TO127 field descriptions

Field	Description
0–31 RMON_T_ P65TO127	RMON Tx 65 to 127 byte packets

52.5.36 RMON Tx 128 to 255 byte packets (FEC_RMON_T_P128TO255)

See Using the RMON and IEEE registers.

Address: 0h base + 230h offset = 230h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_P128TO255																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_P128TO255 field descriptions

Field	Description
0–31 RMON_T_ P128TO255	RMON Tx 128 to 255 byte packets

52.5.37 RMON Tx 256 to 511 byte packets (FEC_RMON_T_P256TO511)

See Using the RMON and IEEE registers.

Address: 0h base + 234h offset = 234h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_P256TO511																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_P256TO511 field descriptions

Field	Description
0–31 RMON_T_ P256TO511	RMON Tx 256 to 511 byte packets

52.5.38 RMON Tx 512 to 1023 byte packets (FEC_RMON_T_P512TO1023)

See Using the RMON and IEEE registers.

Address: 0h base + 238h offset = 238h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_P512TO1023																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_P512TO1023 field descriptions

Field	Description
0–31 RMON_T_ P512TO1023	RMON Tx 512 to 1023 byte packets

52.5.39 RMON Tx 1024 to 2047 byte packets (FEC_RMON_T_P1024TO2047)

See Using the RMON and IEEE registers.

Address: 0h base + 23Ch offset = 23Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_P1024TO2047																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_P1024TO2047 field descriptions

Field	Description
0–31 RMON_T_ P1024TO2047	RMON Tx 1024 to 2047 byte packets

52.5.40 RMON Tx packets with > 2048 bytes (FEC_RMON_T_P_GTE2048)

See Using the RMON and IEEE registers.

Address: 0h base + 240h offset = 240h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_P_GTE2048																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_P_GTE2048 field descriptions

Field	Description
0–31 RMON_T_P_ GTE2048	RMON Tx packets with > 2048 bytes

52.5.41 RMON Tx Octets (FEC_RMON_T_OCTETS)

See Using the RMON and IEEE registers.

Address: 0h base + 244h offset = 244h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_T_OCTETS																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_T_OCTETS field descriptions

Field	Description
0–31 RMON_T_ OCTETS	RMON Tx Octets

52.5.42 Count of transmitted frames not counted correctly (FEC_IEEE_T_DROP)

See Using the RMON and IEEE registers.

Address: 0h base + 248h offset = 248h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_T_DROP																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_T_DROP field descriptions

Field	Description
0–31 IEEE_T_DROP	Count of transmitted frames not counted correctly

52.5.43 Frames transmitted OK (FEC_IEEE_T_FRAME_OK)

See Using the RMON and IEEE registers.

Address: 0h base + 24Ch offset = 24Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_T_FRAME_OK																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_T_FRAME_OK field descriptions

Field	Description
0–31 IEEE_T_ FRAME_OK	Frames transmitted OK

52.5.44 Frames transmitted with single collision (FEC_IEEE_T_1COL)

See Using the RMON and IEEE registers.

Address: 0h base + 250h offset = 250h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_T_1COL																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_T_1COL field descriptions

Field	Description
0–31 IEEE_T_1COL	Frames transmitted with single collision

52.5.45 Frames transmitted with multiple collisions (FEC_IEEE_T_MCOL)

See Using the RMON and IEEE registers.

Address: 0h base + 254h offset = 254h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_T_MCOL																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_T_MCOL field descriptions

Field	Description
0–31 IEEE_T_MCOL	Frames transmitted with multiple collisions

52.5.46 Frames transmitted after deferral delay (FEC_IEEE_T_DEF)

See Using the RMON and IEEE registers.

Address: 0h base + 258h offset = 258h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_T_DEF																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_T_DEF field descriptions

Field	Description
0–31 IEEE_T_DEF	Frames transmitted after deferral delay

52.5.47 Frames transmitted with late collision (FEC_IEEE_T_LCOL)

See Using the RMON and IEEE registers.

Address: 0h base + 25Ch offset = 25Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_T_LCOL																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_T_LCOL field descriptions

Field	Description
0–31 IEEE_T_LCOL	Frames transmitted with late collision

52.5.48 Frames transmitted with excessive collisions (FEC_IEEE_T_EXCOL)

See Using the RMON and IEEE registers.

Address: 0h base + 260h offset = 260h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_T_EXCOL																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_T_EXCOL field descriptions

Field	Description
0–31 IEEE_T_EXCOL	Frames transmitted with excessive collisions

52.5.49 Frames transmitted with Tx FIFO underrun (FEC_IEEE_T_MACERR)

See Using the RMON and IEEE registers.

Address: 0h base + 264h offset = 264h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_T_MACERR																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_T_MACERR field descriptions

Field	Description
0–31 IEEE_T_ MACERR	Frames transmitted with Tx FIFO underrun

52.5.50 Frames transmitted with carrier sense error (FEC_IEEE_T_CSERR)

See Using the RMON and IEEE registers.

Address: 0h base + 268h offset = 268h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_T_CSERR																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_T_CSERR field descriptions

Field	Description
0–31 IEEE_T_CSERR	Frames transmitted with carrier sense error

52.5.51 Frames transmitted with SQE error (FEC_IEEE_T_SQE)

See Using the RMON and IEEE registers.

Address: 0h base + 26Ch offset = 26Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_T_SQE																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_T_SQE field descriptions

Field	Description
0–31 IEEE_T_SQE	Frames transmitted with SQE error

52.5.52 Flow control pause frames transmitted (FEC_IEEE_T_FDXFC)

See Using the RMON and IEEE registers.

Address: 0h base + 270h offset = 270h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_T_FDXFC																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_T_FDXFC field descriptions

Field	Description
0–31 IEEE_T_FDXFC	Flow control pause frames transmitted

52.5.53 Octet count for frames transmitted without error (FEC_IEEE_T_OCTETS_OK)

See Using the RMON and IEEE registers.

Address: 0h base + 274h offset = 274h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_T_OCTETS_OK																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_T_OCTETS_OK field descriptions

Field	Description
0–31 IEEE_T_ OCTETS_OK	Octet count for frames transmitted without error

52.5.54 Count of received frames not counted correctly (FEC_RMON_R_DROP)

See Using the RMON and IEEE registers.

Address: 0h base + 280h offset = 280h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_DROP																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_DROP field descriptions

Field	Description
0–31 RMON_R_DROP	Count of received frames not counted correctly

52.5.55 RMON Rx packet count (FEC_RMON_R_PACKETS)

See Using the RMON and IEEE registers.

Address: 0h base + 284h offset = 284h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_PACKETS																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_PACKETS field descriptions

Field	Description
0–31 RMON_R_ PACKETS	RMON Rx packet count

52.5.56 RMON Rx broadcast packets (FEC_RMON_R_BC_PKT)

See Using the RMON and IEEE registers.

Address: 0h base + 288h offset = 288h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_BC_PKT																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_BC_PKT field descriptions

Field	Description
0–31 RMON_R_BC_ PKT	RMON Rx broadcast packets

52.5.57 RMON Rx multicast packets (FEC_RMON_R_MC_PKT)

See Using the RMON and IEEE registers.

Address: 0h base + 28Ch offset = 28Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_MC_PKT																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_MC_PKT field descriptions

Field	Description
0–31 RMON_R_MC_PKT	RMON Rx multicast packets

52.5.58 RMON Rx packets with CRC/Align error (FEC_RMON_R_CRC_ALIGN)

See Using the RMON and IEEE registers.

Address: 0h base + 290h offset = 290h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_CRC_ALIGN																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_CRC_ALIGN field descriptions

Field	Description
0–31 RMON_R_CRC_ALIGN	RMON Rx packets with CRC/Align error

52.5.59 RMON Rx packets < 64 bytes, good CRC (FEC_RMON_R_UNDERSIZE)

See Using the RMON and IEEE registers.

Address: 0h base + 294h offset = 294h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_UNDERSIZE																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_UNDERSIZE field descriptions

Field	Description
0–31 RMON_R_ UNDERSIZE	RMON Rx packets < 64 bytes, good CRC

52.5.60 RMON Rx packets > MAX_FL bytes, good CRC (FEC_RMON_R_OVERSIZE)

See Using the RMON and IEEE registers.

Address: 0h base + 298h offset = 298h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_OVERSIZE																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_OVERSIZE field descriptions

Field	Description
0–31 RMON_R_ OVERSIZE	RMON Rx packets > MAX_FL bytes, good CRC

52.5.61 RMON Rx packets < 64 bytes, bad CRC (FEC_RMON_R_FRAG)

See Using the RMON and IEEE registers.

Address: 0h base + 29Ch offset = 29Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_FRAG																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_FRAG field descriptions

Field	Description
0–31 RMON_R_FRAG	RMON Rx packets < 64 bytes, bad CRC

52.5.62 RMON Rx packets > MAX_FL bytes, bad CRC (FEC_RMON_R_JAB)

See Using the RMON and IEEE registers.

Address: 0h base + 2A0h offset = 2A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_JAB																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_JAB field descriptions

Field	Description
0–31 RMON_R_JAB	RMON Rx packets > MAX_FL bytes, bad CRC

52.5.63 Reserved (FEC_RMON_R_RESVD_0)

See Using the RMON and IEEE registers.

Address: 0h base + 2A4h offset = 2A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_RESVD_0																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_RESVD_0 field descriptions

Field	Description
0–31 RMON_R_ RESVD_0	Reserved

52.5.64 RMON Rx 64 byte packets (FEC_RMON_R_P64)

See Using the RMON and IEEE registers.

Address: 0h base + 2A8h offset = 2A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_P64																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_P64 field descriptions

Field	Description
0–31 RMON_R_P64	RMON Rx 64 byte packets

52.5.65 RMON Rx 65 to 127 byte packets (FEC_RMON_R_P65TO127)

See Using the RMON and IEEE registers.

Address: 0h base + 2ACh offset = 2ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_P65TO127																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_P65TO127 field descriptions

Field	Description
0–31 RMON_R_ P65TO127	RMON Rx 65 to 127 byte packets

52.5.66 RMON Rx 128 to 255 byte packets (FEC_RMON_R_P128TO255)

See Using the RMON and IEEE registers.

Address: 0h base + 2B0h offset = 2B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_P128TO255																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_P128TO255 field descriptions

Field	Description
0–31 RMON_R_ P128TO255	RMON Rx 128 to 255 byte packets

52.5.67 RMON Rx 256 to 511 byte packets (FEC_RMON_R_P256TO511)

See Using the RMON and IEEE registers.

Address: 0h base + 2B4h offset = 2B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_P256TO511																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_P256TO511 field descriptions

Field	Description
0–31 RMON_R_ P256TO511	RMON Rx 256 to 511 byte packets

52.5.68 RMON Rx 512 to 1023 byte packets (FEC_RMON_R_P512TO1023)

See Using the RMON and IEEE registers.

Address: 0h base + 2B8h offset = 2B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_P512TO1023																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_P512TO1023 field descriptions

Field	Description
0–31 RMON_R_ P512TO1023	RMON Rx 512 to 1023 byte packets

52.5.69 RMON Rx 1024 to 2047 byte packets (FEC_RMON_R_P1024TO2047)

See Using the RMON and IEEE registers.

Address: 0h base + 2BCh offset = 2BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_P1024TO2047																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_P1024TO2047 field descriptions

Field	Description
0–31 RMON_R_ P1024TO2047	RMON Rx 1024 to 2047 byte packets

52.5.70 RMON Rx packets with > 2048 bytes (FEC_RMON_R_P_GTE2048)

See Using the RMON and IEEE registers.

Address: 0h base + 2C0h offset = 2C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_P_GTE2048																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_P_GTE2048 field descriptions

Field	Description
0–31 RMON_R_P_ GTE2048	RMON Rx packets with > 2048 bytes

52.5.71 RMON Rx octets (FEC_RMON_R_OCTETS)

See Using the RMON and IEEE registers.

Address: 0h base + 2C4h offset = 2C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RMON_R_OCTETS																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_RMON_R_OCTETS field descriptions

Field	Description
0–31 RMON_R_ OCTETS	RMON Rx octets

52.5.72 Count of received frames not counted correctly (FEC_IEEE_R_DROP)

See Using the RMON and IEEE registers.

Address: 0h base + 2C8h offset = 2C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_R_DROP																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_R_DROP field descriptions

Field	Description
0–31 IEEE_R_DROP	Count of received frames not counted correctly

52.5.73 Frames received OK (FEC_IEEE_R_FRAME_OK)

See Using the RMON and IEEE registers.

Address: 0h base + 2CCh offset = 2CCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_R_FRAME_OK																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_R_FRAME_OK field descriptions

Field	Description
0–31 IEEE_R_ FRAME_OK	Frames received OK

52.5.74 Frames received with CRC error (FEC_IEEE_R_CRC)

See Using the RMON and IEEE registers.

Address: 0h base + 2D0h offset = 2D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_R_CRC																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_R_CRC field descriptions

Field	Description
0–31 IEEE_R_CRC	Frames received with CRC error

52.5.75 Frames received with alignment error (FEC_IEEE_R_ALIGN)

See Using the RMON and IEEE registers.

Address: 0h base + 2D4h offset = 2D4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_R_ALIGN																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_R_ALIGN field descriptions

Field	Description
0–31 IEEE_R_ALIGN	Frames received with alignment error

52.5.76 Receive FIFO overflow count (FEC_IEEE_R_MACERR)

See Using the RMON and IEEE registers.

Address: 0h base + 2D8h offset = 2D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_R_MACERR																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_R_MACERR field descriptions

Field	Description
0–31 IEEE_R_ MACERR	Receive FIFO overflow count

52.5.77 Flow control pause frames received (FEC_IEEE_R_FDXFC)

See Using the RMON and IEEE registers.

Address: 0h base + 2DCh offset = 2DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_R_FDXFC																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_R_FDXFC field descriptions

Field	Description
0–31 IEEE_R_FDXFC	Flow control pause frames received

52.5.78 Octet count for frames received without error (FEC_IEEE_R_OCTETS_OK)

See Using the RMON and IEEE registers.

Address: 0h base + 2E0h offset = 2E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IEEE_R_OCTETS_OK																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

FEC_IEEE_R_OCTETS_OK field descriptions

Field	Description
0–31 IEEE_R_ OCTETS_OK	Octet count for frames received without error

52.5.79 Using the RMON and IEEE registers

In order to use the RMON and IEEE registers to monitor FEC performance, you must prepare them using the sequence below:

1. Write zeros to the registers.
2. Enable the FEC.

Until you complete this sequence, the registers are read/write, and their reset value is undefined. After you complete this sequence, the registers become read-only, and you may read them as necessary.

52.6 Functional description

This section describes the operation of the FEC, beginning with the description of two different types of MII frames, the buffer descriptors, the hardware and software initialization sequence, then the software (Ethernet driver) interface for transmitting and receiving frames.

Following the software initialization and operation sections are sections providing a detailed description of the functions of the FEC.

52.6.1 MII data frame

Ethernet/802.3 data frames transmitted across the MII have the format specified in Figure 52-81 and Table 52-86.

<inter-frame> <preamble> <sfd> <data> <efd>

Figure 52-81. MII data frame format

Table 52-86. MII data frame components

Component	Description
<inter-frame>	Inter-frame period. This is an unspecified amount of time during which no data activity occurs on the MII. The de-assertion of RX_DV and TX_EN indicate the absence of data activity.
<preamble>	Preamble that begins a frame. It has the following value: 10101010 10101010 10101010 10101010 10101010 10101010 10101010 The leftmost 1 represents the LSB of the byte.
<sfd>	Start of a frame. It has the value 10101011.
<data>	Data portion of the frame. It consists of N octets which corresponds to 2N nibbles being transmitted. The order of each nibble is defined in Figure 52-82.

Table continues on the next page...

Table 52-86. MII data frame components (continued)

Component	Description
<efd>	End-of-frame delimiter. For TXD data, it is indicated by the de-assertion of the TX_EN signal. For RXD data, it is indicated by the de-assertion of the RX_DV signal.

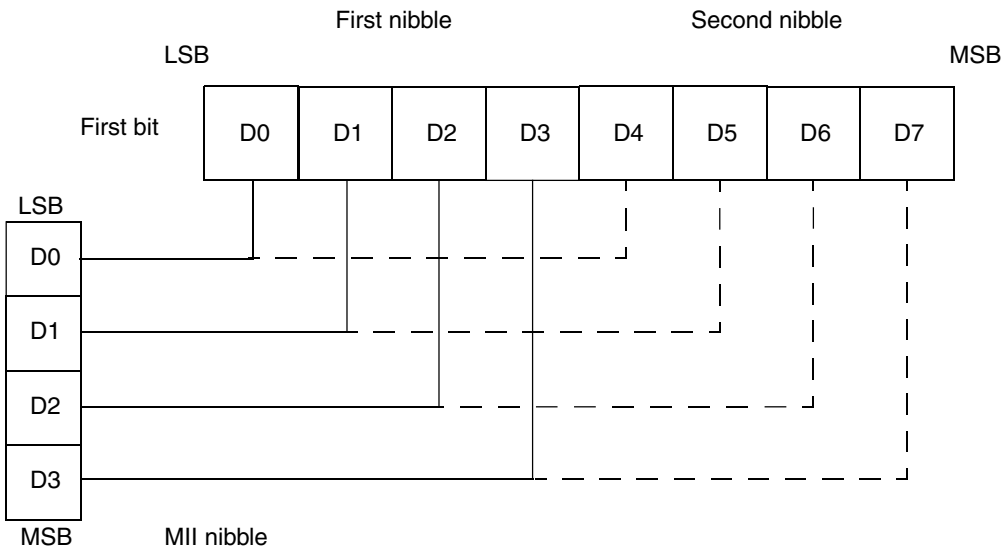


Figure 52-82. MII nibble/octet to octet/nibble mapping

52.6.2 MII management frame structure

A transceiver management frame being transmitted on the MII management interface uses the MDIO and MDC pins. A transaction or frame on this serial interface has the format specified in Figure 52-83 and Table 52-87.

<preamble> <st> <op> <phyad> <regad> <ta> <data> <idle>

Figure 52-83. MII management frame format

Table 52-87. MII management frame components

Component	Description
<preamble>	Optional preamble, consisting of a sequence of 32 continuous logic 1's.
<st>	Start of frame, consisting of a 01 pattern.
<op>	Operation code. 01 Write operation 10 Read instruction

Table continues on the next page...

Table 52-87. MII management frame components (continued)

Component	Description
<phyad>	Five-bit field that allows up to 32 PHYs to be addressed. The first address bit transmitted is the MSB of the address.
<regad>	Five-bit field that allows for 32 registers to be addressed within each PHY. The first register bit transmitted is the MSB of the address.
<ta>	Two-bit field that provides spacing between the register address field and the data field to avoid contention on the MDIO signal during a read operation.
<data>	16-bit data field. Bit 15 is the first data bit transmitted and received.
<idle>	Idle condition (MDIO is in the high-impedance state).

The MII management register set located in the PHY may consist of a basic register set and an extended register set as defined in Table 52-88.

Table 52-88. MII management register set

Register address	Register name	Register set
0	Control	Basic
1	Status	
2-3	PHY identifier	Extended
4	Auto-negotiation advertisement	
5	AN link partner ability	
6	AN expansion	
7	AN next page transmit	
8-15	Reserved	
16-31	Vendor-specific	

52.6.3 Buffer descriptors

This section provides a description of the operation of the driver/DMA via the buffer descriptors. It is followed by a detailed description of the receive and transmit descriptor fields.

52.6.3.1 Driver/DMA operation with buffer descriptors

The data for the FEC frames resides in one or more memory buffers external to the FEC. Associated with each buffer is a buffer descriptor (BD), which contains a starting address (32-bit aligned pointer), data length, and status/control information (which contains the current state for the buffer). To permit maximum user flexibility, the BDs are also located in external memory and are read by the FEC DMA engine.

Software produces buffers by allocating/initializing memory and initializing buffer descriptors. Setting the RxBD[E] or TxBD[R] bit produces the buffer. Software writing to TDAR or RDAR tells the FEC that a buffer is placed in external memory for the transmit or receive data traffic, respectively. The hardware reads the BDs and consumes the buffers after they have been produced. After the data DMA is complete and the DMA engine writes the buffer descriptor status bits, hardware clears RxBD[E] or TxBD[R] to signal the buffer has been consumed. Software may poll the BDs to detect when the buffers are consumed or may rely on the buffer/frame interrupts. The driver may process these buffers, and they can return to the free list.

The ECR[ETHER_EN] bit operates as a reset to the BD/DMA logic. When ECR[ETHER_EN] is cleared, the DMA engine BD pointers are reset to point to the starting transmit and receive BDs. The buffer descriptors are not initialized by hardware during reset. At least one transmit and receive buffer descriptor must be initialized by software before ECR[ETHER_EN] is set.

The buffer descriptors operate as two separate rings. ERDSR defines the starting address for receive BDs and ETDSR defines the starting address for transmit BDs. The wrap (W) bit defines the last buffer descriptor in each ring. When W is set, the next descriptor in the ring is at the location pointed to by ERDSR and ETDSR for the receive and transmit rings, respectively. Buffer descriptor rings must start on a 32-bit boundary; however, it is recommended they be 128-bit aligned.

52.6.3.1.1 Driver/DMA operation with transmit BDs

Typically, a transmit frame is divided between multiple buffers. An example is to have an application payload in one buffer, TCP header in a second buffer, IP header in a third buffer, and Ethernet/IEEE 802.3 header in a fourth buffer. The Ethernet MAC does not prepend the Ethernet header (destination address, source address, length/type field(s)), so the driver must provide this in one of the transmit buffers. The Ethernet MAC can append the Ethernet CRC to the frame. TxBD[TC], which must be set by the driver, determines whether the MAC or driver appends the CRC.

The driver (TxBD software producer) should set up Tx BDs so a complete transmit frame is given to the hardware at once. If a transmit frame consists of three buffers, the BDs should be initialized with pointer, length, and control (W, L, TC, ABC) and then the TxBD[R] bit should be set in reverse order (third, second, then first BD) to ensure that the complete frame is ready in memory before the DMA begins. If the TxBDs are set up in order, the DMA controller could DMA the first BD before the second was made available, potentially causing a transmit FIFO underrun.

In the FEC, the driver notifies the DMA that new transmit frame(s) are available by writing to TDAR. When this register is written to (data value is not significant), the FEC RISC tells the DMA to read the next transmit BD in the ring. After started, the RISC + DMA continues to read and interpret transmit BDs in order and DMA the associated buffers until a transmit BD is encountered with the R bit cleared. At this point, the FEC polls this BD one more time. If the R bit is cleared the second time, RISC stops the transmit descriptor read process until software sets up another transmit frame and writes to TDAR.

When the DMA of each transmit buffer is complete, the DMA writes back to the BD to clear the R bit, indicating that the hardware consumer is finished with the buffer.

52.6.3.1.2 Driver/DMA operation with receive BDs

Unlike transmit, the length of the receive frame is unknown by the driver ahead of time. Therefore, the driver must set a variable to define the length of all receive buffers. In the FEC, this variable is written to the EMRBR register.

The driver (RxBD software producer) should set up some number of empty buffers for the Ethernet by initializing the address field and the E and W bits of the associated receive BDs. The hardware (receive DMA) consumes these buffers by filling them with data as frames are received and clearing the E bit and writing to the L bit (1 indicates last buffer in frame), the frame status bits (if L is set), and the length field.

If a receive frame spans multiple receive buffers, the L bit is only set for the last buffer in the frame. For non-last buffers, the length field in the receive BD is written by the DMA (at the same time the E bit is cleared) with the default receive buffer length value. For end-of-frame buffers, the receive BD is written with L set and information written to the status bits (M, BC, MC, LG, NO, CR, OV, TR). Some of the status bits are error indicators which, if set, indicate the receive frame should be discarded and not given to higher layers. The frame status/length information is written into the receive FIFO following the end of the frame (as a single 32-bit word) by the receive logic. The length field for the end of frame buffer is written with the length of the entire frame, not only the length of the last buffer.

For simplicity, the driver may assign a large enough default receive buffer length to contain an entire frame, keeping in mind that a malfunction on the network or out-of-spec implementation could result in giant frames. Frames of 2K (2048) bytes or larger are truncated by the FEC at 2047 bytes so software never sees a receive frame larger than 2047 bytes.

Similar to transmit, the FEC polls the receive descriptor ring after the driver sets up receive BDs and writes to the RDAR register. As frames are received, the FEC fills receive buffers and updates the associated BDs, then reads the next BD in the receive

descriptor ring. If the FEC reads a receive BD and finds the E bit cleared, it polls this BD once more. If RxBD[E] is clear a second time, FEC stops reading receive BDs until the driver writes to RDAR.

52.6.3.2 Ethernet Receive Buffer Descriptor (RxBD)

In the RxBD, the user initializes the E and W bits in the first longword and the pointer in the second longword. When the buffer has been DMA'd, the Ethernet controller modifies the E, L, M, BC, MC, LG, NO, CR, OV, and TR bits and writes the length of the used portion of the buffer in the first longword. The M, BC, MC, LG, NO, CR, OV, and TR bits in the first longword of the buffer descriptor are only modified by the Ethernet controller when the L bit is set.

Table 52-89. Receive Buffer Descriptor (RxBD)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset+0	E	RO1	W	RO2	L	—	—	M	BC	MC	LG	NO	—	CR	OV	TR
Offset+2	Data Length															
Offset+4	Rx Data Buffer Pointer - A[31:16]															
Offset+6	Rx Data Buffer Pointer - A[15:0]															

Table 52-90. Receive Buffer Descriptor field descriptions

Word	Field	Description
Offset + 0	E	Empty. Written by the FEC (=0) and user (=1). 0: The data buffer associated with this BD is filled with received data, or data reception has aborted due to an error condition. The status and length fields have been updated as required. 1: The data buffer associated with this BD is empty, or reception is currently in progress.
Offset + 0	RO1	Receive software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
Offset + 0	W	Wrap. Written by user. 0: The next buffer descriptor is found in the consecutive location 1: The next buffer descriptor is found at the location defined in ERDSR.
Offset + 0	RO2	Receive software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
Offset + 0	L	Last in frame. Written by the FEC. 0: The buffer is not the last in a frame. 1: The buffer is the last in a frame.

Table continues on the next page...

Table 52-90. Receive Buffer Descriptor field descriptions (continued)

Word	Field	Description
Offset + 0	M	Miss. Written by the FEC. This bit is set by the FEC for frames accepted in promiscuous mode, but flagged as a miss by the internal address recognition. Therefore, while in promiscuous mode, you can use the M-bit to quickly determine whether the frame was destined to this station. This bit is valid only if the L-bit is set and the PROM bit is set. 0: The frame was received because of an address recognition hit. 1: The frame was received because of promiscuous mode.
Offset + 0	BC	Set if the DA is broadcast (FFFF_FFFF_FFFF).
Offset + 0	MC	Set if the DA is multicast and not BC.
Offset + 0	LG	Rx frame length violation. Written by the FEC. A frame length greater than RCR[MAX_FL] was recognized. This bit is valid only if the L-bit is set. The receive data is not altered in any way unless the length exceeds 2047 bytes.
Offset + 0	NO	Receive non-octet aligned frame. Written by the FEC. A frame that contained a number of bits not divisible by 8 was received, and the CRC check that occurred at the preceding byte boundary generated an error. This bit is valid only if the L-bit is set. If this bit is set, the CR bit is not set.
Offset + 0	CR	Receive CRC error. Written by the FEC. This frame contains a CRC error and is an integral number of octets in length. This bit is valid only if the L-bit is set.
Offset + 0	OV	Overrun. Written by the FEC. A receive FIFO overrun occurred during frame reception. If this bit is set, the other status bits, M, LG, NO, CR, and CL lose their normal meaning and are zero. This bit is valid only if the L-bit is set.
Offset + 0	TR	Set if the receive frame is truncated (frame length > 2047 bytes). If the TR bit is set, the frame must be discarded and the other error bits must be ignored as they may be incorrect.
Offset + 2	Data Length	Data length. Written by the FEC. Data length is the number of octets written by the FEC into this BD's data buffer if L equals 0 (the value is equal to EMRBR), or the length of the frame including CRC if L is set. It is written by the FEC once as the BD is closed.
0ffset + 4	A[31:16]	RX data buffer pointer, bits [31:16][1]
Offset + 6	A[15:0]	RX data buffer pointer, bits [15:0]

1. The receive buffer pointer, containing the address of the associated data buffer, must always be evenly divisible by 16. The buffer must reside in memory external to the FEC. The Ethernet controller never modifies this value.

Note

When the software driver sets an E bit in one or more receive descriptors, the driver should follow with a write to RDAR.

52.6.3.3 Ethernet Transmit Buffer Descriptor (TxBD)

Data is presented to the FEC for transmission by arranging it in buffers referenced by the channel's TxBDs. The Ethernet controller confirms transmission by clearing the ready bit (TxBD[R]) when DMA of the buffer is complete. In the TxBD, the user initializes the R, W, L, and TC bits and the length (in bytes) in the first longword and the buffer pointer in the second longword.

The FEC clears the R bit when the buffer is transferred. Status bits for the buffer/frame are not included in the transmit buffer descriptors. Transmit frame status is indicated via individual interrupt bits (error conditions) and in statistic counters in the MIB block.

Table 52-91. Transmit Buffer Descriptor (TxBD)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset+0	R	TO1	W	TO2	L	TC	ABC	—	—	—	—	—	—	—	—	—
Offset+2	Data Length															
Offset+4	Tx Data Buffer Pointer - A[31:16]															
Offset+6	Tx Data Buffer Pointer - A[15:0]															

Table 52-92. Transmit Buffer Descriptor field descriptions

Word	Field	Description
Offset + 0	R	Ready. Written by the FEC and you. 0: The data buffer associated with this BD is not ready for transmission. You are free to manipulate this BD or its associated data buffer. The FEC clears this bit after the buffer has been transmitted or after an error condition is encountered. 1: The data buffer, prepared for transmission by you, has not been transmitted or currently transmits. You may write no fields of this BD after this bit is set.
Offset + 0	TO1	Transmit software ownership. This field is reserved for software use. This read/write bit is not modified by hardware nor does its value affect hardware.
Offset + 0	W	Wrap. Written by user. 0: The next buffer descriptor is found in the consecutive location 1: The next buffer descriptor is found at the location defined in ETDSR.
Offset + 0	TO2	Transmit software ownership. This field is reserved for use by software. This read/write bit is not modified by hardware nor does its value affect hardware.
Offset + 0	L	Last in frame. Written by user. 0: The buffer is not the last in the transmit frame 1: The buffer is the last in the transmit frame
Offset + 0	TC	Transmit CRC. Written by user (only valid if L is set). 0: End transmission immediately after the last data byte 1: Transmit the CRC sequence after the last data byte
Offset + 0	ABC	Append bad CRC. Written by user (only valid if L is set). 0: No effect 1: Transmit the CRC sequence inverted after the last data byte (regardless of TC value)
Offset + 2	Data Length	Data length, written by user. Data length is the number of octets the FEC should transmit from this BD's data buffer. It is never modified by the FEC.
Offset + 4	A[31:16]	Tx data buffer pointer, bits [31:16] [1]
Offset + 6	A[15:0]	Tx data buffer pointer, bits [15:0]

1. The transmit buffer pointer, containing the address of the associated data buffer, must always be evenly divisible by 4. The buffer must reside in memory external to the FEC. This value is never modified by the Ethernet controller.

Note

After the software driver has set up the buffers for a frame, it should set up the corresponding BDs. The last step in setting up the BDs for a transmit frame is setting the R bit in the first BD for the frame. The driver must follow that with a write to TDAR that triggers the FEC to poll the next BD in the ring.

52.6.4 Initialization sequence

This section describes which registers are reset due to hardware reset, which registers are reset by the FEC RISC, and what locations you must initialize prior to enabling the FEC.

52.6.4.1 Hardware Controlled Initialization

In the FEC, hardware resets registers and control logic that generate interrupts. A hardware reset negates output signals and resets general configuration bits.

Other registers reset when the ECR[ETHER_EN] bit is cleared (which is accomplished by a hard reset or software to halt operation). By clearing ECR[ETHER_EN], configuration control registers such as the TCR and RCR are not reset, but the entire data path is reset.

Table 52-93. ECR[ETHER_EN] De-Assertion Effect on FEC

Register/Machine	Reset Value
XMIT block	Transmission is aborted (bad CRC appended)
RECV block	Receive activity is aborted
DMA block	All DMA activity is terminated
RDAR	Cleared
TDAR	Cleared
Descriptor Controller block	Halt operation

52.6.5 User initialization (Prior to Setting ECR[ETHER_EN])

You need to initialize portions the FEC prior to setting the ECR[ETHER_EN] bit. The exact values depend on the particular application. The sequence is not important.

The following registers require initialization:

- EIMR
- EIR (clear by writing FFFF_FFFFh)
- TFWR (optional)
- IALR / IAUR
- GAUR / GALR
- PALR / PAUR (only needed for full duplex flow control)
- OPD (only needed for full duplex flow control)
- RCR
- TCR
- MSCR (optional)
- MIB_RAM (clear)
- FRSR (optional)
- EMRBR
- ERDSR
- ETDSR
- Transmit Descriptor ring (empty)
- Receive Descriptor ring (empty)

52.6.6 Microcontroller initialization

In the FEC, the descriptor control RISC initializes some registers after ECR[ETHER_EN] is asserted. After the microcontroller initialization sequence is complete, hardware is ready for operation.

The sequence is as follows:

1. Initialize the backoff random number seed.
2. Activate receiver.
3. Activate transmitter.

4. Clear transmit FIFO.
5. Clear receive FIFO.
6. Initialize the transmit ring pointer.
7. Initialize the receive ring pointer.
8. Initialize FIFO count registers.

52.6.7 User initialization (after setting ECR[ETHER_EN])

After setting ECR[ETHER_EN], you can set up the buffer/frame descriptors and write to TDAR and RDAR. See Buffer descriptors, for more details.

52.6.8 Network interface options

The FEC supports an MII and reduced MII interface for 10/100 Mbps Ethernet, as well as a 7-wire serial interface for 10 Mbps Ethernet. The RCR[MII_MODE] and RCR [RMII_MODE] bits select the interface mode. In MII mode (RCR[MII_MODE]=1 and RCR [RMII_MODE]=0), there are 18 signals defined by the IEEE 802.3 standard and supported by the EMAC. The following table shows these signals.

Table 52-94. MII mode

Signal description	EMAC pin
Transmit Clock	FEC_TXCLK
Transmit Enable	FEC_TXEN
Transmit Data	FEC_TXD[3:0]
Transmit Error	FEC_TXER
Collision	FEC_COL
Carrier Sense	FEC_CRS
Receive Clock	FEC_RXCLK
Receive Data Valid	FEC_RXDV
Receive Data	FEC_RXD[3:0]
Receive Error	FEC_RXER
Management Data Clock	FEC_MDC
Management Data Input/Output	FEC_MDIO

In RMII mode (RCR[MII_MODE]=1 and RCR[RMII_MODE]=1), the EMAC supports 8 external signals. These signals are shown in the table below.

Table 52-95. RMII mode configuration

Signal description	EMAC pin
Reference Clock	FEC_TXCLK
Transmit Enable	FEC_TXEN
Transmit Data	FEC_TXD[1:0]
Receive Data Valid	FEC_RXDV
Receive Data	FEC_RXD[1:0]
Receive Error	FEC_RXER

The 7-wire serial mode interface (RCR[MII_MODE]=1 and RCR[RMII_MODE]=0 or 1) is generally referred to as AMD mode. The following table shows the 7-wire mode connections to the external transceiver.

Table 52-96. 7-Wire mode configuration

Signal description	EMAC pin
Transmit Clock	FEC_TXCLK
Transmit Enable	FEC_TXEN
Transmit Data	FEC_TXD[0]
Collision	FEC_COL
Receive Clock	FEC_RXCLK
Receive Data Valid	FEC_RXDV
Receive Data	FEC_RXD[0]

52.6.9 FEC frame transmission

The Ethernet transmitter is designed to work with almost no intervention from software. After ECR[ETHER_EN] is set and data appears in the transmit FIFO, the Ethernet MAC can transmit onto the network. The Ethernet controller transmits bytes least significant bit (lsb) first.

When the transmit FIFO fills to the watermark (defined by TFWR), MAC transmit logic asserts FEC_TXEN and starts transmitting the (PA) sequence, the start frame delimiter (SFD), and then the frame information from the FIFO. However, the controller defers the transmission if the network is busy (FEC_CRS is asserted). Before transmitting, the controller waits for carrier sense to become inactive, then determines if carrier sense stays inactive for 60 bit times. If so, transmission begins after waiting an additional 36 bit times (96 bit times after carrier sense originally became inactive). See "Transmission errors" for more details.

If a collision occurs during transmission of the frame (half duplex mode), the Ethernet controller follows the specified backoff procedures and attempts to retransmit the frame until the retry limit is reached. The transmit FIFO stores at least the first 64 bytes of the transmit frame, so they do not have to be retrieved from system memory in case of a collision. This improves bus utilization and latency in case immediate retransmission is necessary.

When all the frame data is transmitted, FCS (frame check sequence) or 32-bit cyclic redundancy check (CRC) bytes are appended if the TC bit is set in the transmit frame control word. If the ABC bit is set in the transmit frame control word, a bad CRC is appended to the frame data regardless of the TC bit value. Following the transmission of the CRC, the Ethernet controller writes the frame status information to the MIB block. Transmit logic automatically pads short frames (if the TC bit in the transmit buffer descriptor for the end of frame buffer is set).

Settings in the EIMR determine interrupts generated to the buffer (TXB) and frame (TXF).

The transmit error interrupts are HBERR, BABT, LC, RL, and UN. If the transmit frame length exceeds MAX_FL bytes, BABT interrupt is asserted. However, the entire frame is transmitted (no truncation).

To pause transmission, set TCR[GTS] (graceful transmit stop). The FEC transmitter stops immediately if transmission is not in progress; otherwise, it continues transmission until the current frame finishes or terminates with a collision. After the transmitter has stopped, the GRA (graceful stop complete) interrupt is asserted. If TCR[GTS] is cleared, the FEC resumes transmission with the next frame.

52.6.9.1 Duplicate frame transmission

The FEC fetches transmit buffer descriptors (TxBDs) and the corresponding transmit data continuously until the transmit FIFO is full. It does not determine whether the TxBD to be fetched is already being processed internally (as a result of a wrap). As the FEC nears the end of the transmission of one frame, it begins to DMA the data for the next frame. To remain one BD ahead of the DMA, it also fetches the TxBD for the next frame. It is possible that the FEC fetches from memory a BD that has already been processed but not yet written back (it is read a second time with the R bit remains set). In this case, the data is fetched and transmitted again.

Using at least three TxBDs fixes this problem for large frames, but not for small frames. To ensure correct operation for large or small frames, one of the following must be true:

- The FEC software driver ensures that there is always at least one TxBD with the ready bit cleared.
- Every frame uses more than one TxBD and every TxBD but the last is written back immediately after the data is fetched.
- The FEC software driver ensures a minimum frame size, *n.* The minimum number of TxBDs is then (Tx FIFO Size ÷ (*n*+4)) rounded up to the nearest integer (though the result cannot be less than three). The default Tx FIFO size is 192 bytes; this size is programmable.

52.6.10 FEC frame reception

The FEC receiver works with almost no intervention from the host and can perform address recognition, CRC checking, short frame checking, and maximum frame length checking. The Ethernet controller receives serial data lsb first.

When the driver enables the FEC receiver by setting ECR[ETHER_EN], it immediately starts processing receive frames. When FEC_RXDV is asserted, the receiver first checks for a valid PA/SFD header. If the PA/SFD is valid, it is stripped and the receiver processes the frame. If a valid PA/SFD is not found, the frame is ignored.

In serial mode, the first 16 bit times of RX_D0 following assertion of FEC_RXDV are ignored. Following the first 16 bit times, the data sequence is checked for alternating 1/0s. If a 11 or 00 data sequence is detected during bit times 17 to 21, the remainder of the frame is ignored. After bit time 21, the data sequence is monitored for a valid SFD (11). If a 00 is detected, the frame is rejected. When a 11 is detected, the PA/SFD sequence is complete.

In MII mode, the receiver checks for at least one byte matching the SFD. Zero or more PA bytes may occur, but if a 00 bit sequence is detected prior to the SFD byte, the frame is ignored.

After the first 6 bytes of the frame are received, the FEC performs address recognition on the frame.

After a collision window (64 bytes) of data is received and if address recognition has not rejected the frame, the receive FIFO signals the frame is accepted and may be passed on to the DMA. If the frame is a runt (due to collision) or is rejected by address recognition, the receive FIFO is notified to reject the frame. Therefore, no collision fragments are presented to you except late collisions, which indicate serious LAN problems.

During reception, the Ethernet controller checks for various error conditions and after the entire frame is written into the FIFO, a 32-bit frame status word is written into the FIFO. This status word contains the M, BC, MC, LG, NO, CR, OV, and TR status bits, and the frame length. See "Reception errors" for more details.

Receive buffer (RXB) and frame interrupts (RFINT) may be generated if enabled by the EIMR register. A receive error interrupt is a babbling receiver error (BABR). Receive frames are not truncated if they exceed the max frame length (MAX_FL); however, the BABR interrupt occurs and the LG bit in the receive buffer descriptor (RxBD) is set. See Ethernet Receive Buffer Descriptor (RxBD), for more details.

When the receive frame is complete, the FEC sets the L-bit in the RxBD, writes the other frame status bits into the RxBD, and clears the E-bit. The Ethernet controller next generates a maskable interrupt (RFINT bit in EIR, maskable by RFIEN bit in EIMR), indicating that a frame is received and is in memory. The Ethernet controller then waits for a new frame.

52.6.11 Ethernet address recognition

The FEC filters the received frames based on destination address (DA) type — individual (unicast), group (multicast), or broadcast (all-ones group address). The difference between an individual address and a group address is determined by the I/G bit in the destination address field. A flowchart for address recognition on received frames appears in the figures below.

Address recognition is accomplished through the use of the receive block and microcode running on the microcontroller. The flowchart shown in Figure 52-84 illustrates the address recognition decisions made by the receive block, while Figure 52-85 illustrates the decisions made by the microcontroller.

If the DA is a broadcast address and broadcast reject (RCR[BC_REJ]) is cleared, then the frame is accepted unconditionally, as shown in Figure 52-84. Otherwise, if the DA is not a broadcast address, then the microcontroller runs the address recognition subroutine, as shown in Figure 52-85.

If the DA is a group (multicast) address and flow control is disabled, then the microcontroller performs a group hash table lookup using the 64-entry hash table programmed in GAUR and GALR. If a hash match occurs, the receiver accepts the frame.

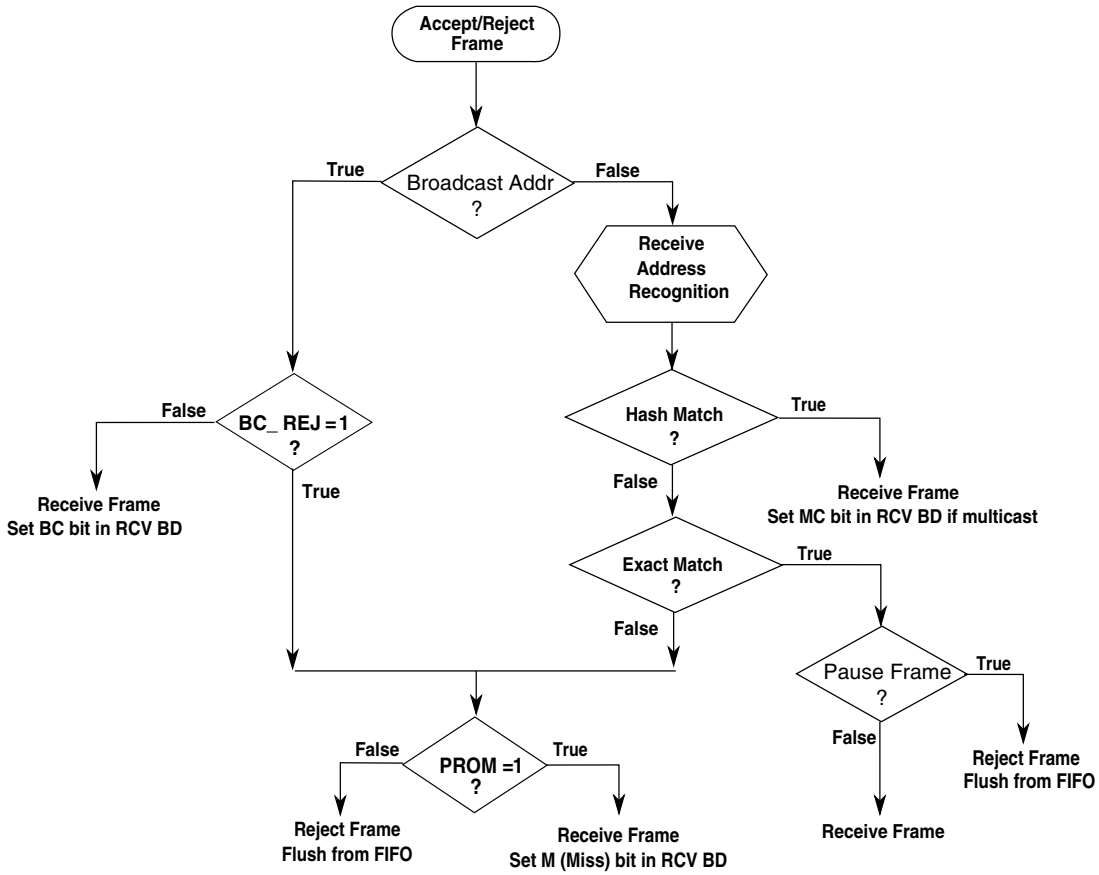
If flow control is enabled, the microcontroller does an exact address match check between the DA and the designated PAUSE DA (01:80:C2:00:00:01). If the receive block determines the received frame is a valid PAUSE frame, the frame is rejected. The receiver detects a PAUSE frame with the DA field set to the designated PAUSE DA or the unicast physical address.

If the DA is the individual (unicast) address, the microcontroller performs an individual exact match comparison between the DA and 48-bit physical address that you program in the PALR and PAUR registers. If an exact match occurs, the frame is accepted; otherwise, the microcontroller does an individual hash table lookup using the 64-entry hash table programmed in registers, IAUR and IALR. In the case of an individual hash match, the frame is accepted. Again, the receiver accepts or rejects the frame based on PAUSE frame detection, shown in Figure 52-84.

If neither a hash match (group or individual) nor an exact match (group or individual) occur, and if promiscuous mode is enabled (RCR[PROM] set), the frame is accepted and the MISS bit in the receive buffer descriptor is set; otherwise, the frame is rejected.

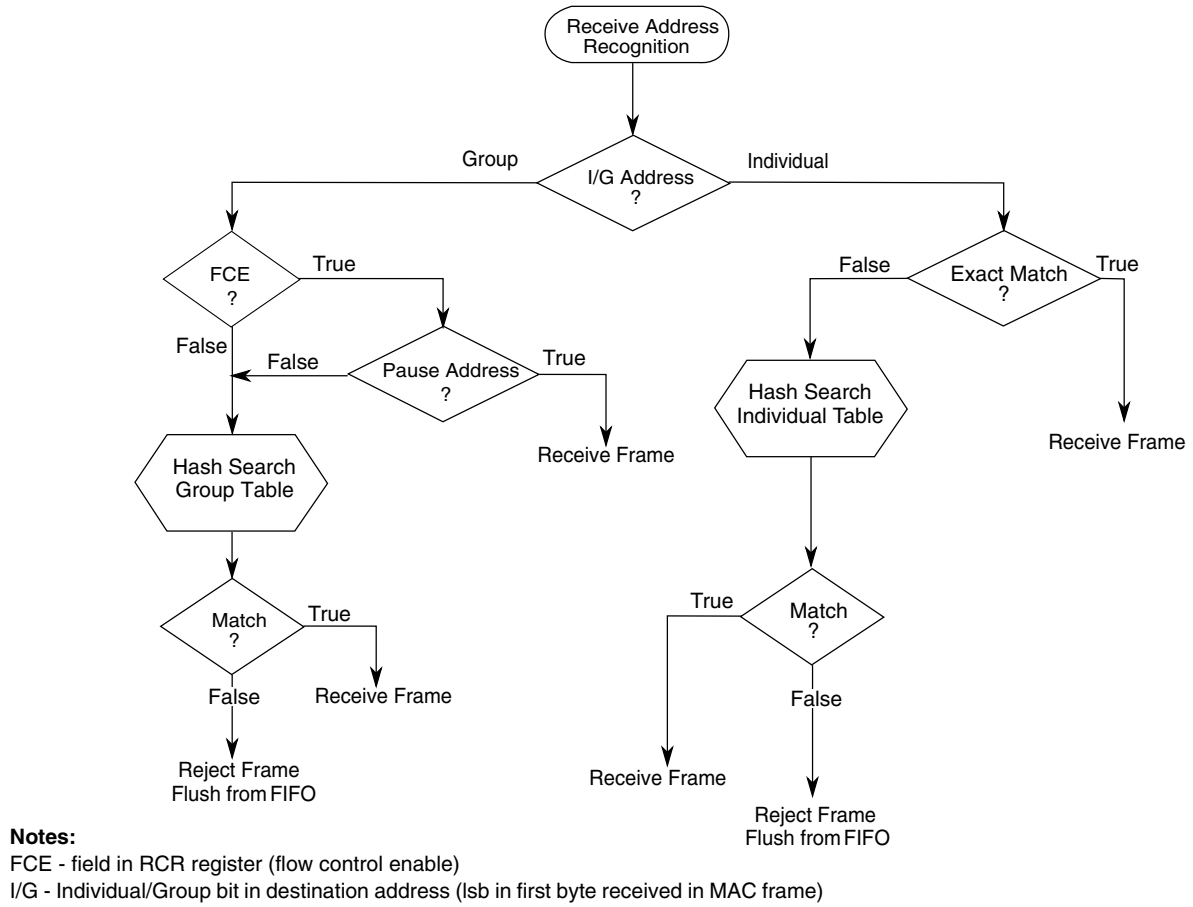
Similarly, if the DA is a broadcast address, broadcast reject (RCR[BC_REJ]) is asserted, and promiscuous mode is enabled, the frame is accepted and the MISS bit in the receive buffer descriptor is set; otherwise, the frame is rejected.

In general, when a frame is rejected, it is flushed from the FIFO.



Notes:
BC_REJ - field in RCR register (BroadCast REJect)
PROM - field in RCR register (PROMiscous mode)
Pause Frame - valid PAUSE frame received

Figure 52-84. Ethernet address recognition—receive block decisions



Notes:
FCE - field in RCR register (flow control enable)
I/G - Individual/Group bit in destination address (lsb in first byte received in MAC frame)

Figure 52-85. Ethernet address recognition—microcode decisions

52.6.12 Hash algorithm

The hash table algorithm used in the group and individual hash filtering operates as follows. The 48-bit destination address is mapped into one of 64 bits, represented by 64 bits stored in GAUR, GALR (group address hash match), or IAUR, IALR (individual address hash match). This mapping is performed by passing the 48-bit address through the on-chip 32-bit CRC generator and selecting the six most significant bits of the CRC-encoded result to generate a number between 0 and 63. The msb of the CRC result selects GAUR (msb = 1) or GALR (msb = 0). The five least significant bits of the hash result select the bit within the selected register. If the CRC generator selects a bit set in the hash table, the frame is accepted; otherwise, it is rejected.

For example, if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (87.5%) of the group address frames from reaching memory. Those that do reach memory must be further filtered by the processor to determine if they truly contain one of the eight desired addresses.

The effectiveness of the hash table declines as the number of addresses increases.

The user must initialize the hash table registers. Use this CRC32 polynomial to compute the hash:

$$X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^{8}+X^{7}+X^{5}+X^{4}+X^{2}+X+1$$

Figure 52-86. CRC32 polynomial for hash computation

The following table contains example destination addresses and corresponding hash values.

Table 52-97. Destination address to 6-bit hash

48-BIT DA	6-bit hash (in hex)	Hash decimal value
65FF_FFFF_FFFF	0h	0
55FF_FFFF_FFFF	1h	1
15FF_FFFF_FFFF	2h	2
35FF_FFFF_FFFF	3h	3
B5FF_FFFF_FFFF	4h	4
95FF_FFFF_FFFF	5h	5
D5FF_FFFF_FFFF	6h	6
F5FF_FFFF_FFFF	7h	7
DBFF_FFFF_FFFF	8h	8
FBFF_FFFF_FFFF	9h	9
BBFF_FFFF_FFFF	Ah	10
8BFF_FFFF_FFFF	Bh	11
0BFF_FFFF_FFFF	Ch	12
3BFF_FFFF_FFFF	Dh	13
7BFF_FFFF_FFFF	Eh	14
5BFF_FFFF_FFFF	Fh	15
27FF_FFFF_FFFF	10h	16
07FF_FFFF_FFFF	11h	17
57FF_FFFF_FFFF	12h	18
77FF_FFFF_FFFF	13h	19
F7FF_FFFF_FFFF	14h	20
C7FF_FFFF_FFFF	15h	21
97FF_FFFF_FFFF	16h	22
A7FF_FFFF_FFFF	17h	23
99FF_FFFF_FFFF	18h	24
B9FF_FFFF_FFFF	19h	25
F9FF_FFFF_FFFF	1Ah	26
C9FF_FFFF_FFFF	1Bh	27

Table continues on the next page...

Table 52-97. Destination address to 6-bit hash (continued)

48-BIT DA	6-bit hash (in hex)	Hash decimal value
59FF_FFFF_FFFF	1Ch	28
79FF_FFFF_FFFF	1Dh	29
29FF_FFFF_FFFF	1Eh	30
19FF_FFFF_FFFF	1Fh	31
D1FF_FFFF_FFFF	20h	32
F1FF_FFFF_FFFF	21h	33
B1FF_FFFF_FFFF	22h	34
91FF_FFFF_FFFF	23h	35
11FF_FFFF_FFFF	24h	36
31FF_FFFF_FFFF	25h	37
71FF_FFFF_FFFF	26h	38
51FF_FFFF_FFFF	27h	39
7FFF_FFFF_FFFF	28h	40
4FFF_FFFF_FFFF	29h	41
1FFF_FFFF_FFFF	2Ah	42
3FFF_FFFF_FFFF	2Bh	43
BFFF_FFFF_FFFF	2Ch	44
9FFF_FFFF_FFFF	2Dh	45
DFFF_FFFF_FFFF	2Eh	46
EFFF_FFFF_FFFF	2Fh	47
93FF_FFFF_FFFF	30h	48
B3FF_FFFF_FFFF	31h	49
F3FF_FFFF_FFFF	32h	50
D3FF_FFFF_FFFF	33h	51
53FF_FFFF_FFFF	34h	52
73FF_FFFF_FFFF	35h	53
23FF_FFFF_FFFF	36h	54
13FF_FFFF_FFFF	37h	55
3DFF_FFFF_FFFF	38h	56
0DFF_FFFF_FFFF	39h	57
5DFF_FFFF_FFFF	3Ah	58
7DFF_FFFF_FFFF	3Bh	59
FDFF_FFFF_FFFF	3Ch	60
DDFF_FFFF_FFFF	3Dh	61
9DFF_FFFF_FFFF	3Eh	62
BDFF_FFFF_FFFF	3Fh	63

52.6.13 Full duplex flow control

Full-duplex flow control allows you to transmit pause frames and to detect received pause frames. Upon detection of a pause frame, MAC data frame transmission stops for a given pause duration.

To enable PAUSE frame detection, the FEC must operate in full-duplex mode (TCR[FDEN] set) with flow control (RCR[FCE] set). The FEC detects a pause frame when the fields of the incoming frame match the pause frame specifications, as shown in the following table. In addition, the receive status associated with the frame should indicate that the frame is valid.

Table 52-98. PAUSE frame field specification

48-bit destination address	0180_C200_0001h or Physical Address
48-bit source address	Any
16-bit type	8808h
16-bit opcode	0001h
16-bit PAUSE duration	0000h — FFFFh

The receiver and microcontroller modules perform PAUSE frame detection. The microcontroller runs an address recognition subroutine to detect the specified pause frame destination address, while the receiver detects the type and opcode pause frame fields. On detection of a pause frame, TCR[GTS] is set by the FEC internally. When transmission has paused, the EIR[GRA] interrupt is asserted and the pause timer begins to increment. The pause timer uses the transmit backoff timer hardware for tracking the appropriate collision backoff time in half-duplex mode. The pause timer increments once every slot time, until OPD[PAUSE_DUR] slot times have expired. On OPD[PAUSE_DUR] expiration, TCR[GTS] is cleared allowing MAC data frame transmission to resume. The receive flow control pause status bit (TCR[RFC_PAUSE]) is set while the transmitter pauses due to reception of a pause frame.

To transmit a pause frame, the FEC must operate in full-duplex mode and you must set flow control pause (TCR[TFC_PAUSE]). After TCR[TFC_PAUSE] is set, the transmitter sets TCR[GTS] internally. When the transmission of data frames stops, the EIR[GRA] (graceful stop complete) interrupt asserts and the pause frame is transmitted. TCR[TFC_PAUSE,GTS] are then cleared internally.

You must specify the desired pause duration in the OPD register.

When the transmitter pauses due to receiver/microcontroller pause frame detection, TCR[TFC_PAUSE] may remain set and cause the transmission of a single pause frame. In this case, the EIR[GRA] interrupt is not asserted.

52.6.14 Inter-Packet Gap (IPG) time

The minimum inter-packet gap time for back-to-back transmission is 96 bit times. After completing a transmission or after the backoff algorithm completes, the transmitter waits for carrier sense to be negated before starting its 96 bit time IPG counter. Frame transmission may begin 96 bit times after carrier sense is negated if it stays negated for at least 60 bit times. If carrier sense asserts during the last 36 bit times, it is ignored and a collision occurs.

The receiver accepts back-to-back frames with a minimum spacing of at least 28 bit times. If an inter-packet gap between receive frames is less than 28 bit times, the receiver may discard the following frame.

52.6.15 Collision managing

If a collision occurs during frame transmission, the Ethernet controller continues the transmission for at least 32 bit times, transmitting a JAM pattern consisting of 32 ones. If the collision occurs during the sequence, a JAM pattern is sent after the end of the preamble sequence.

If a collision occurs within 512 bit times (one slot time), the retry process is initiated. The transmitter waits a random number of slot times. If a collision occurs after 512 bit times, then no retransmission is performed and the end of frame buffer is closed with a Late Collision (LC) error indication.

52.6.16 MII internal and external loopback

Internal and external loopback are supported by the Ethernet controller. In loopback mode, both of the FIFOs are used and the FEC actually operates in a full-duplex fashion. Internal and external loopback are configured using combinations of the RCR[LOOP, DRT] and TCR[FDEN] bits.

Set FDEN for internal and external loopback.

For internal loopback, set RCR[LOOP] and clear RCR[DRT]. FEC_TXEN and FEC_TXER do not assert during internal loopback. During internal loopback, the transmit/receive data rate is higher than in normal operation because the transmit and receive blocks use the internal bus clock instead of the clocks from the external transceiver. This causes an increase in the required system bus bandwidth for transmit

and receive data being DMA'd to/from external memory. It may be necessary to pace the frames on the transmit side and/or limit the size of the frames to prevent transmit FIFO underruns and receive FIFO overflows.

For external loopback, clear RCR[LOOP] and RCR[DRT], and configure the external transceiver for loopback.

52.6.17 RMII loopback

The FEC also supports RMII loopback. In this mode, transmit data is sent to the RMII receive logic and out the FEC_TXD[1:0] pins. To enable RMII loopback mode, set RCR[RMII_MODE, RMII_LOOP] and TCR[FDEN] and clear RCR[RMII_ECHO, LOOP].

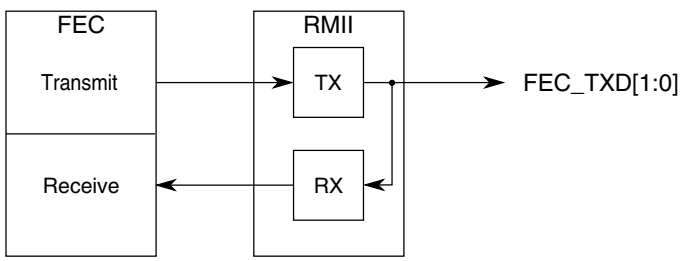


Figure 52-87. RMII loopback mode

52.6.18 RMII echo

The Ethernet controller also supports RMII echo mode, which transmits data on FEC_TXD[1:0] as it is received on FEC_RXD[1:0]. To enable RMII echo mode, set RCR[RMII_MODE, RMII_ECHO] bits and clear RCR[RMII_LOOP, LOOP].

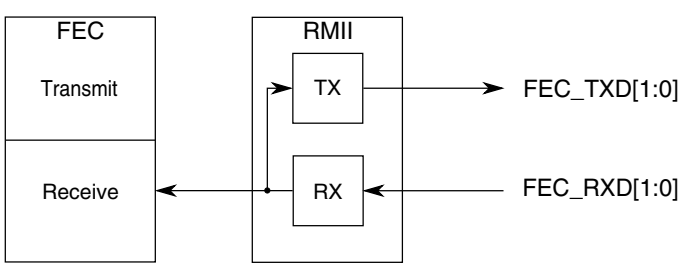


Figure 52-88. RMII echo mode

52.6.19 Ethernet error-managing procedure

The Ethernet controller reports frame reception and transmission error conditions using the MIB block counters, the FEC RxBDs, and the EIR register.

52.6.19.1 Transmission errors

52.6.19.1.1 Transmitter underrun

If this error occurs, the FEC sends 32 bits that ensure a CRC error and stops transmitting. All remaining buffers for that frame are then flushed and closed, and EIR[UN] is set. The FEC then continues to the next transmit buffer descriptor and begin transmitting the next frame. The UN interrupt is asserted if enabled in the EIMR register.

52.6.19.1.2 Retransmission attempts limit expired

When this error occurs, the FEC terminates transmission. All remaining buffers for that frame are flushed and closed, and EIR[RL] is set. The FEC then continues to the next transmit buffer descriptor and begins transmitting the next frame. The RL interrupt is asserted if enabled in the EIMR register.

52.6.19.1.3 Late collision

When a collision occurs after the slot time (512 bits starting at the Preamble), the FEC terminates transmission. All remaining buffers for that frame are flushed and closed, and EIR[LC] is set. The FEC then continues to the next transmit buffer descriptor and begins transmitting the next frame. The LC interrupt is asserted if enabled in the EIMR register.

52.6.19.1.4 Heartbeat

Some transceivers have a self-test feature called heartbeat or signal quality error. To signify a good self-test, the transceiver indicates a collision to the FEC within four microseconds after completion of a frame transmitted by the Ethernet controller. This indication of a collision does not imply a real collision error on the network, but is rather an indication that the transceiver continues to function properly. This is the heartbeat condition.

If TCR[HBC] is set and the heartbeat condition is not detected by the FEC after a frame transmission, a heartbeat error occurs. When this error occurs, the FEC closes the buffer, sets EIR[HB], and generates the HBERR interrupt if it is enabled.

52.6.19.2 Reception errors

52.6.19.2.1 Overrun error

If the receive block has data to put into the receive FIFO and the receive FIFO is full, FEC sets RxBD[OV]. All subsequent data in the frame is discarded and subsequent frames may also be discarded until the receive FIFO is serviced by the DMA and space is made available. At this point the receive frame/status word is written into the FIFO with the OV bit set. The driver must discard this frame.

52.6.19.2.2 Non-octet error (dribbling bits)

The Ethernet controller manages up to seven dribbling bits when the receive frame terminates past an non-octet aligned boundary. Dribbling bits are not used in the CRC calculation. If there is a CRC error, the frame non-octet aligned (NO) error is reported in the RxBD. If there is no CRC error, no error is reported.

52.6.19.2.3 CRC error

When a CRC error occurs with no dribble bits, FEC closes the buffer and sets RxBD[CR]. CRC checking cannot be disabled, but the CRC error can be ignored if checking is not required.

52.6.19.2.4 Frame length violation

When the receive frame length exceeds MAX_FL bytes the BABR interrupt is generated, and RxBD[LG] is set. The frame is not truncated unless the frame length exceeds 2047 bytes.

52.6.19.2.5 Truncation

When the receive frame length exceeds 2047 bytes, frame is truncated and RxBD[TR] is set.

Chapter 53
SENT Receiver (SRX)

53.1 SENT chip-specific modules information

The Single Edge Nibble Transmission (SENT) module is designed as a receiver for transmitting sensor nodes using the SENT protocol (as documented by the SAE J2716 information report). SENT is a unidirectional communications scheme from a sensor (transmitting) device to a controller (receiving) device which does not include a coordination signal from the controller/receiving device.

Table 53-1. SENT modules/channels implemented on this device

SENT_0_CH0
SENT_0_CH1
SENT_0_CH2
SENT_1_CH0
SENT_1_CH1
SENT_1_CH2

Table 53-2. Base addresses of SENT modules implemented on this device

Instantiated module	Base address
SENT_0	FFE5C000
SENT_1	FBE5C000

53.1.1 SRX_GBL_CTRL[NIB_LEN_VAR_LIMIT] setting

Software must set SRX_GBL_CTRL[NIB_LEN_VAR_LIMIT] to 1 to guarantee the performance of this module on this chip.

53.2 Introduction

The Single Edge Nibble Transmission (SENT) Receiver (SRX) module is a multi-channel receiver for receiving serial data frames which are being transmitted by a sensor implementing the SENT encoding scheme and present them to the CPU for further processing.

This module is based on the SAE J2716 information report titled "SENT - Single Edge Nibble Transmission for Automotive Applications" and released on January 27, 2010 (http://www.sae.org), Apr2007 and Feb2008. As per this standard, the SENT protocol is intended for use in applications where high resolution sensor data needs to be communicated from a sensor to an Engine Control Unit (ECU). It is intended as a replacement for the lower resolution methods of 10 bit A/Ds and PWM and as a simpler low-cost alternative to CAN or LIN. The implementation assumes that the sensor is a smart sensor containing a microprocessor or dedicated logic device (ASIC) to create the signal.

The SENT encoding scheme is a unidirectional communications scheme from the sensor / transmitting device to the controller /receiving device which does not include a coordination signal from the controller/receiving device. The sensor signal is transmitted as a series of pulses with data encoded as falling to falling edge periods.

53.2.1 Features

The SENT receiver module has the following features:

- The number of SENT channels supported are device-specific. See the device configuration details.
- Unified message read logic to transfer all messages received to system memory via interrupts or DMA
  - Programmable option per channel to support reading of messages via DMA or Interrupt
  - Internal round robin arbitration sequencer to loop across channels to read messages from each channel leaving CPU/DMA transparent to this process
  - All received messages are stored excluding the synchronization pulse (for Fast messages) and identifiers (for Slow Serial Messages)

    - Separate buffers for storing Fast and Slow Serial Messages
    - Separate DMA and Interrupts generated for Fast and Slow Serial Messages
- Supports compensation for variation in SENT Tx Clock up to ±25% and adjusts its measurement for drift in the Transmitter and Receiver clocks per channel
    - Calibration pulse correction factor per channel is available for use by software
- Implements for each channel all receiver diagnostics as specified by the SAE Specifications
- Internal 4-bit CRC calculator that can be configured to compute CRC in both Legacy and Recommended method (as given in SAE SENT Specification) for both Fast and Short Serial Message per channel
- Internal 6-bit CRC calculator for Enhanced Serial Messages per channel
- Supports all bit rates by having per channel configurable Rx clock periods from 3 µs to 90 µs
- Support for configurable number of data nibbles per channel
- Option to disable individual channel via programmable control bit
- Time stamping on start of each valid message (i.e. without errors) across all channels. This time stamp is appended to each message for synchronization of messages and for application to understand how much time has elapsed between successive sensor readings
- Support for pause pulse per channel. This can be enabled by programming a bit in the control register for that channel
- Input Programmable Filter on each channel to filter any glitches on input. This filter duration is programmable by the user software.
- 32-bit register interface for programming the module's registers and reading the received messages
- Works on two clocks, one for accurate receive operations (high frequency receiver clock or protocol clock) and other for message read logic (system bus clock) and the module can work irrespective of the frequency relationship between the two clocks. See Clocks and resets for more details on the clocks.

53.2.2 Modes of operation

Modes of operation.

53.2.2.1 Debug mode

In debug mode, message reception will be halted (without waiting for the current message being received to be completed) and all existing status and messages will be retained. DBG_FRZ bit is required to be set for this mode. On debug exit, a calibration length error (CAL_LEN_ERR) can be reported and the channels will start receiving messages from a valid calibration pulse.

53.2.2.2 Low power modes

SENT Receiver behavior is affected by chip-specific low power modes (for example, STOP or WAIT modes). These low power modes should either gate-off both the system bus clock and the protocol clocks or reduce their frequencies, depending on the application.

On exiting from these low power modes, calibration length error (CAL_LEN_ERR) can get set in channel status registers as the SENT sensor (transmitter) could be in the middle of a message transmission at that point. Thus, software should clear all error status bits after exiting the low power modes.

Modes where both system bus clock and protocol clock are gated: In this case, both message reception and interrupt / DMA request generation will halt. The internal receiver channels will freeze (since clock is not there) and will not receive any messages. Messages which are completely received before entering the low power mode can be read after the low power mode exit.

Modes where frequency of system bus clock is changed: For reduced power consumption the system bus clock can be reduced but not below a minimum value (See System bus clock requirements) to avoid overflow of received messages. Protocol clock should not go below the minimum value (See High frequency receiver clock (protocol clock) requirements) for accurate receive operations.

Note

Gating of clocks in the low power mode is chip-specific, see the chapter that refers how the modules behave in low power modes.

NOTE

If the SOC requests entry to debug or stop mode, the message being received currently is discarded and the SOC enters debug/ stop mode immediately. This does not affect the messages received completely prior to entering debug mode and will still be present on the message buffer and registers until they are read out.

NOTE

The message read registers [Channel ‘n' Fast Message Data Read Register (n = 0 to (CH-1)) (CHn_FMSG_DATA), Channel ‘n' Fast Message CRC Read Register (n = 0 to (CH-1)) (CHn_FMSG_CRC), Channel ‘n’ Fast Message Time Stamp Read Register (n = 0 to (CH-1)) (CHn_FMSG_TS), Channel ‘n’ Serial Message Read Register (Bit 3) (n = 0 to (CH-1)) (CHn_SMSG_BIT3), Channel ‘n’ Serial Message Read Register (Bit 2) (n = 0 to (CH-1)) (CHn_SMSG_BIT2), Channel ‘n’ Serial Message Time Stamp Read Register (n = 0 to (CH-1)) (CHn_SMSG_TS), DMA Fast Message Data Read Register (DMA_FMSG_DATA), DMA Fast Message CRC Read Register (DMA_FMSG_CRC), DMA Fast Message Time Stamp Read Register (DMA_FMSG_TS), DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3), DMA Slow Serial Message Bit2 Read Register (DMA_SMSG_BIT2) and DMA Slow Serial Message Time Stamp Read Register (DMA_SMSG_TS)] will appear to lose their contents in the following conditions:

- The very first message is being received but not yet completely received and the MCU enters debug or freeze mode, the current message reception will get discarded and message read registers (as mentioned above) will read zeros
- Auto clear functionality is enabled (GBL_CTRL[FAST_CLR] = 1). In this case, when first message is read, it will get clear the message read registers (due to auto clear functionality being enabled). On reading again, the message read registers (as mentioned above) might read zeros.

If the MCU requests entry to debug or stop mode, the message being received currently by SENT Receiver is discarded and the MCU enters debug/stop mode immediately. This does not affect the messages received completely, prior to entering debug

mode and these messages will still be present on the message buffer and registers until they are read out. Thus allow one message to be received completely and do not enable "Auto Clear" (GBL_CTRL[FAST_CLR] = 0), to allow messages to be read in debug or stop mode.

53.2.3 Block diagram

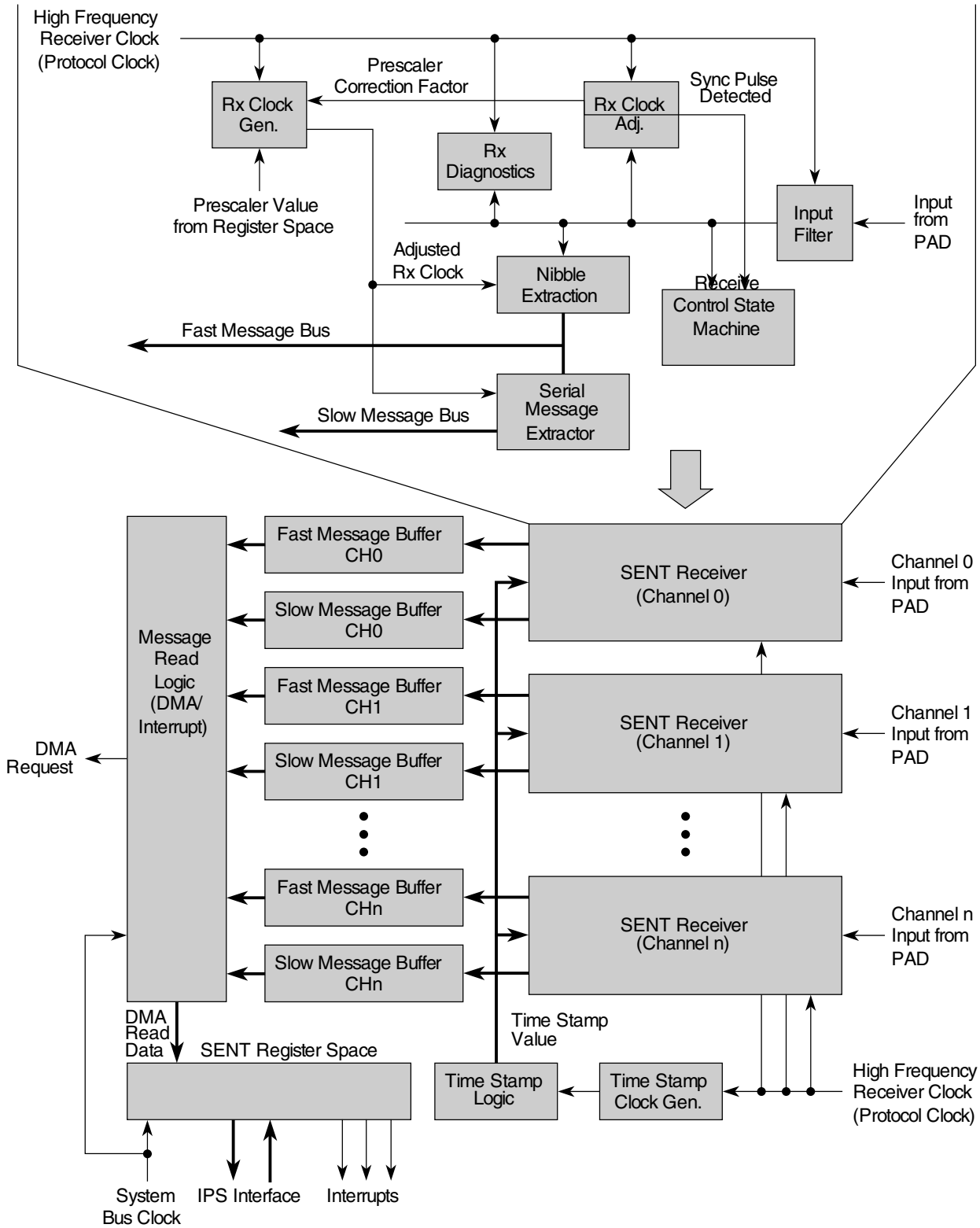


Figure 53-1. SENT Receiver block diagram

53.2.4 Design overview

The SENT Receiver Module is a unified receiver module comprising of a number of receiver channels and a unified DMA and Register Space for accessing the messages from each channel. Figure 53-1 shows the top level block diagram for the SENT Receiver module.

Each channel of the SENT Receiver is composed of a Clock Generation and Adjustment block to generate the receiver clock and adjust it for frequency variation in the transmitter clock. This block also detects the synchronization/calibration pulse which is present in every SENT message.

The SENT Receiver State Machine controls the sequencing of decoding the messages and combining the information as nibbles. In parallel a diagnostic block is running that monitors the incoming messages for any errors and also verifies the CRC of the received messages. The fast and serial message extractors collate the message nibbles and add the channel number and time stamp information to the messages.A free running time stamp counter is used to add the time stamp value to each message received. The value of this counter is input to each channel and the channel logic appends the time stamp to each message. These messages are then formatted to be stored in the corresponding message read buffer. Only the messages that pass the diagnostics checks are forwarded to be stored in the buffers.

The SENT Receiver provides two mechanisms for reading the received messages, which are via DMA transfers and via Interrupts. Based on the control bit settings in the control registers, a DMA request or Interrupt is asserted to signal the DMA controller or CPU that messages are available to be read.

The free running time stamp counter, register block and DMA interface are common for all channels configured. The register space contains all the necessary registers for all channels. The register space is 32-bit aligned and accessible via 8-/16-/32-bit accesses. The registers for all un-configured channels are marked as reserved.

53.2.4.1 Functional overview

After reset, the SENT module comes up in the disabled state. All channel enable bits and the global enable bit are set to 0 and the user software must program the SENT module parameters correctly before enabling it. The user software must correctly configure all parameters to avoid any underflow or overflow in the message buffers.

Once enabled, the Receiver Clock Generation and Adjustment block detects the synchronization pulse. Until then, the state machine blocks all subsequent operations of the receiver. Once the synchronization pulse is detected, the receiver clock tick period is compensated for variation in transmitter clock tick period and the state machine then waits for the reception of data nibbles. The adjusted receiver clock is used to sample the data nibbles from the incoming serial message. The state machine waits until the status and communication nibble and configured number of data nibbles are received. The state machine then waits for the pause pulse (if configured to receive a pause pulse). The fast and serial message extractor block provides the state machine with the information on how many nibbles have been received. All diagnostics checks are computed in parallel. Once all required data nibbles are received, the CRC is checked, time stamp and channel number is added to the message. The time stamp for a Fast Message is sampled at the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) and is stored in the message read buffer along with the message. This final message is stored in the message buffer as described in the subsequent sections. Any error detected during the receive process causes the status bits in the corresponding channel's status register to be set.

The above process repeats across each enabled channel in parallel and keeps pushing error free messages into the buffers on high frequency receiver clock. The module's message read logic synchronizes all these messages on the bus clock and pops out these messages on any DMA or CPU read access. Any DMA underflow errors are stored in the global status register while the overflow errors are stored in the corresponding channel's status register. In case of an overflow in Fast Messages, the newer messages received will overwrite the previous message.

For detailed description on the complete functionality of each block, see Functional description.

53.3 External signal description

This section lists all inputs to the SENT Receiver block.

53.3.1 SENT_RX [(CH-1):0]

These pins serve as the serial data input for each of the SENT Receiver channel. There is one bit per channel supported by the device. The SENT sensor input pads on the device using this module should have pull-up enabled to ensure a 1'b1 is seen at the module input when SENT sensor is not driving any data.

Note

The value of CH is device-specific. See the device configuration details.

53.4 Memory map and register definition

This section provides a detailed description of all registers accessible in the SENT Receiver block.

The following table gives an overview on all SENT Receiver registers. The memory map comprises of 32-bit aligned registers which can be accessed via 8-bit, 16-bit, or 32-bit accesses. Write access to reserved locations will generate transfer error. Read access to reserved locations will also generate transfer error and the read data bus will show all 0s. The Memory Map and complete module is in Big Endian Format.

The module will not check for correctness of programmed values in Register Space and software must ensure that correct values are being written.

NOTE

Register Address = Base Address + Address Offset, where the Base Address is defined at the MCU level and the Address Offset is defined at the module level. The number of registers are device specific. See the device configuration section for details.

NOTE

The value of the CH is device-specific. See the device configuration section for details.

SRX memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Global Control Register (SRX_GBL_CTRL)	32	R/W	0000_0010h	53.4.1/2322
4	Channel Enable Register (SRX_CHNL_EN)	32	R/W	0000_0000h	53.4.2/2323
8	Global Status Register (SRX_GBL_STATUS)	32	w1c	0000_0000h	53.4.3/2325
C	Fast Message Ready Status Register (SRX_FMSG_RDY)	32	R/W	0000_0000h	53.4.4/2326
10	Slow Serial Message Ready Status Register (SRX_SMSG_RDY)	32	R/W	0000_0000h	53.4.5/2327
18	Data Control Register 1 (SRX_DATA_CTRL1)	32	R/W	1111_1100h	53.4.6/2328

Table continues on the next page...

SRX memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
28	Fast Message DMA Control Register (SRX_FDMA_CTRL)	32	R/W	0000_0000h	53.4.7/2330
2C	Slow Serial Message DMA Control Register (SRX_SDMA_CTRL)	32	R/W	0000_0000h	53.4.8/2330
34	Fast Message Ready Interrupt Control Register (SRX_FRDY_IE)	32	R/W	0000_0000h	53.4.9/2331
38	Slow Serial Message Ready Interrupt Enable Register (SRX_SRDY_IE)	32	R/W	0000_0000h	53.4.10/ 2332
40	DMA Fast Message Data Read Register (SRX_DMA_FMSG_DATA)	32	R	0000_0000h	53.4.11/ 2332
44	DMA Fast Message CRC Read Register (SRX_DMA_FMSG_CRC)	32	R	0000_0000h	53.4.12/ 2333
48	DMA Fast Message Time Stamp Read Register (SRX_DMA_FMSG_TS)	32	R	0000_0000h	53.4.13/ 2334
50	DMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3)	32	R	0000_0000h	53.4.14/ 2334
54	DMA Slow Serial Message Bit2 Read Register (SRX_DMA_SMSG_BIT2)	32	R	0000_0000h	53.4.15/ 2335
58	DMA Slow Serial Message Time Stamp Read Register (SRX_DMA_SMSG_TS)	32	R	0000_0000h	53.4.16/ 2336
60	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH0_CLK_CTRL)	32	R/W	0000_8000h	53.4.17/ 2337
64	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH0_STATUS)	32	w1c	0000_0000h	53.4.18/ 2339
68	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH0_CONFIG)	32	R/W	0000_0104h	53.4.19/ 2342
70	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH1_CLK_CTRL)	32	R/W	0000_8000h	53.4.17/ 2337
74	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH1_STATUS)	32	w1c	0000_0000h	53.4.18/ 2339
78	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH1_CONFIG)	32	R/W	0000_0104h	53.4.19/ 2342
80	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH2_CLK_CTRL)	32	R/W	0000_8000h	53.4.17/ 2337
84	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH2_STATUS)	32	w1c	0000_0000h	53.4.18/ 2339
88	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH2_CONFIG)	32	R/W	0000_0104h	53.4.19/ 2342
90	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH3_CLK_CTRL)	32	R/W	0000_8000h	53.4.17/ 2337
94	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH3_STATUS)	32	w1c	0000_0000h	53.4.18/ 2339
98	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH3_CONFIG)	32	R/W	0000_0104h	53.4.19/ 2342
A0	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH4_CLK_CTRL)	32	R/W	0000_8000h	53.4.17/ 2337

Table continues on the next page...

SRX memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A4	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH4_STATUS)	32	w1c	0000_0000h	53.4.18/ 2339
A8	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH4_CONFIG)	32	R/W	0000_0104h	53.4.19/ 2342
B0	Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH5_CLK_CTRL)	32	R/W	0000_8000h	53.4.17/ 2337
B4	Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH5_STATUS)	32	w1c	0000_0000h	53.4.18/ 2339
B8	Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH5_CONFIG)	32	R/W	0000_0104h	53.4.19/ 2342
160	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH0_FMSG_DATA)	32	R	0000_0000h	53.4.20/ 2345
164	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH0_FMSG_CRC)	32	R	0000_0000h	53.4.21/ 2346
168	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH0_FMSG_TS)	32	R	0000_0000h	53.4.22/ 2346
16C	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH0_SMSG_BIT3)	32	R	0000_0000h	53.4.23/ 2347
170	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH0_SMSG_BIT2)	32	R	0000_0000h	53.4.24/ 2348
174	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH0_SMSG_TS)	32	R	0000_0000h	53.4.25/ 2349
178	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH1_FMSG_DATA)	32	R	0000_0000h	53.4.20/ 2345
17C	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH1_FMSG_CRC)	32	R	0000_0000h	53.4.21/ 2346
180	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH1_FMSG_TS)	32	R	0000_0000h	53.4.22/ 2346
184	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH1_SMSG_BIT3)	32	R	0000_0000h	53.4.23/ 2347
188	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH1_SMSG_BIT2)	32	R	0000_0000h	53.4.24/ 2348
18C	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH1_SMSG_TS)	32	R	0000_0000h	53.4.25/ 2349
190	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH2_FMSG_DATA)	32	R	0000_0000h	53.4.20/ 2345
194	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH2_FMSG_CRC)	32	R	0000_0000h	53.4.21/ 2346
198	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH2_FMSG_TS)	32	R	0000_0000h	53.4.22/ 2346
19C	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH2_SMSG_BIT3)	32	R	0000_0000h	53.4.23/ 2347
1A0	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH2_SMSG_BIT2)	32	R	0000_0000h	53.4.24/ 2348

Table continues on the next page...

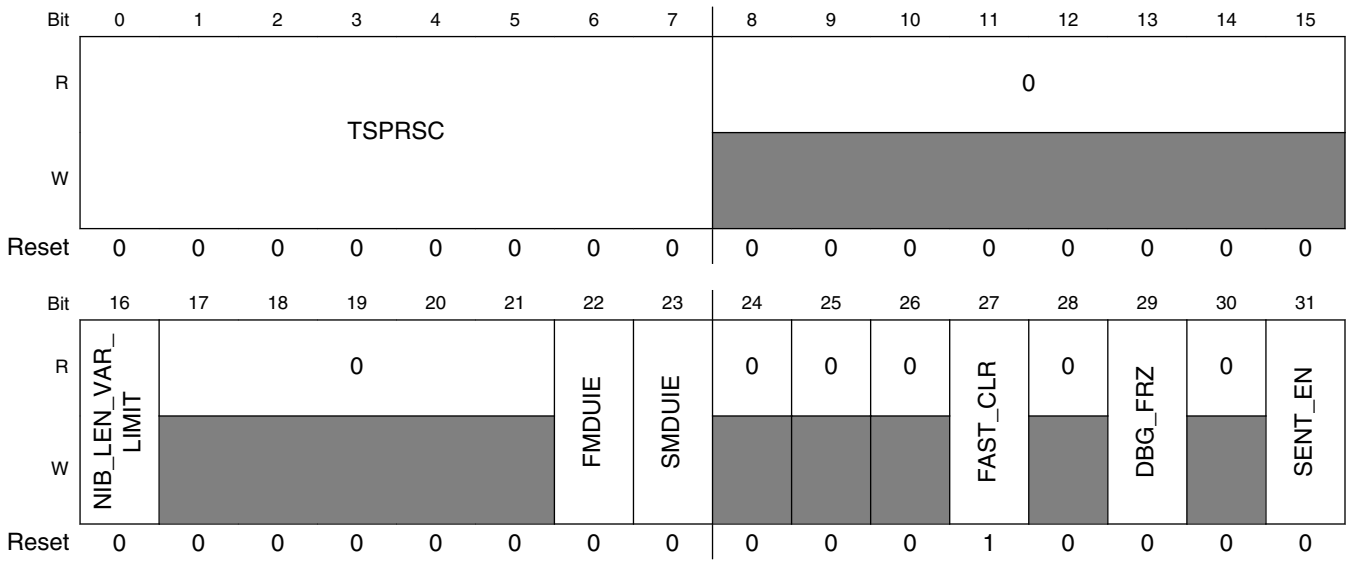
SRX memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1A4	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH2_SMSG_TS)	32	R	0000_0000h	53.4.25/ 2349
1A8	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH3_FMSG_DATA)	32	R	0000_0000h	53.4.20/ 2345
1AC	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH3_FMSG_CRC)	32	R	0000_0000h	53.4.21/ 2346
1B0	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH3_FMSG_TS)	32	R	0000_0000h	53.4.22/ 2346
1B4	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH3_SMSG_BIT3)	32	R	0000_0000h	53.4.23/ 2347
1B8	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH3_SMSG_BIT2)	32	R	0000_0000h	53.4.24/ 2348
1BC	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH3_SMSG_TS)	32	R	0000_0000h	53.4.25/ 2349
1C0	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH4_FMSG_DATA)	32	R	0000_0000h	53.4.20/ 2345
1C4	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH4_FMSG_CRC)	32	R	0000_0000h	53.4.21/ 2346
1C8	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH4_FMSG_TS)	32	R	0000_0000h	53.4.22/ 2346
1CC	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH4_SMSG_BIT3)	32	R	0000_0000h	53.4.23/ 2347
1D0	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH4_SMSG_BIT2)	32	R	0000_0000h	53.4.24/ 2348
1D4	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH4_SMSG_TS)	32	R	0000_0000h	53.4.25/ 2349
1D8	Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH5_FMSG_DATA)	32	R	0000_0000h	53.4.20/ 2345
1DC	Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH5_FMSG_CRC)	32	R	0000_0000h	53.4.21/ 2346
1E0	Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH5_FMSG_TS)	32	R	0000_0000h	53.4.22/ 2346
1E4	Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH5_SMSG_BIT3)	32	R	0000_0000h	53.4.23/ 2347
1E8	Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH5_SMSG_BIT2)	32	R	0000_0000h	53.4.24/ 2348
1EC	Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH5_SMSG_TS)	32	R	0000_0000h	53.4.25/ 2349

53.4.1 Global Control Register (SRX_GBL_CTRL)

This register provides the global control and setting for the SENT Receiver Module. The SENT_EN bit must be programmed after all other modules settings and control bits have been written to.

Address: 0h base + 0h offset = 0h



SRX_GBL_CTRL field descriptions

Field	Description
0–7 TSPRSC	Time Stamp Prescaler Value. Value in number of clock cycles of high frequency receiver clock set in order to obtain a clock frequency for the time stamp counter block. 0 to 255 - Time Stamp Clock = (High Frequency Clock)/(Value + 1). Value set by user software should be such that the clock period is at least 1 µs. If the high frequency clock input is 75 MHz, then the value set in this field would be 0x4A (or 74) to obtain a 1 µs clock. Note: Since the time stamp clock is common for all channels, the timestamp clock frequency must be at least twice the fastest receiver channel in the application. This is to ensure that the time stamp clock is fast enough to show change in timestamp from message to message.
8–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 NIB_LEN_VAR_LIMIT	Decides the allowable jitter/clock drift limit for nibble length measurement. **NOTE:** This bit should be set to 1 for proper functioning of Sent Rx. 0 Allows a jitter/clock drift of approx upto 12.5% of transmitter clock tick period. This actual allowed jitter/drift value is dependent on the transmitter clock tick period and HF receiver clock period. 1 Allows a jitter/clock drift as per SAE spec (50%).

Table continues on the next page...

SRX_GBL_CTRL field descriptions (continued)

Field	Description
17–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 FMDUIE	Fast Message DMA Underflow Interrupt Enable: Enables interrupt assertion when there is an underflow condition on Fast Message DMA registers. Default value is 0. 0 Interrupt is disabled 1 Interrupt is enabled
23 SMDUIE	Slow Serial Message DMA Underflow Interrupt Enable: Enables interrupt assertion when there is an underflow condition on Slow Message DMA registers. Default value is 0. 0 Interrupt is disabled 1 Interrupt is enabled
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 FAST_CLR	Fast Clearing Enable bit. Enables clearing of ready status (for both fast and slow) bit automatically when corresponding message is read. Default value is 1. See Fast Message Ready Status Register (SRX_FMSG_RDY) and Slow Serial Message Ready Status Register (SRX_SMSG_RDY) for details on the ready status bits. 0 Fast Clearing is disabled. Bits will be cleared by writing 1 1 Fast Clearing is enabled
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 DBG_FRZ	Debug Freeze. This bit will enable the debug mode support. SENT Module will freeze in debug mode if this bit is set. Default is 0. 0 No effect in debug mode 1 Freeze in debug mode
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 SENT_EN	SENT Receiver Global Enable. This bit enables or disables the complete receiver module irrespective of the setting of individual channel enable bits. User software must program all channel parameters before setting this bit. Default is 0. 0 Entire module is disabled 1 Module is enabled

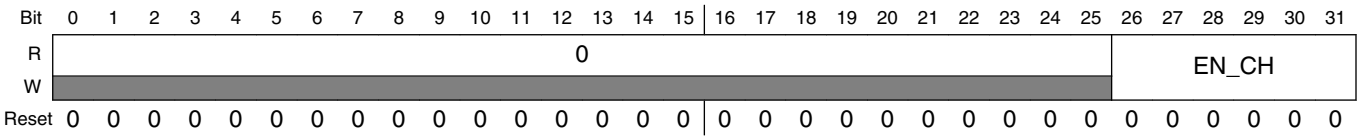
53.4.2 Channel Enable Register (SRX_CHNL_EN)

This register is used to enable individual SENT Receiver channels in the module. The user software must ensure all channel parameters and control settings have been programmed before writing to this bit. In case these parameters are changed while the channel is enabled, normal operation is not guaranteed.

NOTE

Reads of bits beyond those for the supported number of channels should be ignored, and these bits must be written to 0.

Address: 0h base + 4h offset = 4h



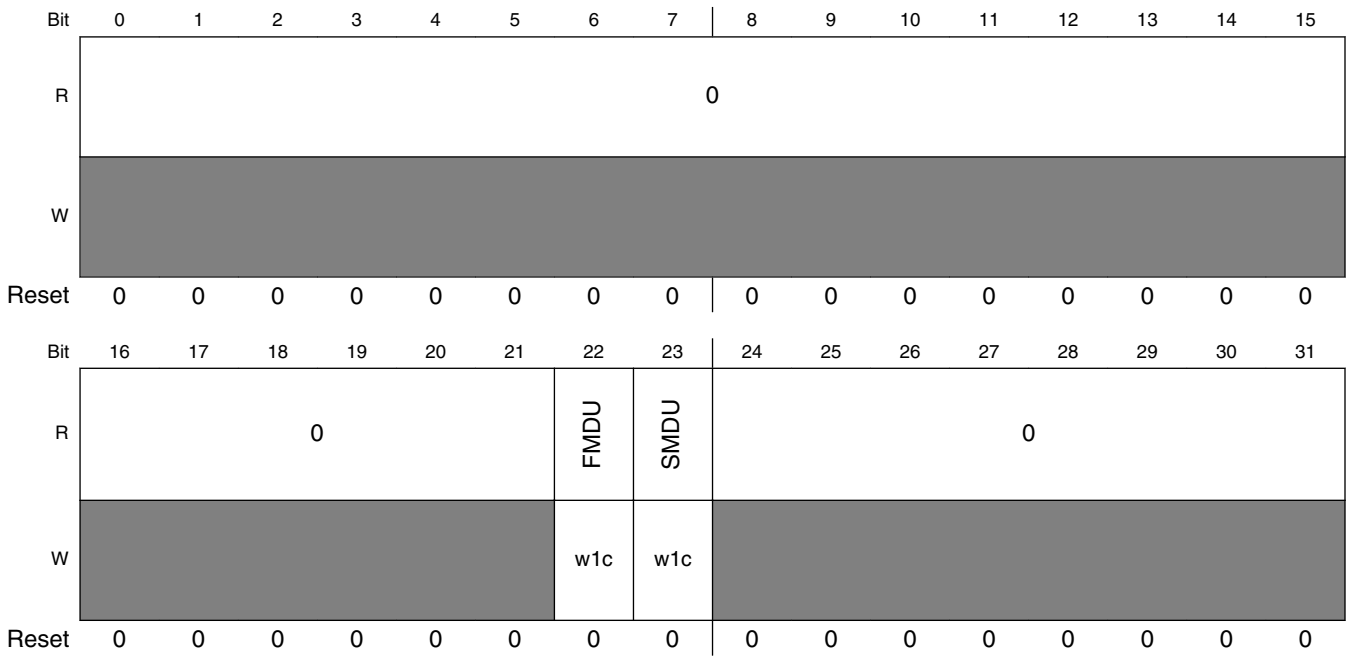
SRX_CHNL_EN field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 EN_CH	Enable bits for channels 5 to 0 . All channels are disabled after reset. User software must program all channel parameters before setting this bit. Once set, other channel parameters in registers should not be changed until this bit is cleared. Only Interrupt Enable and Status bits can be written to. 0 Channel is disabled 1 Channel is enabled. Channel parameter registers are locked for writing

53.4.3 Global Status Register (SRX_GBL_STATUS)

This register provides the status of various conditions in the module that are common across all channels.

Address: 0h base + 8h offset = 8h



SRX_GBL_STATUS field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 FMDU	Fast Message DMA Underflow: Interrupt Status bit that indicates when there is an underflow in Fast Message buffers in any channel which has DMA enabled. This bit is set when there is no message in the DMA buffers for read and the software still reads the Fast Message DMA register. Default value is 0. 0 No underflow 1 Underflow occurred
23 SMDU	Slow Serial Message DMA Underflow: Interrupt Status bit that indicates when there is an Underflow in Slow Serial Message DMA buffers on channel which has DMA enabled. The bit is set when there is no message in the DMA buffers for read and the software still reads the Slow Serial Message registers. Default value is 0. 0 No underflow 1 Underflow occurred
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.4 Fast Message Ready Status Register (SRX_FMSG_RDY)

The bits of this register are asserted when a Fast Message arrives on a corresponding channel that is not configured to be read via DMA. If a channel is not configured for DMA (viaFast Message DMA Control Register (SRX_FDMA_CTRL)), it is assumed to be read via Interrupt or Polling this status bit. Even if the Interrupt Enable bit for that channel is not set (viaFast Message Ready Interrupt Control Register (SRX_FRDY_IE)) and the channel is enabled, the message ready status bit in this register will be asserted on Fast Message reception. Software must ensure that if a channel is enabled; either DMA or Interrupt must be enabled for that channel. Proper flow cannot be guaranteed if neither is selected.

These bits can be configured to be cleared via "Write 1 to Clear" or "Fast Clearing Mechanism". The configuration is done by writing to the FAST_CLR bit in theGlobal Control Register (SRX_GBL_CTRL) .

NOTE

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																										F_RDY					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_FMSG_RDY field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 F_RDY	Fast Message Data Ready Status bits for channels 5 to 0 . Each bit indicates that a Fast Message is ready to be read by CPU from that specific channel. This bit is asserted only when DMA is NOT enabled for that channel (from the DMA Control Register) and an error free message is received, irrespective of the value of corresponding enable bit in Message Ready Interrupt Control Register. Reading the message from the corresponding memory-mapped Fast Message Read Registers would automatically clear this bit (Fast Clearing Mechanism). **NOTE:** These bits are either "write 1 to clear" (w1c) or self-clearing depending on the value of the FAST_CLR BIT in the Global Control Register (GBL_CTRL). The clearing mechanism is selected by user by programming the FAST_CLR bit in the Global Control register. See Section Global Control Register (SRX_GBL_CTRL) for details on FAST_CLR bit. **NOTE:** When using the fast clearing option, these bits will not clear till the complete message is read out which requires the CPU to read all 3 fast message registers via 32-bit read accesses. For 8/16-bit accesses fast clearing mechanism should not be used as it is not supported.

Table continues on the next page...

SRX_FMSG_RDY field descriptions (continued)

Field	Description
	NOTE: When the fast clearing option is not configured, the bits will be cleared only if 1 is written to those bits. 0 No Fast Message is available to be read out via Interrupt 1 Fast Message is available to be read out via Interrupt

53.4.5 Slow Serial Message Ready Status Register (SRX_SMSG_RDY)

The bits of this register are asserted when a Slow Serial Message arrives on a corresponding channel that is not configured to be read via DMA. If a channel is not configured for DMA (viaFast Message DMA Control Register (SRX_FDMA_CTRL) andSlow Serial Message DMA Control Register (SRX_SDMA_CTRL)), it is assumed to be read via Interrupt or Polling this status bit. Even if the Interrupt Enable bit for that channel is not set (viaFast Message Ready Interrupt Control Register (SRX_FRDY_IE) andSlow Serial Message Ready Interrupt Enable Register (SRX_SRDY_IE)) and the channel is enabled, the message ready status bit in this register will be asserted on Slow Message reception. Software must ensure that if a channel is enabled; either DMA or Interrupt must be enabled for that channel. Erroneous flow can happen if neither is selected.

These bits can be configured to be cleared via "Write 1 to Clear" or "Fast Clearing Mechanism". The configuration is done by writing to the FAST_CLR bit in theGlobal Control Register (SRX_GBL_CTRL).

NOTE

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																															S_RDY
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_SMSG_RDY field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SRX_SMSG_RDY field descriptions (continued)

Field	Description
26–31 S_RDY	Slow Serial Message Data Ready Status bits for channels 5 to 0 . Each bit indicates that a Slow Serial Message is ready to be read by CPU from that specific channel. This bit is asserted only when DMA is NOT enabled for that channel (from the DMA Control Register) and an error free message is received, irrespective of the value of corresponding enable bit in Message Ready Interrupt Control Register. Reading the message from the corresponding memory-mapped Serial Message Read Registers would automatically clear this bit (Fast Clearing Mechanism). **NOTE:** These bits are either "write 1 to clear" (w1c) or self-clearing depending on the value of the FAST_CLR bit in the Global Control Register (GBL_CTRL). The clearing mechanism is selected by user by programming the FAST_CLR bit in the Global Control register. See Section Global Control Register (SRX_GBL_CTRL) for details on FAST_CLR bit. **NOTE:** When using the fast clearing option, these bits will not clear till the complete message is read out which requires the CPU to read all 3 Fast Message registers via 32-bit read accesses. For 8/16-bit accesses fast clearing mechanism should not be used as it is not supported. **NOTE:** When the fast clearing option is not configured, the bits will be cleared only if 1 is written to those bits. 0 No Slow Serial Message is available to be read out via Interrupt 1 Slow Serial Message is available to be read out via Interrupt

53.4.6 Data Control Register 1 (SRX_DATA_CTRL1)

The Data Control Register 1 (DATA_CTRL1) is described below.

NOTE

Reset value changes depending on the number of channels that are enabled on the device.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	NIBBCH0			0	NIBBCH1			0	NIBBCH2			0	NIBBCH3		
W																
Reset	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	NIBBCH4			0	NIBBCH5			0	Reserved			0	Reserved		
W																
Reset	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0

SRX_DATA_CTRL1 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 NIBBCH0	Number of Data Nibbles supported in Channel 0

Table continues on the next page...

SRX_DATA_CTRL1 field descriptions (continued)

Field	Description
	0 and 7 Invalid value. Must not be used 1 to 6 Number of data nibbles to be supported by respective channel
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 NIBBCH1	Number of Data Nibbles supported in Channel 1 0 and 7 Invalid value. Must not be used 1 to 6 Number of data nibbles to be supported by respective channel
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–11 NIBBCH2	Number of Data Nibbles supported in Channel 2 0 and 7 Invalid value. Must not be used 1 to 6 Number of data nibbles to be supported by respective channel
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 NIBBCH3	Number of Data Nibbles supported in Channel 3 0 and 7 Invalid value. Must not be used 1 to 6 Number of data nibbles to be supported by respective channel
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–19 NIBBCH4	Number of Data Nibbles supported in Channel 4 0 and 7 Invalid value. Must not be used 1 to 6 Number of data nibbles to be supported by respective channel
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–23 NIBBCH5	Number of Data Nibbles supported in Channel 5 0 and 7 Invalid value. Must not be used 1 to 6 Number of data nibbles to be supported by respective channel
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–27 Reserved	This field is reserved.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 Reserved	This field is reserved.

53.4.7 Fast Message DMA Control Register (SRX_FDMA_CTRL)

NOTE

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

RW

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																										FDMA_EN					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_FDMA_CTRL field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 FDMA_EN	Enable DMA for Fast Messages on channels 5 to 0 . These bits are writable when corresponding Fast Message Ready Interrupt Enable bits are set to 0. 0 DMA for Fast Messages is disabled 1 DMA for Fast Messages is enabled

53.4.8 Slow Serial Message DMA Control Register (SRX_SDMA_CTRL)

NOTE

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																										SDMA_EN					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_SDMA_CTRL field descriptions

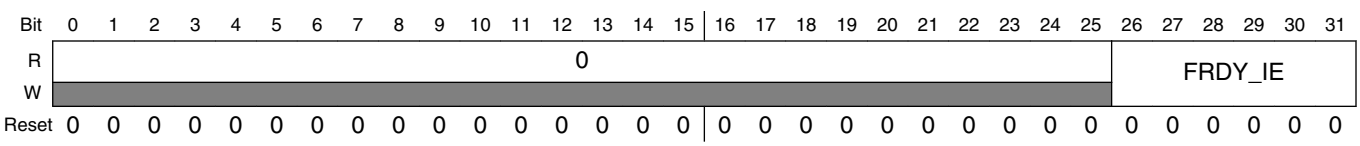
Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 SDMA_EN	Enable DMA for Slow Serial Messages on channels 5 to 0 . These bits are writable when corresponding Slow Serial Message Ready Interrupt Enable bits are set to 0. 0 DMA for Slow Serial Messages is disabled 1 DMA for Slow Serial Messages is enabled

53.4.9 Fast Message Ready Interrupt Control Register (SRX_FRDY_IE)

NOTE

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

Address: 0h base + 34h offset = 34h



SRX_FRDY_IE field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 FRDY_IE	Enable for Fast Message Ready Interrupt on channels 5 to 0 . These bits are writable when corresponding DMA enable bits are set to 0 in the Fast Message DMA Control Register. The availability of message is indicated via assertion of corresponding bit in Fast Message Ready Status Register irrespective of the value set in corresponding interrupt enable bit and DMA is not enabled. When a bit is asserted in this register, an interrupt for corresponding channel is generated. 0 Interrupt on reception of Fast Messages is disabled 1 Interrupt on reception of Fast Messages is enabled

53.4.10 Slow Serial Message Ready Interrupt Enable Register (SRX_SRDY_IE)

NOTE

Reads of bits beyond those for the supported number of channels should be ignored. Writes to these bits is ignored by the Receiver.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																										SRDY_IE					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_SRDY_IE field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 SRDY_IE	Enable for Slow Serial Message Ready Interrupt on channels 5 to 0 . These bits are writable when corresponding DMA enable bits are set to 0 in the Slow Serial DMA Control Register. The availability of message is indicated via assertion of corresponding bit in Slow Serial Message Ready Status Register irrespective of the value set in corresponding interrupt enable bit and DMA is not enabled. When a bit in this register is asserted, an interrupt for corresponding channel is generated. 0 Interrupt on reception of Slow Serial Messages is disabled 1 Interrupt on reception of Slow Serial Messages is enabled

53.4.11 DMA Fast Message Data Read Register (SRX_DMA_FMSG_DATA)

DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_DATA) via DMA.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CHNUM				SCNIB				DNIB1				DNIB2				DNIB3				DNIB4				DNIB5				DNIB6			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_DMA_FMSG_DATA field descriptions

Field	Description
0–3 CHNUM	Channel Number. Indicates the channel number of the received message being read by DMA. Valid values are 0 to 5 .
4–7 SCNIB	Status and Communication Nibble of message
8–11 DNIB1	Data Nibble 1. Always supported
12–15 DNIB2	Data Nibble 2. Should be ignored if number of data nibbles supported by channel is 1
16–19 DNIB3	Data Nibble 3. Should be ignored if number of data nibbles supported by channel is 2 or less
20–23 DNIB4	Data Nibble 4. Should be ignored if number of data nibbles supported by channel is 3 or less
24–27 DNIB5	Data Nibble 5. Should be ignored if number of data nibbles supported by channel is 4 or less
28–31 DNIB6	Data Nibble 6. Should be ignored if number of data nibbles supported by channel is 5 or less

53.4.12 DMA Fast Message CRC Read Register (SRX_DMA_FMSG_CRC)

DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_CRC) via DMA.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												CRC4b				0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_DMA_FMSG_CRC field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 CRC4b	4-bit CRC value of the message.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.13 DMA Fast Message Time Stamp Read Register (SRX_DMA_FMSG_TS)

The resolution of the time stamp is set by programming the Time Stamp Prescaler Value in the Global Control Register (SRX_GBL_CTRL) register. DMA Source Size must be set to 32 bits when reading this register (DMA_FMSG_TS) via DMA.

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_DMA_FMSG_TS field descriptions

Field	Description
0–31 TS	Time Stamp for received message. Indicates the relative time when the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) was detected

53.4.14 DMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3)

The above register (DMA_SMSG_BIT3) is common for all types of slow serial messages. Bit C defines the type of enhanced packet and bit TYPE defines whether it is short serial or enhanced serial message. Since this register is used to read both Short and Enhanced Serial Messages, the register fields have been placed in such a way to match the bit positions in actual message defined by SAE Specification as shown inTable 53-83,Table 53-84 andTable 53-85.Table 53-83 andTable 53-84 shows the enhanced serial messages format to be used when reading this register for these two types of messages.Table 53-85 shows the short serial message.

DMA Source Size must be set to 32 bits when reading this register via DMA.

NOTE

If the TYPE bit in the register reads '0', then remainder bits (for CFG, ID and DATA) will be redundant and read zeros. The short serial message will be located in DMA_SMSG_BIT2 register fields. If TYPE bit is '1', then the message in register is Enhanced Serial Message and using CFG bit, software can decode the ID and DATA fields.

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHNUM				TYPE	0										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					CFG	ID7_4_ID3_0				0	ID3_0_DATA15_12				0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_DMA_SMSG_BIT3 field descriptions

Field	Description
0–3 CHNUM	Channel Number. Indicates the channel number of the received message being read by DMA. Valid values are 0 to 5 .
4 TYPE	Serial Message Type. Indicates the type of Serial Message received 0 Short Serial Message 1 Enhanced Serial Message
5–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 CFG	Configuration bit 'C'. Indicates the type of Enhanced Serial message 0 Enhanced Serial Message with 8-bit ID field 1 Enhanced Serial Message with 4-bit ID field
22–25 ID7_4_ID3_0	ID Field. Value depends on setting of 'C' bit. If 'C' bit is 0, this field is ID[7:4] and if the 'C' bit is 1, this field is ID[3:0]
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–30 ID3_0_DATA15_12	ID or Data Field. Value depends on setting of 'C' bit. If 'C' bit is 0, this field is ID[3:0] and if the 'C' bit is 1, this field is Data[15:12]
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.15 DMA Slow Serial Message Bit2 Read Register (SRX_DMA_SMSG_BIT2)

This register (DMA_SMSG_BIT2) is common for all types of slow serial messages. Hence, the register fields have been placed in such a way to match the bit positions in actual message defined by SAE Specification as shown inTable 53-83,Table 53-84 andTable 53-85.Table 53-83, andTable 53-84 shows the enhanced serial messages format to be used when reading this register for these two types of messages.Table 53-85 shows the short serial message.

DMA Source Size must be set to 32 bits when reading this register via DMA.

Address: 0h base + 54h offset = 54h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										SMCRC						0				DATA[11:0]											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_DMA_SMSG_BIT2 field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 SMCRC	6-bit CRC value of the message. When TYPE bit in DMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3) is set to 0, this CRC is of 4-bits for Short Serial Messages and when TYPE bit is set to 1, this CRC is of 6-bits for Enhanced Serial Messages.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 DATA[11:0]	12-bit Data Value

53.4.16 DMA Slow Serial Message Time Stamp Read Register (SRX_DMA_SMSG_TS)

The resolution of the time stamp is set by programming the Time Stamp Prescaler Value in the Global Control Register (SRX_GBL_CTRL) register.

DMA Source Size must be set to 32 bits when reading this register (DMA_SMSG_TS) via DMA.

Address: 0h base + 58h offset = 58h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_DMA_SMSG_TS field descriptions

Field	Description
0–31 TS	Time Stamp for the received message. Indicates the relative time of detection of the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) of the last Fast Message that completed the Serial Message. Time Stamp for Slow Serial Messages is not latched when the Slow Serial Message starts but when the Slow Serial Message is completely received.

53.4.17 Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH*n*_CLK_CTRL)

NOTE

The value of the CH is device-specific. See the device configuration section for details.

The SENT Receiver module generates the Receiver Sensor Clock (or simply Receiver Clock) internally to sample the incoming filtered sensor data. The user software must program the prescaler value (for Receiver Clock) to match the ideal Sensor Tx Clock period. The generated Receiver Clock is compensated for variations and clock drift that might occur in the Sensor Tx Clock. SeeAdjustment for variation in sensor (Tx) clock for more details.

This register configures the generation and subsequent compensation logic of the Receiver Clock. User must program the contents of this register before enabling the channel by writing to the corresponding channel enable bit in theChannel Enable Register (SRX_CHNL_EN) .

Address: 0h base + 60h offset + (16d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	CM_PRSC														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COMP_EN	0	PRSC													
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CH*n*_CLK_CTRL field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–15 CM_PRSC	Compensated Prescaler Value. This is the compensated prescaler value calculated by Clock Compensation logic and this value is used to compensate for sensor Tx clock variation. This value is indicated as a read only field for use by user software, if needed. This is a 15-bit value. '= Rx Prescaler Value' No variation, hence no compensation or Compensation is disabled '< Rx Prescaler Value' Tx Clock is SLOWER than the programmed receive clock '> Rx Prescaler Value' Tx Clock is FASTER than the programmed receive clock Thus, the new Receiver Clock Tick Period can be computed by software as Compensated Prescaler Value x High Frequency Receiver Clock Period Note: The value provided in this field is ceil value of actual tick period as measured by the Receiver.
16 COMP_EN	Compensation Enable. To enable compensation logic to adjust the receiver clock against variation in Tx clock on this channel. User must set the Prescaler Value before enabling this bit. 0 Compensation is disabled 1 Compensation is enabled
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–31 PRSC	Rx Prescaler Value. This factor is required to generate required Receiver Clock tick from the High Frequency Receiver Clock. This value can only be written when Channel Enable bit is Zero. Hence this value must be programmed before enabling the channel. If required receiver clock tick period (Trx_clk) is not divisible by the high frequency receiver clock (protocol clock) period (Thf_clk), the required prescaler value will have a fractional part which cannot be programmed. In this case then user is advised to program the prescaler value to the lower integer value. 0 Prescaler is bypassed. Should not be used. Non-Zero Value Enables Prescaler The required value is computed by software as follows: $\text{Prescaler_value} = \text{floor}\left(\frac{T_{\text{rx_clk}}}{T_{\text{hf_clk}}}\right)$ (both clock periods taken in µs). Please seeHigh frequency receiver clock (protocol clock) requirements before setting this value. e.g. if the required receiver clock tick period is 43 µs and the High Frequency Receiver clock is selected to be 62.5 MHz (or 0.016 µs) then, Prescaler Value = 43/0.016 = 2687.5. Hence user must program 2687 in this register.

53.4.18 Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH*n*_STATUS)

This registers stores information about certain events and error conditions that occur in the channel during the message reception process. The assertion of these bit is not linked to any message but it is a general alert to the CPU that a particular condition (error or normal event) happened in the recent past. Messages are automatically discarded when an error is detected.

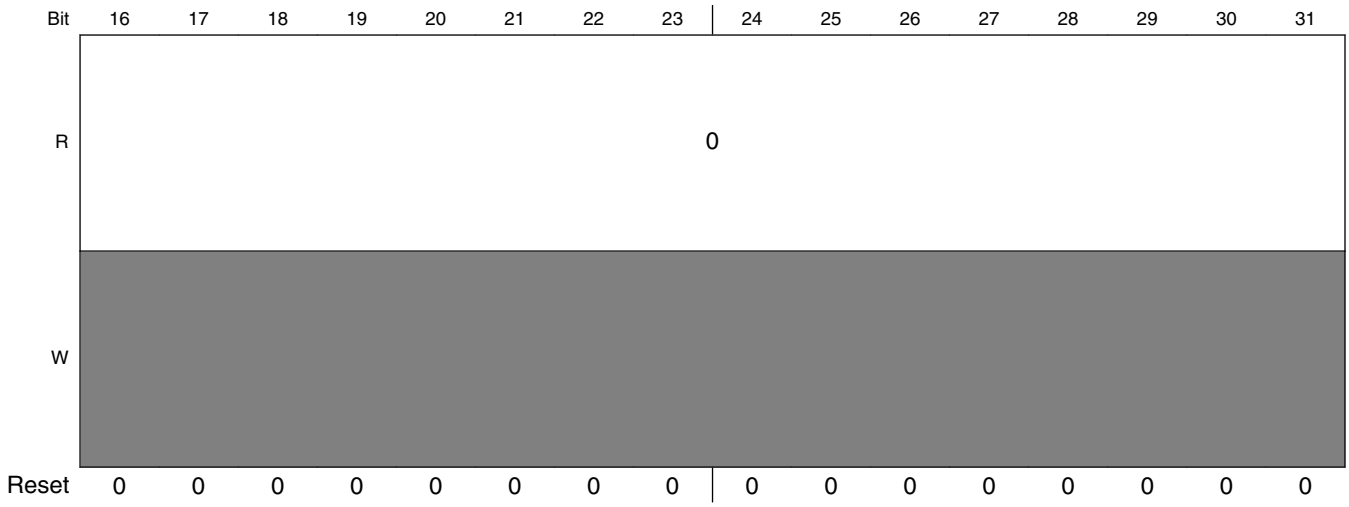
User software can clear these bit at any time by writing 1 to them. This register can be updated at any time irrespective of the setting in the Channel Enable Register (SRX_CHNL_EN) bit for the corresponding channel.

NOTE

The value of the CH is device-specific. See the device configuration section for details.

Address: 0h base + 64h offset + (16d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BUS_IDLE	0			CAL_RESYNC	CAL_20_25	SMSG_OFLW	FMSG_OFLW	0	PP_DIAG_ERR	CAL_LEN_ERR	CAL_DIAG_ERR	NIB_VAL_ERR	SMSG_CRC_ERR	FMSG_CRC_ERR	NUM_EDGES_ERR
W	w1c				w1c	w1c	w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



SRX_CH*n*_STATUS field descriptions

Field	Description
0 BUS_IDLE	Bus Idle Status. This bit indicates that the sensor interface has been idle for more than the period defined by the programmed Bus Idle Count value(in Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH*n*_CONFIG)). CPU should write 1 to this clear this bit once it is read. 0 Bus is not idle 1 Channel has been idle for more than the allowed value
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 CAL_RESYNC	Successive Calibration Check (Option 2 of SAE SENT Spec) Resynchronized. This bit indicates that Successive Calibration Check (Option 2 of SAE Spec) has failed three times and has resynchronized to assume 3rd message to be correct and will be received. The successive calibration check error will also be asserted during re-synchronization (to indicate the 3rd error being detected). 0 No interrupt 1 Interrupt Status condition has occurred
5 CAL_20_25	Calibration Variation 20 - 25% Interrupt Status. This bit indicates that the calibration pulse received on this channel has variation in between ±20% to ±25% from 56 ticks. 0 No Interrupt 1 Interrupt Status condition has occurred
6 SMSG_OFLW	Slow Serial Message Overflow Status. This bit indicates overflow occurred on this channel. Overflow will cause state machine to halt stopping reception of new messages. 0 No Interrupt 1 Interrupt Status condition has occurred
7 FMSG_OFLW	Fast Message Overflow Status. This bit indicates overflow occurred on this channel. Overflow happens when effective data read rate by CPU is slower than data receive rate on channel. On overflow, new incoming messages will overwrite the previous messages (that are not read out) stored in the receiver buffers. However, any message that is currently being read out by DMA or CPU will not get over-written. 0 No Interrupt 1 Interrupt Status condition has occurred

Table continues on the next page...

SRX_CH*n*_STATUS field descriptions (continued)

Field	Description
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 PP_DIAG_ERR	This diagnostic checks status bit indicates that the ratio of calibration pulse length to overall message length (with pause pulse) is more than ±1.5625% between two messages. This check is valid only for messages with pause pulse. This bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
10 CAL_LEN_ERR	This diagnostic checks status bit indicates that Calibration pulse is more than 56 ticks ±25%. This bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
11 CAL_DIAG_ERR	Successive Calibration pulses differ by ±1.56%. SAE spec defines it be 1.5625% but the accuracy of this check depends on the frequency of protocol clock (SeeHigh frequency receiver clock (protocol clock) requirements). This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
12 NIB_VAL_ERR	Any nibble data value <0 or >15. This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
13 SMSG_CRC_ERR	Checksum error in Slow Serial Message. This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
14 FMSG_CRC_ERR	Checksum error in Fast Message. This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel, and is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
15 NUM_EDGES_ERR	Not the expected number of negative edges between calibration pulse. This diagnostic checks status bit indicates whether this diagnostic has failed or passed on this channel. This bit will not set in case of "Option 2" for successive calibration pulse check is selected. This error will only set when "Option 1" for successive calibration check is selected (SUCC_CAL_CHK). This bit is cleared by writing a 1 to it. 0 Error Check has passed 1 Error Check has failed
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.19 Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH*n*_CONFIG)

This register (CHn_CONFIG) provides the configurability for interrupt assertion for various events and error as explained in Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH*n*_STATUS) and control bits for selection of various receive options that might be required by the user software. This register also configures the width of noise glitches (on the sensor input) to be filtered out by the input programmable filter.

User must program the contents of this register before enabling the channel by writing to the corresponding channel enable bit in the Channel Enable Register (SRX_CHNL_EN) .

NOTE

The value of the CH is device-specific. See the device configuration section for details.

Address: 0h base + 68h offset + (16d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BUS_IDLE_CNT				IE_CAL_RESYNC	IE_CAL_20_25	IE_SMSG_OFLW	IE_FMSG_OFLW	FCRC_CHK_OFF	IE_PP_DIAG_ERR	IE_CAL_LEN_ERR	IE_CAL_DIAG_ERR	IE_NIB_VAL_ERR	IE_SMSG_CRC_ERR	IE_FMSG_CRC_ERR	IE_NUM_EDGES_ERR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DCHNG_INT	CAL_RNG	PP_CHKSEL	FCRC_TYPE	FCRC_SC_EN	SCRC_TYPE	PAUSE_EN	SUCC_CAL_CHK	FIL_CNT							
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0

SRX_CH*n*_CONFIG field descriptions

Field	Description
0–3 BUS_IDLE_CNT	Bus Idle Count. This register value defines the maximum allowable idle period on the sensor interface of this channel. The value is defined as follows: 0000 Disabled. 0001 127*2 Receiver Clock Tick Counts 0010 127*4 Receiver Clock Tick Counts 0100 127*8 Receiver Clock Tick Counts 1000 127*16 Receiver Clock Tick Counts

Table continues on the next page...

SRX_CH*n*_CONFIG field descriptions (continued)

Field	Description
4 IE_CAL_ RESYNC	Successive Calibration Check Resynchronized Interrupt Enable: This bit enables interrupt assertion when Successive Calibration diagnosis has failed three times in case of "Option 2" being selected for Successive Calibration Check Method and the check resynchronizes the take the third message to be correct. 0 Interrupt is disabled 1 Interrupt is enabled
5 IE_CAL_20_25	Calibration Variation 20 - 25% Interrupt Enable. This bit enables interrupt assertion when corresponding status bit is set. 0 Interrupt is disabled 1 Interrupt is enabled
6 IE_SMSG_OFLW	Slow Serial Message Overflow Interrupt Enable. This bit enables interrupt assertion when overflow occurs on reception of Slow Serial Messages in corresponding channel 0 Interrupt is disabled 1 Interrupt is enabled
7 IE_FMSG_OFLW	Fast Message Overflow Interrupt Enable. This bit enables interrupt assertion when overflow occurs on reception of Fast Messages in corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
8 FCRC_CHK_ OFF	Fast Message CRC Check Off: This bit can be used to switch off CRC check in Fast Message 0 Check is enabled 1 Check is disabled/off
9 IE_PP_DIAG_ ERR	Ratio of calibration pulse length to message length varies by more than ±1.5625% between two frames. Valid for messages with pause pulse. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
10 IE_CAL_LEN_ ERR	Calibration pulse is wider than 56 ticks ±25%. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
11 IE_CAL_DIAG_ ERR	Successive Calibration pulses differ by more than ±1.56%. SAE spec defines it be 1.5625% but the accuracy of this check depends on the protocol clock. For instance for 62.5 MHz, this check will fail for 1.56% and passes for 1.55%. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
12 IE_NIB_VAL_ ERR	Any nibble data value <0 or >15. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled

Table continues on the next page...

SRX_CH*n*_CONFIG field descriptions (continued)

Field	Description
13 IE_SMSG_CRC_ERR	Checksum error in Slow Serial Message. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
14 IE_FMSG_CRC_ERR	Checksum error in Fast Message. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
15 IE_NUM_EDGES_ERR	Not the expected number of negative edges between calibration pulse. This bit enables interrupt assertion when its respective diagnostic check fails on the corresponding channel. 0 Interrupt is disabled 1 Interrupt is enabled
16 DCHNG_INT	Enable for Interrupt on Reception of Fast Message with Changed Data: This bit is used to enable storing and generating corresponding DMA or Message Ready Interrupt Request for Fast Message only when any of the data nibbles after different from previously received Fast Message on that channel. If a new Fast Message is received that has same values for all data nibbles as that of previous message then the current message will be discarded and will not be stored in the buffer, and neither Fast Message Ready Interrupt nor DMA request will be generated. 0 All Fast Messages will be received even if data nibbles are same 1 Only Fast Messages with differing values of data nibbles will be received
17 CAL_RNG	Valid Calibration Pulse Range Selection: This bit is used to control whether 20% variation or 25% variation in Calibration Pulse will be tolerated when detecting a Calibration Pulse 0 20% variation is acceptable 1 25% variation is acceptable
18 PP_CHKSEL	Pause Pulse Diagnostic Check Selection: This bit controls which diagnostic checks to run when messages are enabled to be received with pause pulses. 0 Both Successive Calibration Pulse Check and Pause Pulse Diagnostic are run 1 Only Pause Pulse Diagnostic is run. Successive Calibration Pulse Check is disabled.
19 FCRC_TYPE	Fast Message CRC Type. This indicates the type of CRC method to be used for CRC Diagnostic Check on the received fast messages 0 Recommended implementation (additional 0 data nibble XORed with the rest of the nibbles) 1 Legacy implementation (no additional 0 data nibble is XORed)
20 FCRC_SC_EN	Fast Message CRC Status and Communication Nibble Enable. This bit enables the inclusion of Status and Communication Nibble when CRC is calculated for Fast Messages. 0 Status and Communication Nibble not included in CRC calculation 1 Status and Communication Nibble is included in CRC calculation
21 SCRC_TYPE	Slow Serial Message CRC Type. This indicates the type of CRC method to be used for CRC Diagnostic Check on the received slow serial messages. 0 Recommended implementation (additional 0 data nibble XORed with the rest of the nibbles) 1 Legacy implementation (no additional 0 data nibble XORed)
22 PAUSE_EN	Pause Pulse Enable. Enables the channel receiver to detect a pause pulse.

Table continues on the next page...

SRX_CH*n*_CONFIG field descriptions (continued)

Field	Description
	0 Detection of Pause Pulse is disabled 1 Detection of Pause Pulse enabled
23 SUCC_CAL_CHK	Successive Calibration Pulse Check Method. This bit indicates the method to be used for performing the successive calibration pulse check. Default value is 1. 0 Option 2 i.e. Low Latency Option as per SAE Specification 1 Option 1 i.e. Preferred but High Latency Option as per SAE Specification
24–31 FIL_CNT	Input Filter Sample Count. This indicates twice the number of Protocol Clock (of High Frequency Receiver Clock) cycles required for channel input from device's pad to remain stable before it is sampled as 0 or 1. This field defines the width of glitches to be filtered out by the input programmable filter on that channel. Only one bit should be set in this field and the output of filter will be offset by twice the same number of clocks plus additional 3 clocks due to synchronization. Default value is '4'. 0 No filtering. Input from device's pad is only synchronized 1 Non-Zero Filtering is enabled

53.4.20 Channel 'n' Fast Message Data Read Register (n=0 to (CH-1)) (SRX_CH*n*_FMSG_DATA)

Address offset: 0x0160 + 0x18 x n

This is register CHn_FMSG_DATA. The contents of this register are same as DMA Fast Message Data Read Register (SRX_DMA_FMSG_DATA) .

NOTE

The value of the CH is device-specific. See the device configuration section for details.

Address: 0h base + 160h offset + (24d × i), where i=0d to 5d

Bit	0–3	4–7	8–11	12–15	16–19	20–23	24–27	28–31
R	CHNUM	SCNIB	DNIB1	DNIB2	DNIB3	DNIB4	DNIB5	DNIB6
W								
Reset	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

SRX_CH*n*_FMSG_DATA field descriptions

Field	Description
0–3 CHNUM	Channel Number. Indicates the channel number of the received message being read by DMA. Valid values are 0 to 5 .
4–7 SCNIB	Status and Communication Nibble of message
8–11 DNIB1	Data Nibble 1. Always supported

Table continues on the next page...

SRX_CH*n*_FMSG_DATA field descriptions (continued)

Field	Description
12–15 DNIB2	Data Nibble 2. Should be ignored if number of data nibbles supported by channel is 1
16–19 DNIB3	Data Nibble 3. Should be ignored if number of data nibbles supported by channel is 2 or less
20–23 DNIB4	Data Nibble 4. Should be ignored if number of data nibbles supported by channel is 3 or less
24–27 DNIB5	Data Nibble 5. Should be ignored if number of data nibbles supported by channel is 4 or less
28–31 DNIB6	Data Nibble 6. Should be ignored if number of data nibbles supported by channel is 5 or less

53.4.21 Channel 'n' Fast Message CRC Read Register (n=0 to (CH-1)) (SRX_CH*n*_FMSG_CRC)

Address offset: 0x0164 + 0x18 x n

This is register CHn_FMSG_CRC. The contents of this register are same asDMA Fast Message CRC Read Register (SRX_DMA_FMSG_CRC) .

Address: 0h base + 164h offset + (24d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												CRC4b				0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CH*n*_FMSG_CRC field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 CRC4b	4-bit CRC value of the message.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.22 Channel 'n' Fast Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH*n*_FMSG_TS)

Address offset: 0x0168 + 0x18 x n

This is register CHn_FMSG_TS. The contents of this register are same as DMA Fast Message Time Stamp Read Register (SRX_DMA_FMSG_TS).

Address: 0h base + 168h offset + (24d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CH*n*_FMSG_TS field descriptions

Field	Description
0–31 TS	Time Stamp for received message. Indicates the relative time when the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) was detected

53.4.23 Channel 'n' Serial Message Read Register (Bit 3) (n=0 to (CH-1)) (SRX_CH*n*_SMSG_BIT3)

Address offset: 0x016C + 0x18 x n

This is register CHn_SMSG_BIT3. The contents of this register are same asDMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3) .

NOTE

The value of the CH is device-specific. See the device configuration section for details.

Address: 0h base + 16Ch offset + (24d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CHNUM				TYPE	0										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					CFG	ID7_4_ID3_0				0	ID3_0_DATA15_12				0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CH*n*_SMSG_BIT3 field descriptions

Field	Description
0–3 CHNUM	Channel Number. Indicates the channel number of the received message being read by DMA. Valid values are 0 to 5 .
4 TYPE	Serial Message Type. Indicates the type of Serial Message received 0 Short Serial Message 1 Enhanced Serial Message

Table continues on the next page...

SRX_CH*n*_SMSG_BIT3 field descriptions (continued)

Field	Description
5–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 CFG	Configuration bit 'C'. Indicates the type of Enhanced Serial message 0 Enhanced Serial Message with 8-bit ID field 1 Enhanced Serial Message with 4-bit ID field
22–25 ID7_4_ID3_0	ID Field. Value depends on setting of 'C' bit. If 'C' bit is 0, this field is ID[7:4] and if the 'C' bit is 1, this field is ID[3:0]
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–30 ID3_0_DATA15_12	ID or Data Field. Value depends on setting of 'C' bit. If 'C' bit is 0, this field is ID[3:0] and if the 'C' bit is 1, this field is Data[15:12]
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

53.4.24 Channel 'n' Serial Message Read Register (Bit 2)(n=0 to (CH-1)) (SRX_CH*n*_SMSG_BIT2)

Address offset: 0x0170 + 0x18 x n

This is register CHn_SMSG_BIT2. The contents of this register are same asDMA Slow Serial Message Bit2 Read Register (SRX_DMA_SMSG_BIT2) .

Address: 0h base + 170h offset + (24d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										SMCRC						0				DATA[11:0]											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CH*n*_SMSG_BIT2 field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 SMCRC	6-bit CRC value of the message. When TYPE bit in DMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3) is set to 0, this CRC is of 4-bits for Short Serial Messages and when TYPE bit is set to 1, this CRC is of 6-bits for Enhanced Serial Messages.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 DATA[11:0]	12-bit Data Value

53.4.25 Channel 'n' Serial Message Time Stamp Read Register (n=0 to (CH-1)) (SRX_CH*n*_SMSG_TS)

Address offset: 0x0174 + 0x18 x n

This is register CHn_SMSG_TS. The contents of this register are same as DMA Slow Serial Message Time Stamp Read Register (SRX_DMA_SMSG_TS).

Address: 0h base + 174h offset + (24d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRX_CH*n*_SMSG_TS field descriptions

Field	Description
0–31 TS	Time Stamp for the received message. Indicates the relative time of detection of the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) of the last Fast Message that completed the Serial Message. Time Stamp for Slow Serial Messages is not latched when the Slow Serial Message starts but when the Slow Serial Message is completely received.

53.5 Functional description

This section describes the function of the SRX.

53.5.1 Initialization sequence

All the channels are required to be configured before enabling them by writing into corresponding channel enable bit in the Channel Enable Register (SRX_CHNL_EN). There are some registers which must be configured and some which are to be changed if reset value is not as required for that channel. After enabling the channel, no configuration should be changed and if required to be changed, the user needs to first disable the channel and then reconfigure the channel parameters. The following must to be configured before enabling the corresponding channel:

- Time Stamp Prescaler Value in Global Control Register (SRX_GBL_CTRL) to generate the time stamping clock

- Rx Prescaler Value in Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH*n*_CLK_CTRL) togenerate the channel's receiver clock
- Compensation enable in Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH*n*_CLK_CTRL)
- Channel Configuration settings in the Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH*n*_CONFIG)
- Number of Data Nibbles in Channel in the Data Control Register 1 (SRX_DATA_CTRL1)
- Control bits that control whether messages received in channel are to read by interrupt or DMA in Fast Message DMA Control Register (SRX_FDMA_CTRL), Slow Serial Message DMA Control Register (SRX_SDMA_CTRL), Fast Message Ready Interrupt Control Register (SRX_FRDY_IE) and Slow Serial Message DMA Control Register (SRX_SDMA_CTRL)

53.5.2 DMA read logic

The SENT Receiver module generates separate DMA requests for all Fast Messages and Slow Serial Messages received. The DMA requests are common for all channels. Each packet stored in the channels' buffers are appended with a time stamp taken from a free running time keeping counter and the channel number on which it was received. Details on this time stamp counter can be found in Time stamp logic.

53.5.2.1 DMA request for Fast Message reading

Fast Messages received are stored in a single system bus clock domain buffer to be read out by the DMA Controller via the DMA Fast Message Read register set. Note the DMA Controller must read all 3 DMA Read Registers to obtain the complete message. The data flow will happen as follows:

- A DMA request will be asserted when any of the DMA enabled channels has successfully received one complete Fast Message. The request is kept asserted till all messages (across DMA enabled channels) have been read out.
- DMA Controller should read the complete message by accessing each register of the DMA Fast Message Read register set, the registers being:
  - DMA Fast Message Data Read Register (SRX_DMA_FMSG_DATA)

  - DMA Fast Message CRC Read Register (SRX_DMA_FMSG_CRC)
  - DMA Fast Message Time Stamp Read Register (SRX_DMA_FMSG_TS)

- A fixed round robin sequencing runs, as shown in Figure 53-81, and this logic checks for messages in sequence 0→1→3 →5→7 and back to channel 0 and so on. Channel 2 is skipped as it is not DMA enabled) and channel 4 and 6 are skipped as message is not received completely. This sequence continues like this.
- The sequence in which messages are read through DMA interface is not deterministic and depends on the sequence in which they are completely received.
- When the sequencing logic detects a message ready to be read, it will update the DMA Fast Message Read register set for DMA read and move to the next channel when this message is read out completely (i.e., 3 read accesses to the above mentioned registers)
- The DMA controller should be programmed to read only one message in one DMA transfer (i.e. 12 bytes). If it is programmed to read more than one message in one DMA transfer, underflow in buffers might occur
- User software should ensure there is no overflow in channels. In case overflow occurs, the message in the channel buffer will get overwritten by the newly received message. However, when a DMA read is in progress, the buffers will not be overwritten if overflow occurs to maintain data integrity.
- The Round Robin sequencing logic loops through each DMA enabled channel and halts at any channel having data till it is read out by the DMA Controller.

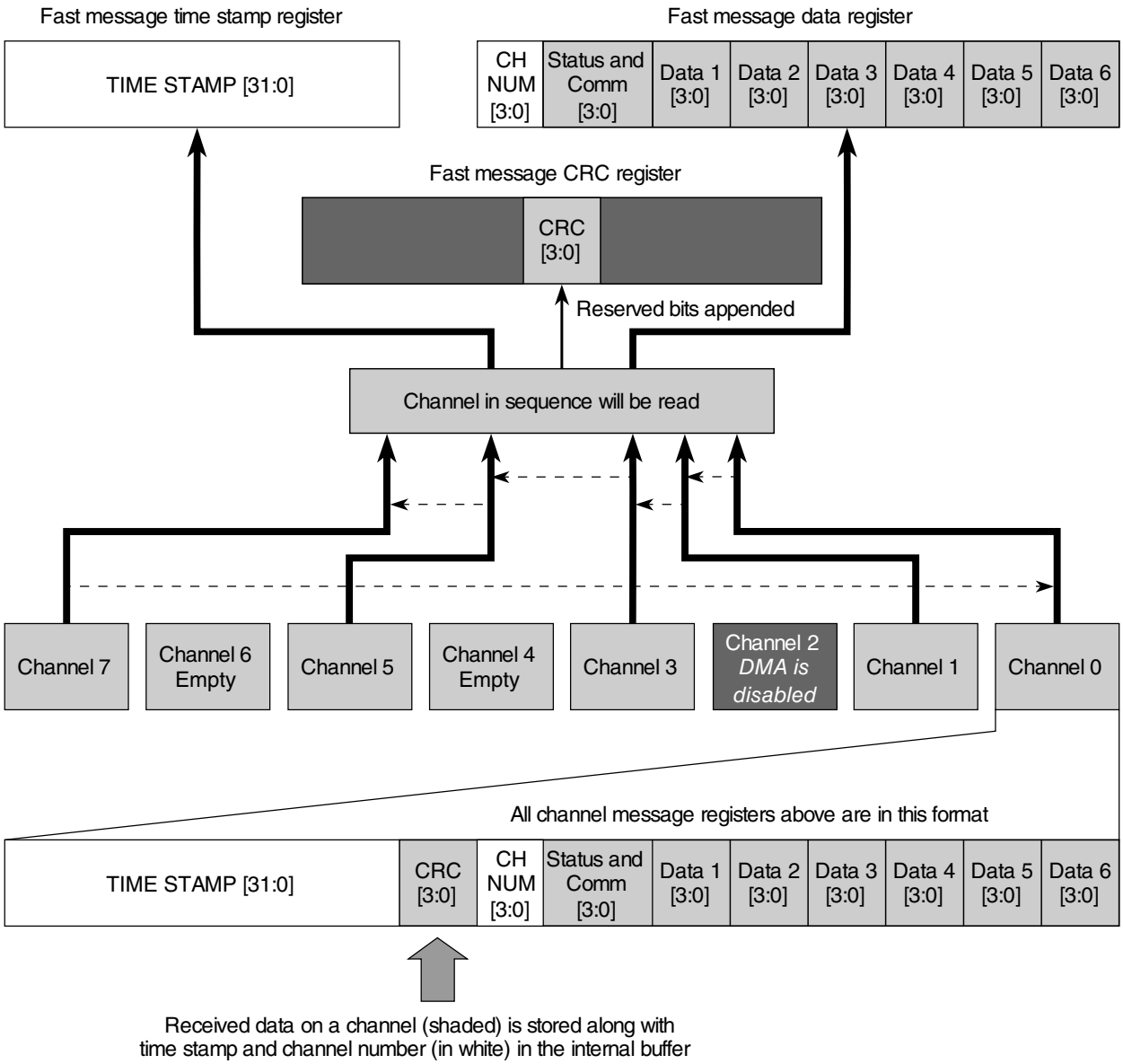


Figure 53-81. Fast Message DMA read logic

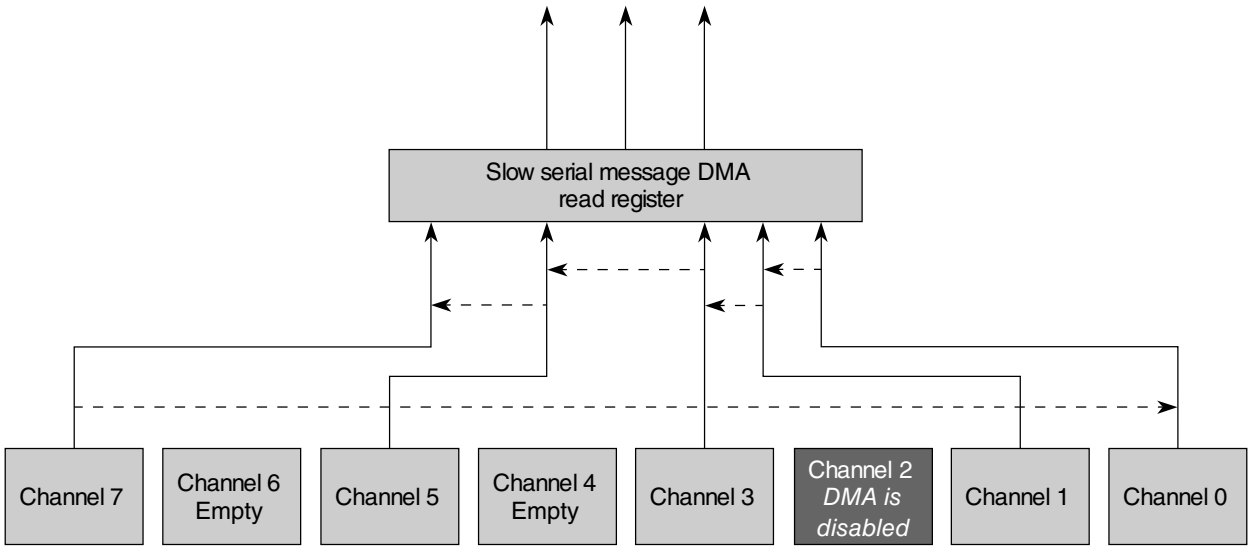
53.5.2.2 DMA request for Slow Serial Message reading

A separate DMA request (different from DMA request for Fast Message) is generated for all slow serial messages received on all channels. The data flow will happen as follows:

- Request will be asserted when any of the DMA enabled channels has successfully received one complete serial message (short or enhanced) and kept asserted till all messages have been read from their respective memory-mapped registers

- Similar to DMA read for Fast Messages, data for slow serial messages will be read from DMA Slow Serial Message Read register set in memory map, the register set comprising of:
  - DMA Slow Serial Message Bit3 Read Register (SRX_DMA_SMSG_BIT3)
  - DMA Slow Serial Message Bit2 Read Register (SRX_DMA_SMSG_BIT2)
  - DMA Slow Serial Message Time Stamp Read Register (SRX_DMA_SMSG_TS)
- A fixed round robin sequencing logic will be running and it will update the DMA Slow Serial Message Read register set for DMA read and move to the next channel when this message is read out completely (i.e., 3 DMA read accesses to the above mentioned registers)
- The channel number and time stamp will also be appended for every message
- A message type bit (TYPE) will also be appended to distinguish between short and enhanced serial messages
- The DMA controller should be programmed to read only one message in one DMA transfer (i.e. 12 bytes). If it is programmed to read more than one message in one DMA transfer, underflow in buffers might occur
- Since serial messages receive rate is very slow, user software should ensure there is no overflow
- The fixed round robin sequence logic for DMA read is the same as that used for Fast Messages (see DMA request for Fast Message reading)

Figure 53-82 shows the sequencing of channels for serial messages to be read when DMA read is done on DMA Slow Serial Message Read register set.



Note: The arrows in the figure indicate next DMA transfer. One DMA Read Register set (3 registers) will be read in

Figure 53-82. Slow Serial Message DMA read logic

The formats in which the 3 types of Serial Messages will be stored in the buffers are shown in figures below.

Table 53-83. Enhanced Serial Message with 8-bit ID field

31	1		16	15	0						0
CH NUM [3:0]	P				0	C	ID [7:4]	0	ID [3:0]		0
		CRC [5:0]			Data Field [11:0]						
TIME STAMP [31:0]											

Table 53-84. Enhanced Serial Message with 4-bit ID field

31	1		16	15	1						0
CH NUM [3:0]	P				0	C	ID [3:0]	0	Data Field [15:12]		0
		CRC [5:0]			Data Field [11:0]						
TIME STAMP [31:0]											

Table 53-83 and Table 53-84 shows the enhanced serial messages. Bit C (shown in figure) or CFG (as shown in DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3) description) defines the type of enhanced packet and bit P (shown in figure) or bit TYPE (shown in DMA Slow Serial Message Bit3 Read Register (DMA_SMSG_BIT3) description) defines whether it is short serial or enhanced serial message. CRC, ID and Data fields are stored in format as received and shown in SAE specification with just one exception that CRC is shifted to align with 16-bit boundary.

Table 53-85 shows the short serial message. Field locations have been placed in such a way to allow a common register to read all the slow serial messages and also match the bit positions in actual message defined by SAE Specification.

Table 53-85. Short serial message

31	0	16	15	0
CH NUM [3:0]	P			
		CRC [3:0]	ID [3:0]	Data Field [7:0]
TIME STAMP [31:0]				

Again, three 32-bit accesses to be made by DMA to get one complete message with time stamp.

53.5.3 Message reading via interrupts

There is one interrupt request per channel for a Fast message received and one for a serial message received. Each channel is individually configurable to enable or disable interrupt on reception of Fast or Slow Serial messages separately. In case of interrupt request of Fast Message, an interrupt will be asserted when any of the channels has received a complete Fast Message without errors and that channel is not configured for DMA Reads. The interrupt routine should be coded in such a way to take the least amount of overhead (in CPU cycles) while entering the routine and start reading the available messages. The routine should be capable of accessing all channels that are enabled to give out messages via interrupts through the Register Interface.

In response to these interrupts, the CPU is expected to read interrupt status register for fast or Slow Messages (depending on which interrupt was asserted) and store it in system memory. Using this status register value, the interrupt routine can decide from which channels the message is to be read. For instance if bit corresponding to channel 0 is set then routine should read from the address location corresponding to Channel 0 Fast Message Read registers and move onto the next asserted bit. The interrupt routine should scan through all the bits of this stored interrupt status register value and read corresponding channel's fast or Slow Message registers. In this manner all available messages can be read. After scanning through all the bits of status register stored in system memory, routine can read again the message ready interrupt status register to find out if any other channel has received in the mean time. This will save another interrupt routine time mainly from stacking/unstacking delay.

It is assumed that the interrupt controller will be able to respond fast enough to avoid any overflow as only one message per channel will be stored and the system bus clock frequency is sufficient to avoid this.

53.5.3.1 Suggested software interrupt service routine steps

These are suggested step to have an interrupt routine that can read all messages quickly and avoid any underflow/overflow. Actual implementations can vary.

When separate Fast Message and Slow Serial Message Interrupts are used:

The Fast Message Interrupt should have higher priority than Slow Serial Message Interrupt. The suggested ISR steps for a Fast Message Interrupts can be

- Read the Fast Message Ready Register and store it in a variable
- Shift out bits starting from MSB from this variable
- If bit is 1, perform read operation on corresponding channel's memory-mapped Fast Message Read Register set (3 registers). Three read accesses will fetch complete message. Shift out new bit. Repeat Step.
- Else if bit is 0, shift out new bit
- When all bits are shifted out, read the Fast Message Status Registers again to ensure no new message has arrived during the ISR.
- If there are no bits set, exit the ISR, else repeat this process.

The ISR steps for Slow Serial Message Interrupt can be same as for Fast Message ISR.

When common Fast Message and Slow Serial Message Interrupts are used:

The suggested ISR steps can be

- Read the Fast Message Status Register and store it in a variable
- Shift out bits starting from MSB from this variable
- If bit is 1, perform read operation on corresponding channel's memory-mapped Fast Message Read Register. Three read accesses will fetch complete message. Shift out new bit. Repeat Step.
- Else if bit is 0, shift out new bit
- When all bits are shifted out, read the Slow Serial Message Status Register into a new variable

- Shift out bits starting from MSB from this new variable
- If bit is 1, perform read operation on corresponding channel's memory-mapped Slow Serial Message Read Register. Three read accesses will fetch complete message. Shift out new bit. Repeat Step.
- Else if bit is 0, shift out new bit
- When all bits are shifted out, software can again check both Fast and Slow Serial Message Ready register to ready any message that might have been received while the routine was being serviced, or can choose to exit the routine.

53.5.4 Overflow behavior

This section describes the overflow behavior.

53.5.4.1 Fast messages buffers

Overflow in Fast Message buffers can occur when the CPU or DMA does not read messages from that channel for a long period of time. The receiver channel stores two messages in its buffers and indicates an overflow when these buffers are full.

When an overflow event occurs, the newly received message overwrite the previously stored message and the overflow status bit is set. However, if overflow asserts while CPU/DMA is reading that channel (i.e. all 3 read registers have not yet been read), then that message is not overwritten. While messages are being overwritten, the slow message reception is not hampered.

NOTE

In the case of a fast message overflow and continuous reading of register SRX_CHn_FMSG_DATA, SRX_CHn_FMSG_CRC, and SRX_CHn_FMSG_TS without clearing the SRX_FMSG_RDY[F_RDYn] bit there is a possibility that one message will be lost. Additionally, if the pause pulse feature is enabled the module asserts up to two NUM_EDGES_ERR in the status register (SRX_CHn_STATUS). In this case up to two frames can be lost. Note that some debuggers perform a continuous read of memory which can cause this issue to occur.

In order to avoid this software should ensure that fast message overflow does not occur. If interrupts are used to read the SENT messages the interrupt for data reception should be enabled for every channel n (SRX_FRDY_IE[FRDY_IEn]=1) and the interrupt priority should be such that the software is able to read the message before the next message arrives. When using Direct Memory Accesses (eDMA) to access the messages the DMA request from the SENT module should be serviced before the next message arrives. The minimum duration between the reception of two consecutive messages in one channel is 92 times the utick length (time).

Else ensure that the SRX_CHn_FMSG_DATA, SRX_CHn_FMSG_CRC, and SRX_CHn_FMSG_TS registers are not read continuously either in the software code or as a result of a debugger being connected. They should be read once per message and the FMSG_RDY[F_RDYn] bit should be cleared after the reads.

53.5.4.2 Slow message buffers

Overflow in Slow Message buffers can occur when the CPU or DMA does not read messages from that channel for a long period of time. The receiver channel stores two messages in its buffers and indicates an overflow when these buffers are full.

When an overflow event occurs, further reception of slow messages is halted until the buffers are read out by CPU or DMA. Since it is less likely that CPU or DMA do not read the read registers for 32 or 36 fast message length duration, an overflow in slow messages would not be as frequent as in fast messages.

53.5.5 Adjustment for variation in sensor (Tx) clock

The clock used by the Channel Receiver (referred to as Receiver Clock, or Rx_CLK) to sample the message nibbles is generated from a High Frequency Receiver Clock or Protocol Clock. This high frequency receiver clock is divided using a prescaler counter which generates a tick when it rollovers on reaching a count equal to the Prescaler Value programmed in the channel's clock control register. The receive clock is generated separately for each channel. By varying this Prescaler Factor, we can control the period of the receive clock and make it close to the Transmitter clock used by the sensor for the particular channel.

After reset and once channel is enabled, the compensated prescaler value is set to Zero and the user software programs the Prescaler Value to obtain the desired receiver clock frequency for the particular channel. Thus, the receiver starts working by assuming that the transmitter is sending messages on the frequency that is programmed in its channel clock control register. The channel's input after filtering is continuously measured by the Prescaler Counter. The compensation logic checks each pulse length and determines the calibration pulse from them. When a calibration pulse is detected, the compensated prescaler value is computed and stored in the Channel's Clock Control Register (Channel 'n' Clock Control Register (n = 0 to (CH-1)) (SRX_CH*n*_CLK_CTRL)). Software can use this value to determine the variation in Tx Clock.

Note

A pause pulse may incidentally be detected as a synchronization/calibration pulse during start up after reset. The correction value would be computed on the pause pulse too but since the next synchronization pulse would follow, the logic would again detect the falling edge on the synchronization pulse and the correction value would now be correctly computed on the correct synchronization pulse.

The Clock Compensation or Correction happens for every messages that is received and the compensated prescaler value remains constant for that message.

53.5.5.1 Adjustment for nibble length variation

Section 6.2 of SAE Specifications specifies the maximum allowable limits for clock drift and jitter for the transmitter and receiver separately. This can cause the length of nibbles to vary and can cause incorrect sampling of data nibbles. This variation is adjusted by the SENT Receiver when nibbles are being sampled.

53.5.6 Input programmable filter

The Input Programmable Filter ensures that only valid input pin transitions are received by the SENT Receiver module. The input programmable filter is an 8-bit programmable up counter that increments on the high frequency receiver clock. The sensor input signal from device's pad is synchronized by high frequency receiver clock. When a state change occurs in the sensor input signal, the 8-bit counter resets and starts counting up on the high frequency receiver clock. As long as the new state is stable on the pin, the counter remains incrementing. If a counter overflows occurs, the new pin value (i.e. sensor input) is validated and latched in the filtered output flop (also running on high frequency

receiver clock). If the opposite edge appears on the sensor input before validation (i.e. overflow), the counter is reset and the input is not passed onto the filtered output. At the next pin transition, the counter starts counting again. Any pulse that is shorter than a full range of the masked counter is regarded as a glitch and it is not passed on to the filter output. The number of samples (or the match value of the up counter) is programmable by the user in the channel's configuration register Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH*n*_CONFIG). When zero is programmed into the registers, it bypasses the filter and the synchronized input is output as the filtered output. A non-zero value delays the filtered output by the number of clocks proportional to the programmed value in the Channel Configuration Register (Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH*n*_CONFIG)). A timing diagram of the input filter when FIL_CNT is 4, is shown in the following figure.

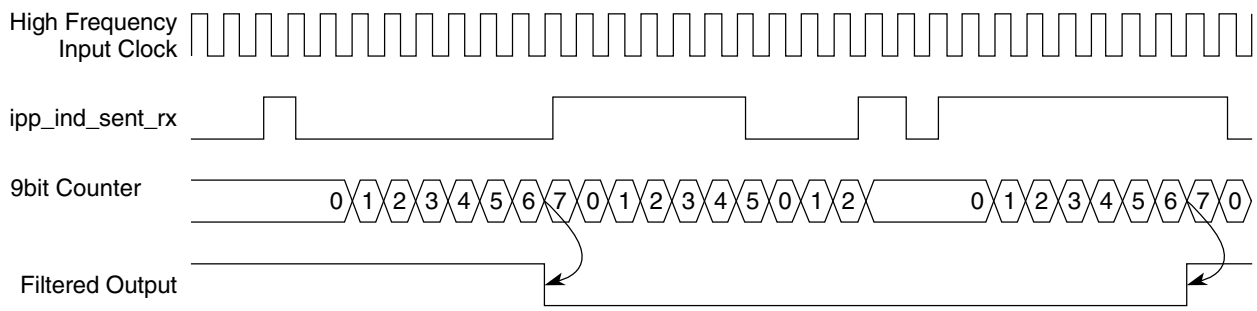


Figure 53-83. Input programmable filter timing diagram

53.5.7 Receiver diagnostics

Most of the receiver diagnostics are controlled by the Fast Message State Machine. The following checks are conducted by the receiver in each channel

- Calibration pulse length < 56 clock ticks – 25% or > 56 clock ticks + 25%
- Not the expected number of falling edges between calibration pulses. (Message length is pre-defined by each sensor device and programmed by user for each channel)
- Checksum error. Two 4-bit CRC checks and one 6-bit CRC check according to the message type. User has a programmable option to select the method of CRC to use i.e. Legacy or Recommended.
- Any nibble data values measured as < 0 or > 15

- Successive calibration pulses differ by > +1.5625% (1/64) or < -1.5625%. The accuracy of this check will depend on the frequency of the protocol clock used.
- For messages with pause pulse, an additional diagnostic is done; however, this diagnostic will not fail any message (to reduce latency in message reception) but the failure of this diagnostic will be indicated in the channel's status register.

On detection of any of the above errors, the receiver rejects the current packet until a new calibration pulse is detected. Occurrence of an error is flagged in the respective channel's status register (Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH*n*_STATUS)). If error interrupt is enabled, an error interrupt will be generated. There is a single error interrupt generated for all errors, so the CPU must read the status register to find out which error occurred. Once error condition is flagged, subsequent errors (except NUM_EDGES_ERR) are automatically masked out until a valid calibration pulse is detected, after which all checks are automatically enabled. This is done to avoid multiple assertions of the interrupt to CPU in case the channel become unstable and send wrong data continuously.

53.5.7.1 Calibration pulse length check

The calibration pulse length check happens in conjunction with Clock Compensation Logic since current calibration pulse is detected in this block. The Clock Compensation Logic detects a calibration pulse of length between 42 and 70 tick counts. This check will assert the CAL_LEN_ERR status bit on failure.

53.5.7.2 Successive calibration pulse check

The successive calibration pulse check checks if successive calibration pulses differ by > +1.5625% (1/64) or < -1.5625%. The accuracy of this check will depend on the frequency of the protocol clock used. Specifically, the higher the frequency, the closer the check will be to 1.5625%. So if protocol clock is T_{HF_CLK} and channel tick period is T_{TX_CLK} then the check will be of 1.5625% ± {100 × (4 × T_{HF_CLK}) / (42 × T_{TX_CLK})}. So it won't allow variance in clock of a channel by 1.5625% - {100 × (4 × T_{HF_CLK}) / (42 × T_{TX_CLK})}. This check will assert the CAL_DIAG_ERR status bit on failure.

When Option 2 of successive calibration pulse check is enabled, the CAL_RESYNC bit will be asserted on every third successive error detected by this check.

53.5.7.3 Not the expected number of edges check

The number of negative edges between two successive calibration pulses is counted and compared with expected number of edges in case of preferred option of successive calibration pulse check. The expected number of nibbles is the sum of following:

- Status and Communication nibble (1 No.)
- Data nibbles (Programmable as per user software)
- CRC nibble (1 No.)
- Pause Pulse, if configured (1 No.)

This check ("Not the expected number of edges check") will assert the NUM_EDGES_ERR status bit on failure. The check does not run when Option 2 (low latency) of successive calibration pulse check is enabled. This check is not masked after any other error is detected so it could be redundant if it is flagged along with CAL_DIAG_ERR and CAL_LEN_ERR. However, this is true only when CPU is clearing error status bits as and when they are asserted. If errors are accumulated, the above cannot be deduced.

53.5.7.4 Nibble value check

As part of this check, the lengths of the following nibbles are checked to remain in between 0 and 15 nibble values(11.5 ticks to 27.5 ticks):

- Status and Communication nibble
- all Data nibbles
- CRC nibble

The nibble value check will assert the NIB_VAL_ERR status bit on failure. This error can be flagged along with NUM_EDGES_ERR in which case the former can be ignored as the number of pulses in the message is incorrect. However, this is true only when CPU is clearing error status bits as and when they are asserted. In case, errors are accumulated, the above cannot be deduced.

53.5.7.5 CRC check

The CRC check will compute both 4-bit and 6-bit CRC as requested by State Machine control logic. Two 4-bit CRCs and one 6-bit CRC are supported to compute the CRC on Fast/Slow Serial Message (4-bit CRC) and Enhanced Serial Message (6-bit CRC) respectively. The SENT Receiver Module allows the user to calculate the 4-bit CRC using either the legacy CRC implementation method or the new recommended CRC implementation via a control bit in the channel's corresponding control register. Please refer to SAE SENT Specification for details on CRC algorithm.

Asserting the CHn_CONFIG_REG[11] bit, causes the Status and Communication nibble to be included in the 4-bit CRC calculation for Fast Serial Messages. De-asserting this bit enables normal operation as described above.

This check will assert the FMSG_CRC_ERR or SMSG_CRC_ERR status bit on failure, depending on whether the check is run for Fast Message or Slow Message.

The FMSG_CRC_ERR can be flagged along with the NUM_EDGES_ERR, in which case the CRC error can be ignored as the number of pulses in the message was incorrect. However, this is true only when CPU is clearing error status bits as and when they are asserted. If errors are accumulated, the above cannot be deduced.

53.5.7.6 Pause pulse diagnostic

The pause pulse diagnostic checks whether the ratio of calibration pulse length to full message length (including pause pulse) varies from one message to another by less than 1.5625% (as per spec). In order to have reduced latency of operation, message reception is not gated due to this check and no messages are discarded. However, when this check fails, a status bit is asserted to let the software know of this condition. The successive calibration pulse check can be configured to operate along with this check. This configurability is added to avoid conflicting results from the two checks.

There can be some inaccuracy in measuring pause pulse length as SAE specification specifies maximum clock drift/jitter up to maximum nibble length only and the frequency of high frequency receiver clock (protocol clock) used, allows clock compensation logic to ensure accuracy up to maximum nibble length. Hence there could be some inaccuracy in this check when sampling pause pulses that are larger than the maximum nibble pulse length.

This check will assert the PP_DIAG_ERR status bit on failure. This check does not run when reception of pause pulse is disabled.

When the Single Edge Nibble Transmission (SENT) Receiver (SRX) is configured to receive a pause pulse (Channel ‘n' Configuration Register - CHn_CONFIG[PAUSE_EN] = 1) and the length of the pause pulse from the sensor is in range of the valid calibration pulse length, the NUM_EDGES error can get asserted spuriously (Channel ‘n' Status Register - CHn_STATUS(NUM_EDGES_ERR] = 1) when there is any diagnostic error (other than number of expected edges error) or overflow in the incoming messages from the sensor.

Software can distinguish a spurious NUM_EDGES_ERR error from a real one by monitoring other error bits. The following tables will help distinguish between a false and real assertion of NUM_EDGES_ERR error and other errors. Software should handle the first error detected as per application needs and other bits can be evaluated based on these tables. Table 1 contains information due to erratum behavior and Table 2 below contains clarification of normal NUM_EDGES_ERR behavior.

Table 53-86. Behavior of NUM_EDGES_ERR

First Error Detected	Other error bits asserted	Cause for extra error bits getting asserted	Action
NIB_VAL_ERR	NUM_EDGES_ERR asserted twice	Upon detection of the first error, the state machine goes into a state where it waits for a calibration pulse, the first NUM_EDGES_ERR error is for the current message as the state machine does not detect an end of message. The second error comes when both the Pause pulse and the Calibration pulse are seen as back to back calibration pulses and no edges in between.	Ignore both NUM_EDGES_ERR error
FMSG_CRC_ERR	NUM_EDGES_ERR asserted twice	Upon detection of the first error, the state machine goes into a state where it waits for a calibration pulse, the first NUM_EDGES_ERR error is for the current message as the state machine does not detect an end of message. The second error comes when both the Pause pulse and the Calibration pulse are seen as back to back calibration pulses and no edges in between.	Ignore both NUM_EDGES_ERR errors

Table continues on the next page...

Table 53-86. Behavior of NUM_EDGES_ERR (continued)

CAL_LEN_ERR	NUM_EDGES_ERR asserted once	Since the calibration pulse is not detected as a valid calibration pulse, the internal edges counter does not detect the end of one message and start of bad message (which has CAL_LEN_ERR); hence the NUM_EDGES_ERR gets asserted.	Ignore NUM_EDGES_ERR error

Table 53-87. Expected behavior, clarification of NUM_EDGES_ERR cases

First Error Detected	Other error bits asserted	Cause for extra error bits getting asserted	Action
NUM_EDGES_ERR (when edges are less than expected)	NIB_VAL_ERR is asserted	When the actual number of edges in the message are less than expected, then a pause pulse gets detected as a nibble since the state machine expects nibbles when actually there is a pause pulse present. This generates NIB_VAL_ERR.	Ignore the NIB_VAL_ERR
NUM_EDGES_ERR (when edges are more than expected)	NIB_VAL_ERR and PP_DIAG_ERR are asserted	When the actual number of edges in a message are more than expected, then after receiving the programmed number of data nibbles, the state machine expects a pause pulse. However, the pause pulse comes later and gets detected as a nibble and hence NIB_VAL_ERR is asserted. Since the message length is not correct, PP_DIAG_ERR is also asserted.	Ignore NIB_VAL_ERR and PP_DIAG_ERR

53.5.8 Time stamp logic

A single counter maintains the time stamp for all channels which can be used by the software to determine the relative time of arrival of the messages for each channel. The time stamp counter is a 32-bit counter.

The clock for this counter is derived from the high frequency receiver clock using a Time Stamp Prescaler Counter that divides the high frequency receiver clock to generate the clock of required resolution. The user must program the time stamp prescaler value (in

Global Control Register (SRX_GBL_CTRL)) in order to generate the time stamp clock. The period of the time stamp clock must be less than that of the fastest receiver channel being used, or it can be set at some fixed value determined by application, such as 1 µs. For example, if the high frequency receiver clock is 60 MHz, the required prescaler value is 59 to generate a time stamp clock of 1 µs resolution.

Note

The rollover of time stamp values should be determined in user software, by comparing the time stamp values in previous and current messages that are read out. User software should adjust the time stamp values in these messages accordingly, after reading them.

The time stamp for a Fast Message is sampled at the falling edge that comes at the end of the calibration pulse (which is also the start of the status and communication nibble pulse) and is stored in the message read buffer. However, if the current message is found to be in error, the message buffer is discarded. Thus, the time stamp will be available to user software for only those messages which do not have any error.

The time stamp for a serial message is sampled at the end of the message i.e. when the serial message is completely received and when the message is found to have no errors. the time stamp is stored in the message read buffer to be read by the user software. Again, for messages with error the message and time stamp are discarded.

53.5.8.1 Limitation

Since the same counter is used for all channels, it may happen that more than 1 message is received within the same time stamp across different channels. Hence the software would not be able to determine the sequence of arrival of messages which have the same time stamp value.

53.5.9 Bus Idle Diagnostic

This diagnostic checks the connected sensors for inactivity. SENT module can indicate if the idle period on a particular channel has crossed the period defined by the programmed value in Bus Idle Count (in Channel 'n' Configuration Register (n=0 to (CH-1)) (SRX_CH*n*_CONFIG)). Status bit (bus_idle in Channel 'n' Status Register (n=0 to (CH-1)) (SRX_CH*n*_STATUS)) will be set when idle period crosses the allowed value. CPU should write 1 to this clear this bit once it is read.

53.6 Clocks and resets

The SENT Receiver works on two clocks. First is the System Bus Clock that is used to program the registers and read messages via DMA or Interrupt from the module's Register Interface. Second is the High Frequency Receiver Clock or Protocol Clock that is used for accurate message receiving operations. There is a single asynchronous reset which is the system asynchronous reset and one internal reset for each of the channel logic.

53.6.1 Clocking strategy

The following figure shows the clock domain in which each module is functioning.

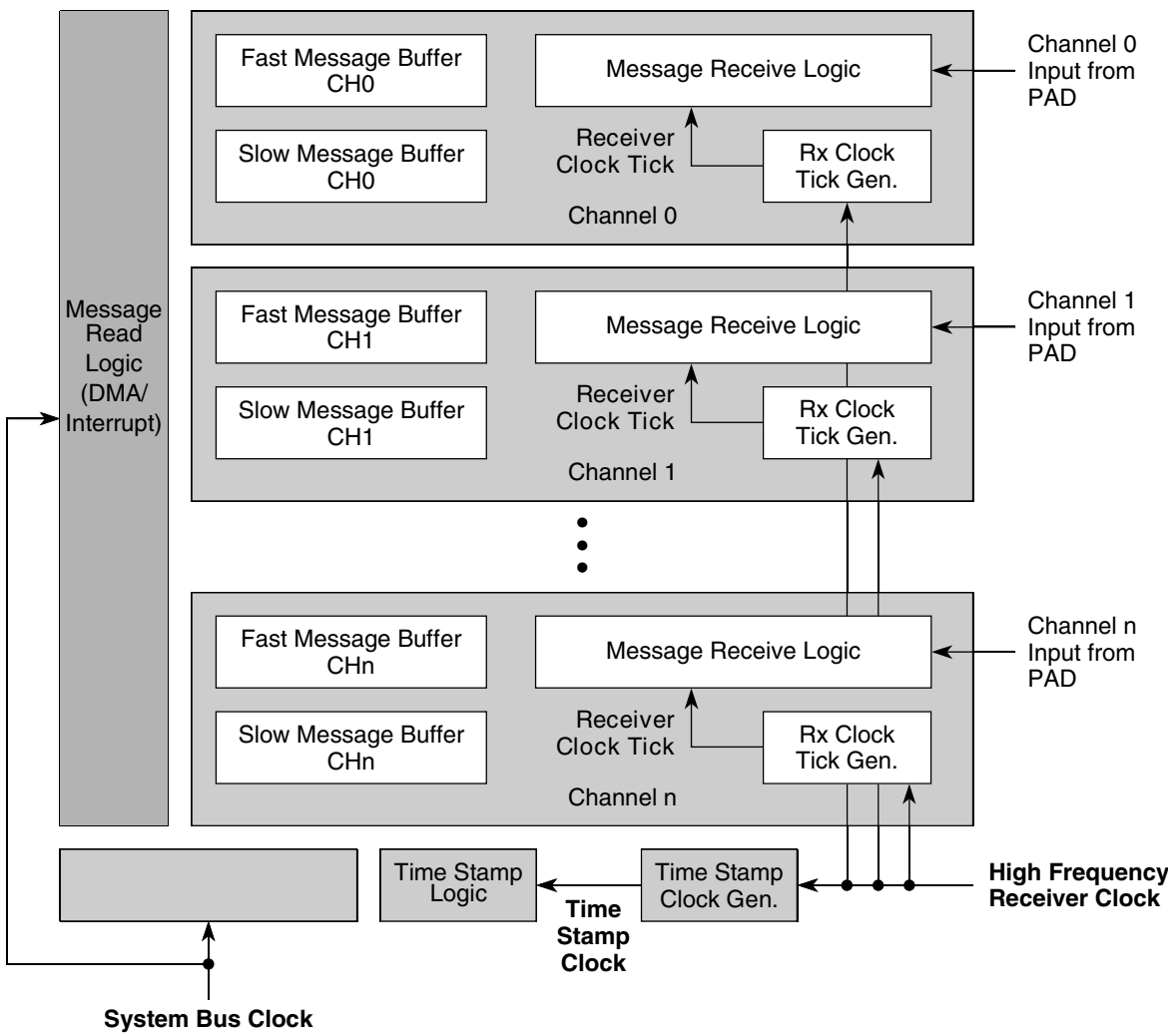


Figure 53-84. SENT Receiver module clock domains

53.6.2 System bus clock requirements

Since all the received messages are read on the System Bus Clock it is necessary that both the interrupt service routine and the DMA controller must be able to access all channels with messages ready to be read within a time to prevent overflow in any channel. The minimum time available to the CPU or DMA is the case when the smallest Fast Message is received back to back on all channels while working on the smallest receive clock tick period.

Now, the smallest Fast Message size with valid checksum as per SAE Specifications is 154 ticks. At 3 µs receive clock tick with a -20% variation in the transmitter clock tick, the effective receiver clock tick is 2.4 µs (after compensation). This gives us the time in which the minimum size Fast Message is received in any channel as 154 × 2.4 = 369.6 µs. Now since messages are arriving in parallel, 369.6 µs is the time available for CPU or DMA to access all channels to prevent an overflow in the channel that received the very first message.

Thus, if assume that 'N_{CHNL}' channels are being used and all channels have messages ready to be read together from either DMA or CPU then the time available to read a message in one channel will be 369.6/N_{CHNL} µs or ~370/N_{CHNL} µs. Since the CPU or the DMA needs to make three 32-bit accesses in order to read a complete message and taking the time required for one 32-bit read access as 'T_{RD}' (assuming it to be same for DMA and CPU); we can determine the minimum frequency for the System Bus Clock as:

For CPU,

$$3 \times T_{RD} + T_{ISR} + T_{WAKUP} < 370 / N_{CHNL}$$

where T_{ISR} is the overhead in initiating an ISR and T_{WAKUP} is the time to wakeup from low power mode

For DMA,

$$3 \times T_{RD} + T_{WAKUP} < 370 / N_{CHNL}$$

Replacing $T_{RD} = N_{RD} \times T_{SYS_CLK}$ and $T_{ISR} = N_{ISR} \times T_{SYS_CLK}$, where N_{RD} is the number of System Bus Clock cycles in one 32-bit read access, N_{ISR} is the number of System Bus Clock Cycles in initiating an interrupt service routine and T_{SYS_CLK} is the period of the same clock, we get

For CPU,

$$(3 \times N_{RD} + N_{ISR}) \times T_{SYS_CLK} + T_{WAKUP} < 370 / N_{CHNL}$$

(all units is µs)

For DMA,

$3 \times N_{RD} \times T_{SYS_CLK} + T_{WAKUP} < 370 / N_{CHNL}$

(all units is μs)

53.6.3 High frequency receiver clock (protocol clock) requirements

For accurate receive operations, the protocol is required to satisfy a minimum frequency defined as per formula: Guard Band > 2.5 * T_{HF_CLK}

where T_{HF_CLK} is the High Frequency Receiver Clock or Protocol Clock in ns and Guard Band is guard band in ns as specified in SAE specification, at particular transmitter uTick.

Note

T_{HF_CLK} in the above equation should be chosen based on the fastest sensor clock (with minimum guard band as specified in SAE spec) among all channels in the device.

Chapter 54
LINFlexD

54.1 LinFlexD chip-specific module information

The device contains 6 LinFlexD instances. The following table provides an overview of the modes supported by the instances.

Table 54-1. LINFlexD instances

LINFlexD instance	Master mode	Slave mode	LIN	Note
LINFLEX_0	Yes	Yes	Yes	LINFlex/UART interface compliant with LIN 1.3, LIN 2.0 and LIN 2.1 standards.
LINFLEX_1 , LINFLEX_2, LINFLEX_3	Yes	No	Yes	
LINFLEX_M0, LINFLEX_M1	No	Yes	No	For MSC operation the interface will support MSC upstream frames reception only.

NOTE

Only linflex_0 implements the IFCR register. All other LINFlexD instances do not implement the IFCR register.

NOTE

The LINFlex modules inplemented on MPC5746R do not support communication with Special Word Length in UART mode set (UARTCR[WLS]=1) in buffer mode or in FIFO mode with UARTCR[WL1, WL0]=0,1.

54.2 Introduction

The LINFlexD controller is designed to manage a high number of LIN messages efficiently with a minimum of CPU load. To reduce the CPU loading in Master mode, LINFlexD autonomously handles the LIN messages once software has triggered the header transmission, until the next header transmission request in transmitter mode or until checksum reception in the receiver mode.

The LINFlexD supports LIN protocol version 1.3, 2.0, and 2.1. It also consists of an 8-byte buffer for transmission/reception data.

The LINFlexD also provides support for some of the basic UART transfers of 8-bit, 9-bit, 16-bit, and 17-bit frames and also 12-bit data frame + parity reception in UART mode for MSC support.

The LINFlexD supports multi-channels and parametric DMA Tx/Rx interface in LIN/UART operating mode.

54.2.1 Glossary and acronyms

Table 54-2. Glossary and acronyms

Term	Description
LIN	Local interconnect network
UART	Universal asynchronous transmitter receiver
ID	LIN identifier field
DMA	Direct memory access
TCD	Transfer control descriptor
IPS	Peripheral Bus Interface
IRQ	Interrupt request
FSM	Finite state machine
WS	Wait state
SW	Software
CPU	Central processing unit
SoC	System on a chip

54.2.2 References

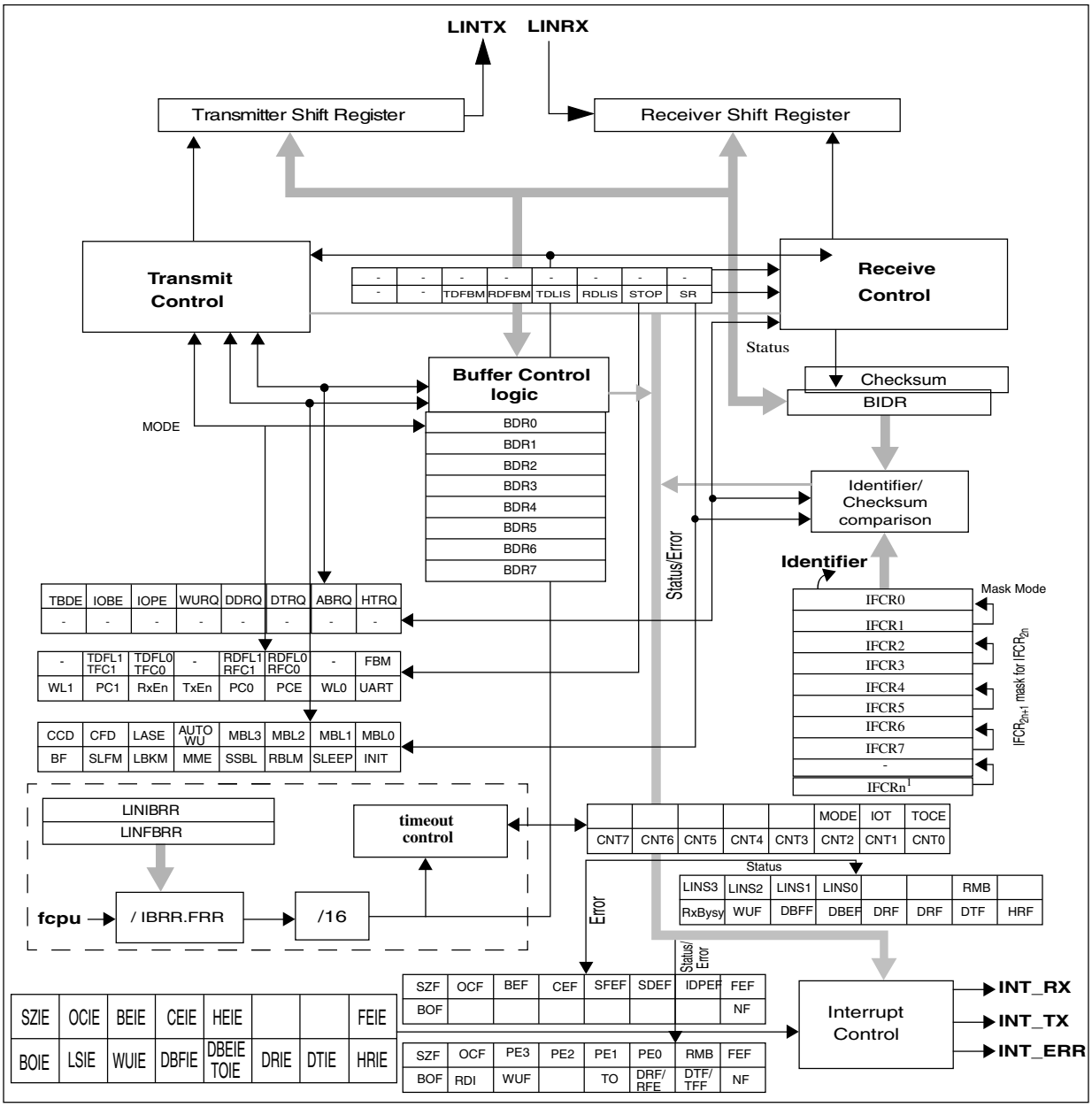


Figure 54-1. Block diagram

Notes:

1: The value of n depends on the no_of_filters. Refer to the chip configuration details to see the number of filters used in this device.

2: For LIN mode N = 16 and N is programmable by user in UART mode.

54.3 Main features

The LINFlexD controller can operate in several modes, each of which has a distinct set of features that are described in the following sections. In addition, the LINFlexD controller has several features common to all modes:

- Fractional baud rate generator
- 3 operating modes for power saving and configuration registers lock
  - Initialization
  - Normal
  - Sleep
- Test Mode:Loop Back
- Maskable interrupts
- A maximum of 16 possible identifiers can be programmed into the identifier list

54.3.1 LIN mode features

- Supports LIN protocol version 1.3, 2.0, and 2.1
- Bit rates up to 20 Kbit/s (LIN protocol)
- Master/Slave mode
- Classic and Enhanced Checksum calculation and check
- Single 8-byte buffer or FIFO for Transmission/Reception
- Timeout management
- Identifier filters
- DMA interface
- Supports a maximum of 16 possible identifiers
- Master mode with autonomous message handling
- Extended frame mode for in-application programming purposes
- Wakeup event on dominant bit detection

- True LIN field state machine
- Advanced LIN error detection
- Header, response, and frame timeout
- Slave mode
  - Autonomous header handling
  - Autonomous transmit/receive data handling
- Identifier filters for autonomous message handling in Slave mode
- Separate clock for baud rate calculation
  - The relationship “(2/3)* LIN_CLK > PBRIDGEx_CLK > 1/3*LIN_CLK” should be maintained.

54.3.2 UART mode features

- Full-duplex communication
- Baud rate is a function of baud clock, LINIBRR and LINFBRR registers - see Baud rate generation
- Separate clock for baud rate calculation
  - The relationship “(2/3)* LIN_CLK > PBRIDGEx_CLK > 1/3*LIN_CLK” should be maintained.
- 15/16/7/8 bits data, parity
- 1/2/3 stop bits
- 12-bit + parity reception
- 4-byte buffer for reception, 4-byte buffer for transmission
- 12-bit counter for timeout management
- The maximum baud rate achievable is 25 Mbit/s.
- For bit rate ≤ 6.25 Mbit/s
  - Sixteen times oversampling
  - 3:1 majority voting
- For 6.25 Mbit/s < bit rate ≤ 12.5 Mbit/s

- Reduced over sampling programmable by the user
- 3:1 majority voting for reduced over sampling of 8

- For 12.5 Mbit/s < bit rate ≤ 25 Mbit/s
  - Reduced over sampling programmable by the user
  - 1:1 voting for all reduced over sampling of 4, 5 and 6

54.4 Functional description

54.4.1 LIN protocol

The LIN (Local Interconnect Network) is a serial communication protocol. A LIN cluster consists of one master task and several slave tasks. A master node contains the master task as well as a slave task. All other nodes contain a slave task only. The master node decides when and which frame is transferred on the bus. The slave task provides the data to be transported by the frame.

54.4.1.1 Frames

A frame consists of a header provided by the master task and a response provided by the slave task. The header consists of a break, a sync pattern, and an identifier. The break is followed by the sync pattern and the sync pattern is followed by the identifier. The slave task associated with the identifier provides the response. The response consists of a data field and checksum. The slave task designated to receive the data associated with the identifier receives the response and verifies the checksum.

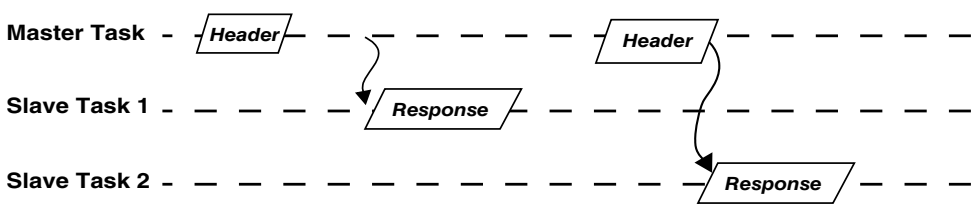


Figure 54-2. Frames

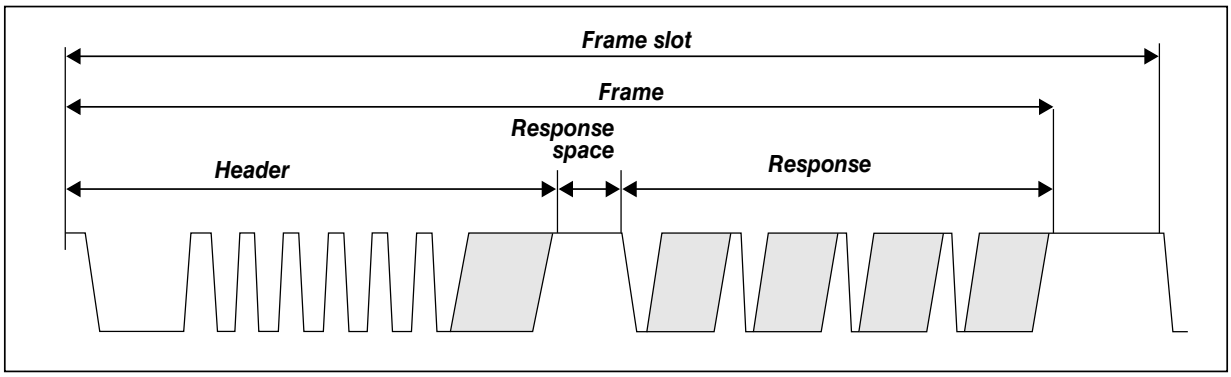


Figure 54-3. Structure of LIN frame

54.4.1.2 Data field

Each byte is transmitted as shown in Figure 54-4. The LSB of the data is sent first and the MSB is sent last. The start bit is encoded as a bit with value zero (dominant) and stop bit is encoded as bit value one (recessive).

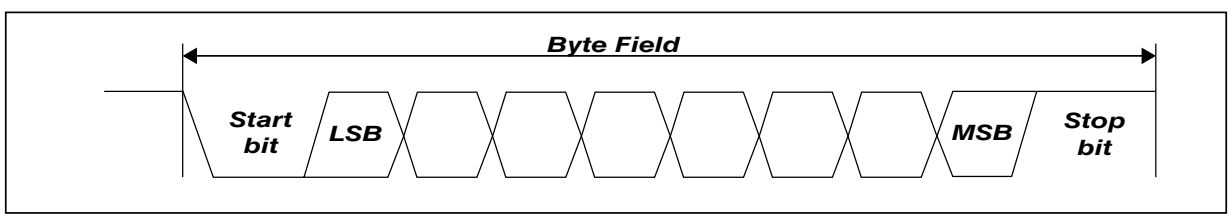


Figure 54-4. Structure of byte field

54.4.1.3 Break

The break symbol is used to signal the beginning of a new frame. It is the only field that does not comply with the above figure. The break is always generated by the master and shall be at least 13 bits of dominant value including the start bit, followed by a break delimiter as shown in Figure 54-5. The break delimiter must be of two-bit duration (to be compliant with LIN protocol 2.1).

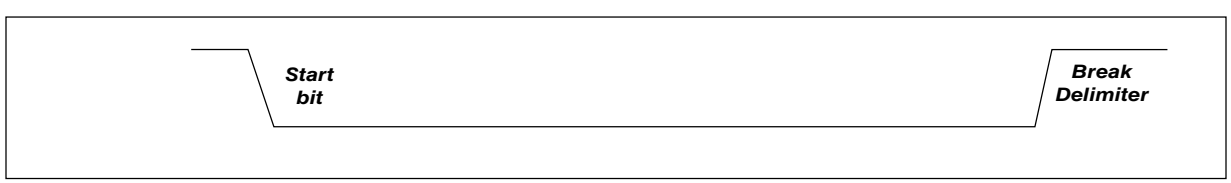


Figure 54-5. Break field

54.4.1.4 Sync byte

Sync is a byte field with the data value of 0x55.

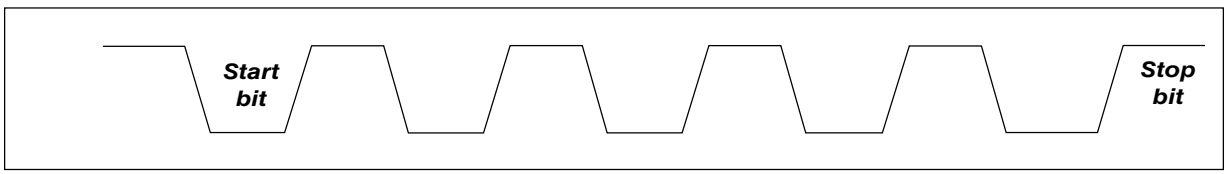


Figure 54-6. Sync byte field

54.4.1.5 Identifer

The Identifier field consists of two sub-fields, the identifier and the identifier parity. Bits 0 to 5 are the identifier and bits 6 and 7 indicate the parity.

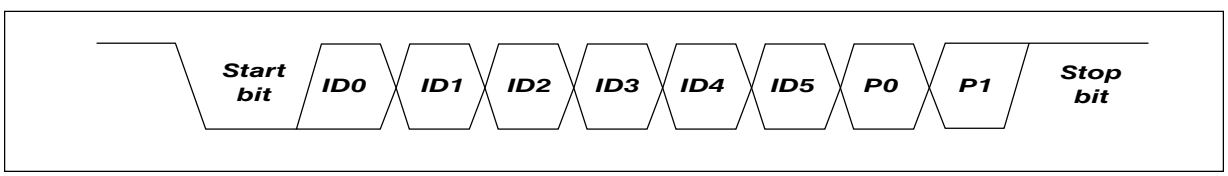


Figure 54-7. Identifier field

P0 = ID0 XOR ID1 XOR ID2 XOR ID4

P1 = NOT (ID1 XOR ID3 XOR ID4 XOR ID5)

54.4.1.6 Checksum

The last field of a frame is the checksum. The checksum contains the inverted 8-bit sum with carryover of all data bytes or all data bytes and the identifier. Checksum calculation over the data bytes only is called classic checksum and is used for communication with LIN 1.3 slaves. Checksum calculation over the data bytes and the protected identifier byte is called enhanced checksum and is used for communication with LIN 2.0 slaves.

54.4.2 LINFlexD features

54.4.2.1 Operating modes

The LINFlexD has three operating modes: **Initialization, Normal** and **Sleep** modes. After hardware reset, the LINFlexD enters sleep mode to reduce power consumption.

Initialization mode (INIT)

To enter this mode, software sets the INIT bit in the LINCR1. To exit the initialization mode, software should reset the INIT bit.

When in initialization mode, all message transfers to and from the LIN bus are stopped and the status of LIN bus output LINTX is recessive (high). If software invokes the initialization mode when a bus transfer is in progress, the transfer is aborted. The software should therefore check the LIN state before setting this bit.

To initialize the LINFlexD controller software must:

1. Set up the baud rate registers
2. Reset UART bit
3. Select the mode (master or slave)
4. Configure checksum control bits
5. Initialize the identifier list (Slave mode)

Normal mode (NM)

Once software has completed initialization of the LINFlexD controller, it can enter the Normal mode by clearing the INIT bit.

Sleep mode (SM)

Sleep mode in LINFlexD reduces power consumption. This mode is entered by setting the SLEEP bit in LINCR1. In this mode the LINFlexD clock is stopped. LINFlexD can be awakened from Sleep mode by clearing the SLEEP bit.

If software detects a wakeup pulse of 150 µs on the LIN Bus, it can request LINFlexD to wake up from Sleep mode. Refer to Wakeup management.

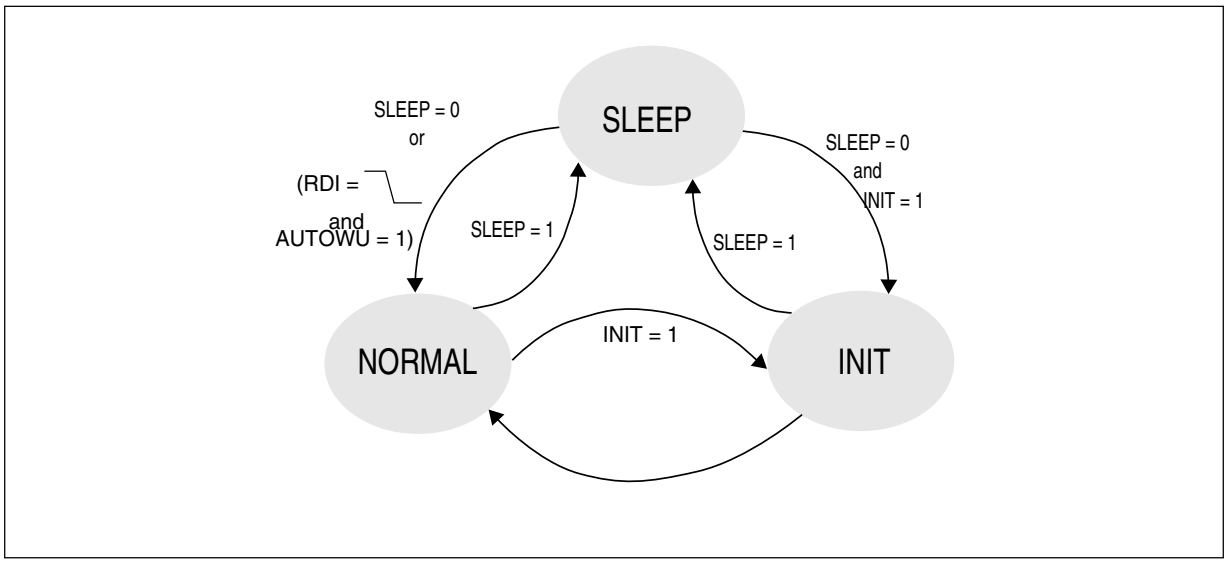


Figure 54-8. Operating modes

54.4.2.2 Test mode

Loop Back mode

This mode is entered by setting the LBKM bit in LINCR1. In this mode, the LINFlexD receives the Identifier and Data transmitted by itself and writes the same in the BIDR and Data buffers respectively. This mode is provided for selftest functions. Bit error is checked in this mode.

The LINFlexD ignores the LINRX signal. There is an internal feedback from its TX output to its RX input. The TX pin can be disconnected from LINTX pin by SIUL2 setting.

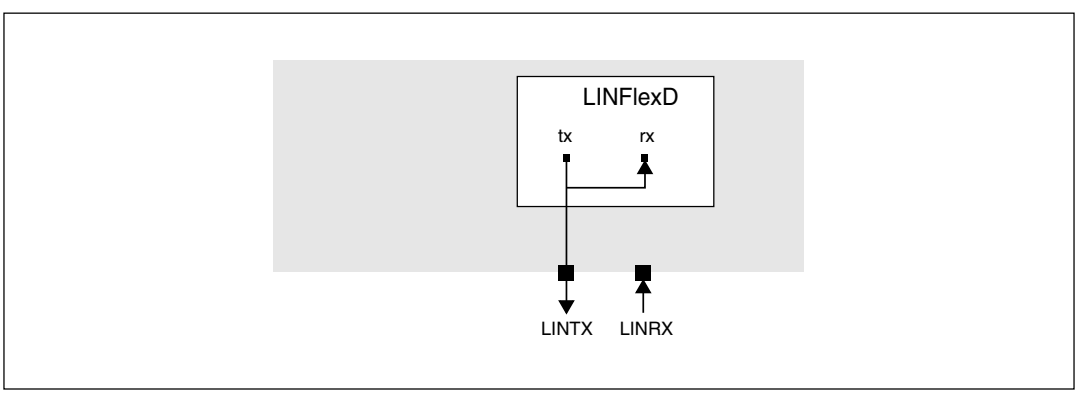


Figure 54-9. LINFlexD in Loop Back Mode

54.4.2.3 Master mode

Master mode is selected by means of the MME bit in LINCR1 register.

Header transmission

According to the LIN protocol any communication on the LIN bus is triggered by the master sending a header. The header is transmitted by the master task of a node while the response is transmitted by the slave task of a node.

To transmit the header, first program the Identifier, data field length, message direction and checksum enable in the BIDR register; then set the HTRQ bit in LINCR2. Once header transmission starts, the user should not modify BIDR register bits untill the current frame is complete. The transmitted ID is also received by the master node and copied to BIDR.

Data transmission

When the master node is the publisher of the data corresponding to the Identifier sent by the master, then the slave task of the node should send the data in the response part of the frame. Hence software must provide the data to the LINFlexD before the header transmission is requested. The data to be transmitted is stored in the message buffer BDRL and BDRM. The number of bytes to be transmitted depends on the Data Field length in BIDR. The software uses the BIDR[CCS] bit to configure the checksum type (classic or enhanced) for each message.

If the response has been sent successfully, LINSR[DTF] is set. In case of error, the DTF flag is not set and the corresponding error flag is set in the LINESR (refer to error handling).

It is possible to handle frames with a response size larger than eight bytes of data (extended frames). If the data field length in the BIDR is configured with a value higher than eight data bytes, LINSR[DBEF] is set once the first eight bytes have been transmitted. The application has to update the buffer BDR before resetting the DBEF bit. The transmission of the next bytes starts when the DBEF bit is reset. Once the last data byte (or the checksum byte) has been sent, the DTF flag is set.

The direction of the message buffer is decided by the DIR bit in the BIDR. The transmitted data is also received by the same node and copied to the buffer.

Data reception

To receive data from a slave node, the master sends a header with the corresponding Identifier. The data received from the slave is stored in the message buffer and the status of the message is stored in the LINSR.

If the response has been received successfully, the LINSR (DRF) bit is set. In case of error, the DRF flag is not set and the corresponding error flag is set in the LINESR (refer to Error handling).

It is possible to handle frames with a Response size larger than eight bytes of data (extended frames). If the data field length in the BIDR is configured with a value higher than eight data bytes, the LINSR (DBFF) bit is set once the first eight bytes have been received. The application has to read the buffer BDR before resetting the DBFF bit. Once the last data byte (or the checksum byte) has been received, the DRF flag is set.

Data Discard

If the user wants to discard the data after header transmission, then the DDRQ bit in LINCR2 should be set.

54.4.2.4 Slave mode

This mode is selected when the MME bit of the LINCR1 register is cleared.

Data transmission

On header reception, the HRF bit is set and an RX interrupt is generated. The software must then:

1. Read the received ID in the BIDR register
2. Fill the BDR[0:x] register
3. Program the CCS and DIR bits in the BIDR register
4. Specify the data field length DFL[5:0] bits in the BIDR register
5. Trigger the data transmission by setting the DTRQ bit

Note that the HRF bit should be reset only after the DTRQ bit is set. For the DTRQ to be effective, the HRF bit must be set. This is to ensure that DTRQ is not set randomly, but only after a header reception. It must be noted that you cannot set the DIR and DTRQ bits once RXbusy is asserted in LINSR.

Alternately, one or more Identifier filters are configured for transmission by setting the DIR bit, and activated by setting the enable bits in IFER. When at least one Identifier filter is active and configured for transmission and the received ID matches the filter, a TX interrupt is generated. The software can use the index in the IFMI register to point directly to the corresponding data array in the RAM and copy this data to the BDR[0:7].

The use of a filter saves software the processing time required to read the ID value in the BIDR, match it, and configure the data field length and checksum type.

If the number of filters provided by LINFlexD are not sufficient for the application, mask mode can be used for the filters.

The transmitted data is also received by the same node and copied to the buffer.

Data reception

When LINFlexD is the subscriber of the data of the received identifier, then on header reception the HRF flag is set and an RX interrupt is generated. The software must read the received ID from the BIDR register and specify the data field length before the reception of the stop bit of the first data byte. When the checksum is received, an RX interrupt is generated for software to read the received data from BDR[0:7] and the RMB bit is set. Software must then release the data buffer by resetting the RMB bit in the LINSR.

When at least one identifier filter is active and configured for reception, an RX interrupt is generated only after the checksum reception. No interrupt request is generated on reception of the ID.

If the Identifier is filtered by software then you can discard the data by setting the DDRQ bit in the LINCR2, while HRF is set.

Note that for software filtering, software must decide the type of checksum (configure the CCS bit of the BIDR register) before reception of data. Otherwise the previous value of the CCS bit is considered while calculating checksum.

54.4.2.5 Errors

54.4.2.5.1 Header error

A header error is an error during the header reception. The error types are:

1. Sync Del error (SDEF)
2. Sync field error (SFEF)
3. Identifier Parity error (IDPEF)

Sync Del error (SDEF)

The delimiter should be 1 for at least one bit time; otherwise it is considered short and consequently the receiver discards synchronization on the current header. Hence the frame is discarded. An interrupt is generated if *HEIE* bit of LINEIER is set.

Sync field error (SFEF)

The SFEF error condition is monitored differently depending on whether autosynchronization (LASE) is ON or OFF.

Case 1: Autosynchronization ON (LASE bit of LINCR1 = 1):

When Autosynchronization is enabled, the SFEF flag indicates:

- The deviation error on the Sync Field is outside the LIN specification, which allows up to 14% of period deviation between the slave and master oscillators or
- An overflow has occurred during the Sync Field Measurement, which leads to an overflow of the divider registers.

Deviation error on the Sync Field

The deviation error is checked by comparing the current baud rate (relative to the slave oscillator) with the received LIN Sync Field (relative to the master oscillator).

This check is based on a measurement between the first falling edge and the last falling edge of the Sync Field. Let's refer to this period deviation as D.

If SFEF field is asserted it means that:

D > 14.0625%

If there is no error, it means that:

D < 14.84375%

If 14.0625% < D < 14.84375%, then the Sync Field could be either consistent or inconsistent depending on dephasing between the signal on the RDI line and the LIN_CLK.

Overflow during Sync Field Measurement

This check is based on the measurement of each bit time between both edges of the Sync Field. This checks that each of these bit times is large enough (more than 12 samples) compared to the bit time of the current baud rate.

Case 2: Autosynchronization OFF (LASE bit of LINCR1 = '0')

In this case the Sync character is received as a normal character. If the received character is 0x55 then the Sync Field is OK.

On any occurrence of an inconsistent Sync Field, the receiver immediately exits from Sync_Field state and the frame is consequently discarded. The SFEF bit of LINESR is set.

Identifier Parity error(IDPEF)

There are two parity bits in the identifier field. These bits are checked against the hardware-calculated parity over the other six bits of identifier, according to the formula:

P0 = ID0 XOR ID1 XOR ID2 XOR ID4

P1 = NOT (ID1 XOR ID3 XOR ID4 XOR ID5)

This parity is checked after the ID is transferred to the BIDR register (after stop of ID has been detected properly) and upon parity mismatch, the receiver state machine exits from Identifier state immediately if the IOPE bit of LINCR2 is set.

54.4.2.5.2 Bit error

This error is flagged in transmission mode when the value read back from the bus is different from the value transmitted. Bit error checking on each bit is guaranteed if transceiver delay is less than one bit time minus 6 LIN_CPU cycles. Bit error is not checked during break field transmission.

1-bit time at 20 Kbit/s = 50 µs

6 LIN_CLK cycles at 80 MHz = 75 ns

Thus, (1-bit time) - (6 LIN_CLK cycles) = 49.925 µs

Transmission of the frame is stopped after the corrupted bit if the IOBE bit in LINCR2 is set. If IOBE is reset, the transmitter continues to transmit in spite of the bit error. An interrupt is generated if the BEIER bit is set in LINIER.

Note

If the break delimiter is not detected by the master within one bit time after delimiter transmission due to transceiver delay or error on the bus, then a train of bit error interrupts are generated. In this case, the Identifier may not be replaced in the BIDR.

Similarly, if the start of a falling edge of data is not detected by the transmitter node within one bit time after start bit transmission, then a train of bit error interrupts are generated. In this case also, data and checksum replacements in the BDR and CFR respectively are not guaranteed.

54.4.2.5.3 Framing error

This error is flagged when a dominant state is sampled on stop bit of the current received character (sync field, identifier field, data field, checksum field). LINFlexD discards the current frame and returns to Idle state. An interrupt is generated if FEIE bit is set in LINEIER.

The byte that caused framing error is also shifted to the buffer but DRF or DBFF is never set in this case.

54.4.2.5.4 Checksum error

This error is flagged when the checksum computed by hardware does not match the received checksum.

LINFlexD discards the received frame and returns to Idle state. An interrupt is generated if the CEIE bit is set in LINEIER.

54.4.2.5.5 Overrun error

Once the message buffer is full (RMB is set), the next valid message reception will lead to an overrun and the message will be lost. The hardware signals the overrun condition by setting the BOF bit. Which message is lost depends on the buffer lock function control bit RBLM.

If RBLM is cleared, the old message in the buffer is overwritten by the most recent message. If RBLM bit is set, the most recent message is discarded and the oldest message is available to the software. In the case of slave, if buffer is not released (RMB is not reset) before reception of next Identifier and if RBLM is set, then the ID along with the data is discarded.

54.4.2.5.6 Timeout error

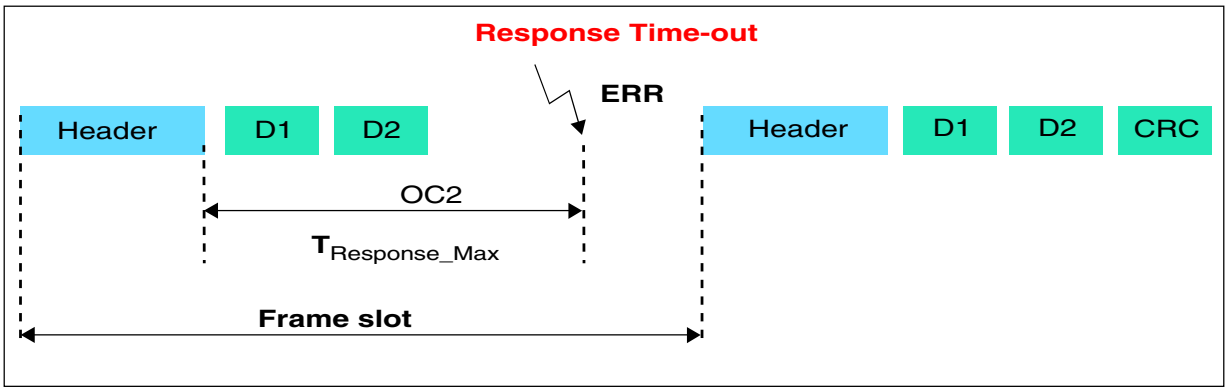


Figure 54-10. Incomplete response (for example, missing checksum)

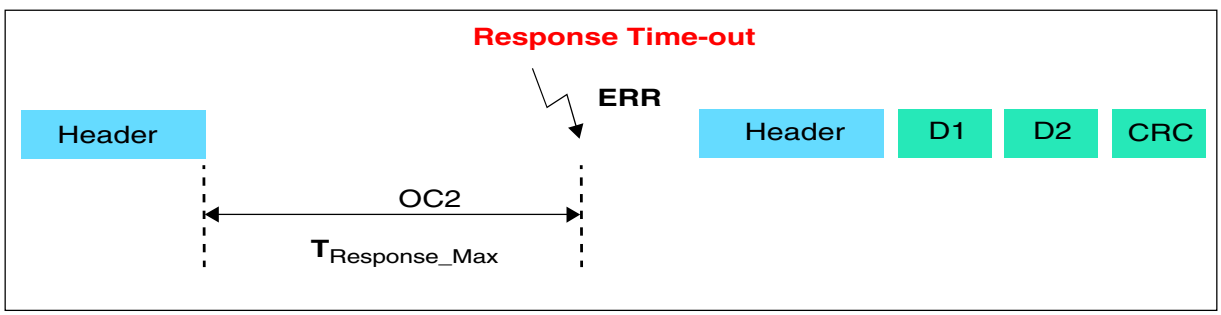


Figure 54-11. No response

Response timeout mechanism

- Master mode
  - OC2 is loaded with Nominal_time_out + LINFlexD_LINTCSR[CNT] at the end of the stop bit of the Identifier field (where nominal_time_out = 1.4 × ((DFL + 2) × 10 bit time).

    This loading takes place at the end of id field (and not at the beginning of data field). Moreover, the correct value, which depends on DFL, is loaded immediately. No further OC2 update is required.
  - In case of no response at all within this time (in other words, start of data is not received), timeout takes place when the counter reaches the OC2 value (no change compared to earlier implementation).
  - In case of an incomplete response, timeout also occurs when the counter reaches the OC2 value (no change compared to earlier implementation).
- Slave mode

- **Case 1**: The received identifier is managed by a filter. The implementation can be similar to the master mode, because the DFL value is loaded by hardware. Thus, OC2 can be loaded with Nominal_time_out + LINFlexD_LINTCSR[CNT] at the end of the stop bit of the Identifier field (where nominal_time_out = 1.4 × ((DFL + 2) × 10 bit time))
- **Case 2**: The received identifier is not managed by a filter. As the DFL value needs to be updated by software after the identifier field has been received, the implementation is the following:
  - At the end of the ID, OC2 is loaded with 36 (maximum possible response space) + LINFlexD_LINTCSR[CNT].
  - At the end of the first_data_byte it is reloaded again according to DFL
  - Before reloading, LINFlexD checks the count_val. If count value is higher than the value to be reloaded, timeout takes place immediately and no reloading occurs.

NOTE

The value of nominal_time_out (1.4 × ((DFL +2) × 10 bit time)) shown is as per the reset value of LINTOCR[RTO] and LINCR1[CFD]. It changes with change in value of LINTOCR[RTO] and LINCR1[CFD]. Here, 1.4 corresponds to (LINTOCR[RTO])/10 and 2 corresponds to (2- LINCR1[CFD])

Header timeout mechanism

- Master mode: As the header is generated by the LINFlexD, there are only two cases:
  - no error on the bus and timing is correct (nominal header length),
  - an error occurs on the bus and LINFlexD flags it in LINESR register (typically a bit error).

  Therefore there is no meaning of header timeout in master mode, so it is disabled.
- Slave mode
  - header_nominal = 13 + 2 + 10 + 10 = 35 Tbit
  - header_max = 1.4 × Header_nominal = 49 Tbit

- taking into account a possible 14% clock deviation, header_max seen by LINFlexD is 49 × 1.14 = 56 Tbit
- If the counter starts after 11 Tbit (break duration), the HTO value is 56 –11 = 45 Tbit.

The reset value of HTO is 45, and this register can only be programmed in slave mode. Counter restarts after break duration and OC1 gets loaded with the value of HTO.

As response space is not included in the response, frame timeout is no longer needed and is removed completely. Indeed, header timeout and response timeout cover all cases.

The timeout counter can be used to detect other timeouts. In this case, the MODE bit must be reset and the output compare value can be updated in the LINTOCR register by software.

Stuck at zero timeout error

If the dominant pulse lasts for a time of at least 100 bits, the SZF bit in LINESR is set. If the same dominant pulse prolongs, the subsequent SZF setting will be 87 bit times apart (instead of 100 bit times).

54.4.2.5.7 Noise

During reception each bit is sampled 16 times and the value of the bit is obtained by taking the majority value of the 8th, 9th and 10th samples. If any one of these three samples has a value different from the other two, this error is flagged.

When OSR = 8, majority of 2nd, 3rd, and 4th samples are taken into account to determine noise. Noise checking is disabled for all OSR less than eight.

This error is flagged when there is noise detected in the start bit (see LIN Error Status Register (LINFlexD_LINESR).

54.4.2.6 Identifier filtering

In LIN protocol the identifier of a message is not associated with the address of a node but is related to the content of the message. A transmitter broadcasts its message to all the receivers. Based on the header received, the receiver decides whether to receive or transmit a response (depending on the identifier value). If the message does not target the node, it should be discarded.

To fulfill this requirement, the LINFlexD provides configurable filters in order to eliminate software intervention. This hardware filtering saves CPU resources that would otherwise be required to perform filtering by software.

There are a maximum of 16 filters (depending on the generic no_of_filters) in the LINFlexD, which can be programmed by the user only during Initialization mode. In order to activate a filter the corresponding FACT bit in IFER needs to be set. There are two modes possible for each identifier depending on the corresponding IFM bit of IFMR.

Identifier list mode

If the n^{th} bit of IFMR is cleared, then filter number 2n and 2n+1 is in identifier list mode. In this mode, the maximum number of filters that can be configured for transmission/ reception equals no_of_filters, depending on the FACT bit of IFER. In this mode the identifier received should match bit by bit to the ID field of IFCR2n or IFCR2n+1 (if the corresponding FACT bit is set).

Identifier mask mode

If the number of filters required is more than no_of_filters, then filters should be configured in Mask mode. If the n^{th} bit of IFMR is set, then filter number 2n is the filter and 2n+1 acts as a mask for it. The FACT bit for filter 2n+1 has no effect in this case. In this mode, if the x^{th} bit of the mask is set, then the x^{th} bit of the received identifier must match the x^{th} bit of filter.

If there is a match of identifier with the mth filter in any mode, then m+1 is loaded in the IFMI register by hardware. No match condition is denoted by IFMI = 0.

Upon matching DFL (2:0), the CCS and DIR bits of the BIDR register are copied from the filter by hardware and from then on, the BIDR register is read-only untill the end of the frame. Now if the DIR bit of BIDR is set, then a TXI interrupt is generated if the HRIE bit of LINIER is set. In this case, software uses the IFMI register to transfer the relevant data from the RAM area to BDR, and after complete transfer the DTRQ bit of LINCR2 is set to start the transmission. If the DIR bit is cleared then an RXI interrupt is generated (provided DRIE bit of LINIER is set) when the checksum has been received and there is no checksum error.

In case of no filter match condition (IFMI = 0) and if the BF bit of LINCR1 is set, then an RXI is generated. Now it is the responsibility of software to configure BIDR and start transmission (by loading the BDR buffer and setting the DTRQ bit of LINCR2) or discard reception (by setting the DDRQ bit of LINCR2). If the BF bit is reset, then the receiver discards the received identifier and turns to Idle state in search of a new break.

Note

If one identifier matches with two filters (one is in the list and the other in Mask mode) then List mode prevails over Mask mode. In mask mode if two filters match with the identifier then the filter having the lower number prevails.

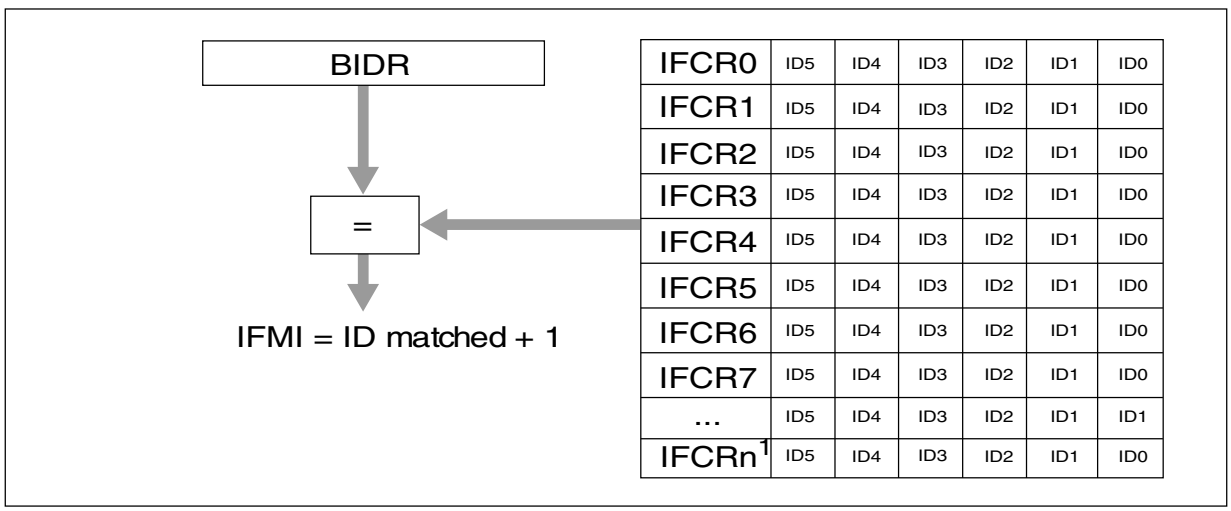


Figure 54-12. Identifier List mode

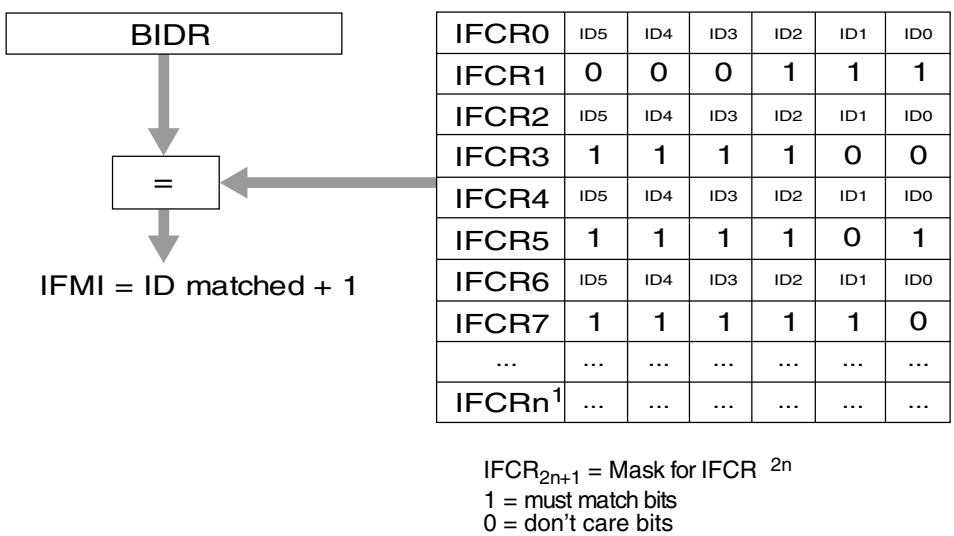


Figure 54-13. Identifier Mask mode

54.4.2.7 Start detection and break delimiter detection in receiver

There is a 10-bit-shift register sample register in the receiver that shifts signal data to the next least significant bit on each incoming sample.

Table 54-3. Start Detection and Delimiter Detection in receiver

9	8	7	6	5	4	3	2	1	0	Sample_reg
—	—	—	—	—	—	—	—	—	RT1	counter = 0
—	—	—	—	—	—	—	—	RT1	RT2	counter = 1
—	—	—	—	—	—	—	RT1	RT2	RT3	counter = 2
—	—	—	—	—	—	RT1	RT2	RT3	RT4	counter = 3
—	—	—	—	—	RT1	RT2	RT3	RT4	RT5	counter = 4
—	—	—	—	RT1	RT2	RT3	RT4	RT5	RT6	counter = 5
—	—	—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	counter = 6
—	—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	counter = 7
—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	counter = 8
RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	counter = 9
RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	counter = 10
RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	counter = 11
RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	counter = 12
RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	counter = 13
RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	RT15	counter = 14
RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	RT15	RT16	counter = 15
1	1	1	0	x	0	x	0	x	0	start detect
1	1	1	0	x	0	x	0	x	1	
1	1	1	0	x	0	x	1	x	0	
1	1	1	0	x	1	x	0	x	0	
0	0	0	1	x	1	x	1	x	1	delimiter detect
0	0	0	1	x	1	x	1	x	0	
0	0	0	1	x	1	x	0	x	1	
0	0	0	1	x	0	x	1	x	1	

Hence, a start is detected as soon as it is qualified with 1110 in the sample register and verified with at least two out of three predefined verification samples being zero. Similarly for delimiter detection, qualification samples are 0001 and at least two of the three verification samples are one. The Noise Flag is set if the start is verified with only two valid samples.

54.4.2.7.1 Start detection mechanism

The following steps are followed for detecting a start bit:

1. Refer to Table 54-3 for the status of the sample register (shift register) when count = 6
2. At this point, if RT1 (the first incoming sample) = 0 and the previous three samples already received are all ones, then it might be a start bit.
3. To make sure it is indeed a start bit, the incoming data samples in the sample register (4), sample register (2) and sample register (0) which correspond to RT3, RT5, and RT7 is verified using the steps mentioned below.
4. If the majority (i.e., 2) out of these 3 samples are equal to '0', then start bit is said to be detected.
5. These three samples RT3, RT5, and RT7 are called the "verification samples" which are checked at count = 6.
6. The sample register bits 9, 8, 7, and 6 are called "qualification samples" which are checked at count = 6.
7. At count = 9, if majority value of RT8, RT9, and RT10 is not equal to '0' then Noise flag is set.

54.4.2.7.2 Break delimiter detection mechanism

The following steps are followed for detecting a delimiter bit:

1. Refer to Table 54-3 for the status of the sample register (shift register) when count = 6
2. At this point, if RT1 (the first incoming sample) = 1 and the previous three samples already received are all zeros, then it might be a delimiter bit.
3. To make sure it is indeed a delimiter bit, the incoming data samples in the sample register (4), sample register (2) and sample register (0) which correspond to RT3, RT5, and RT7 is verified using the steps mentioned below.
4. If the majority (i.e., 2) out of these 3 samples are equal to '1', then delimiter bit is said to be detected.
5. These three samples RT3, RT5, and RT7 are called the "verification samples" which are checked at count = 6.
6. The sample register bits 9, 8, 7, and 6 are called "qualification samples" which are checked at count = 6.

Hence, a start is detected as soon as it is qualified with 1110 in the sample register and verified with at least two out of three predefined verification samples being zero. Similarly for delimiter detection, qualification samples are 0001 and at least two out of the three verification samples are one. The Noise Flag is set if the start is verified with only two valid samples.

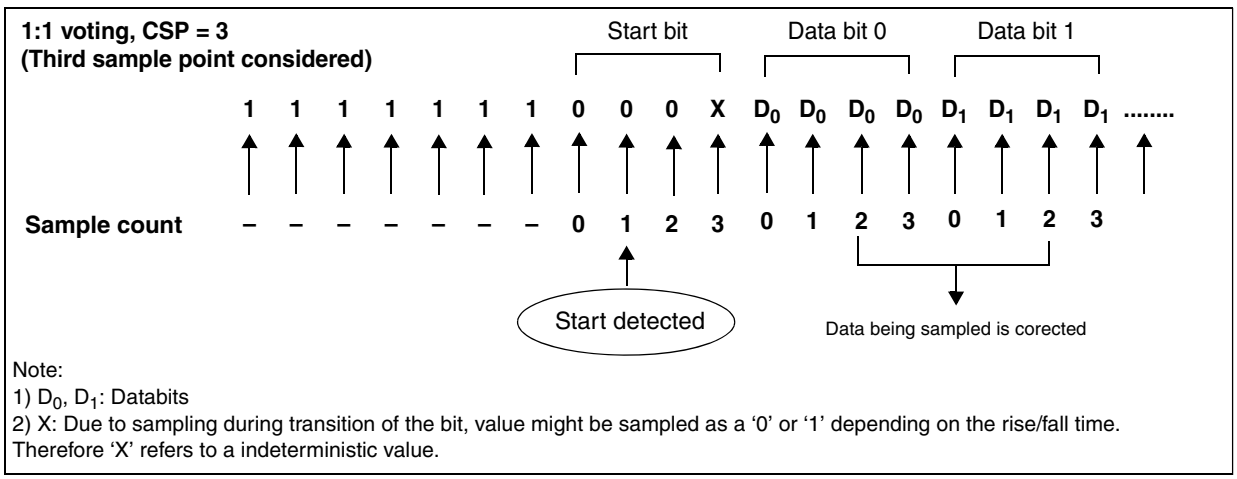


Figure 54-14. Start detection and sampling for over sampling rate = 4 (Case 1)

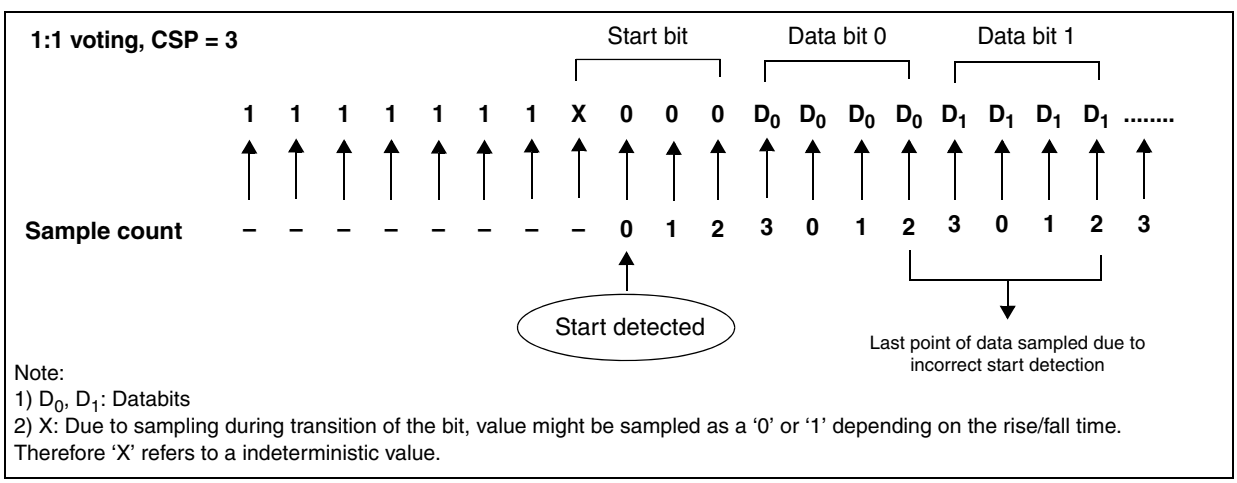


Figure 54-15. Start detection and sampling for over sampling rate = 4 (Case 2a)

The choice of the correct sample points depends on the external Rx signal quality. During the application development process, the error information indicated by the parity error can help to find the best setting for an application-specific hardware signal.

To sample at the correct point therefore, CSP has to be changed to two; resulting in the following situation:

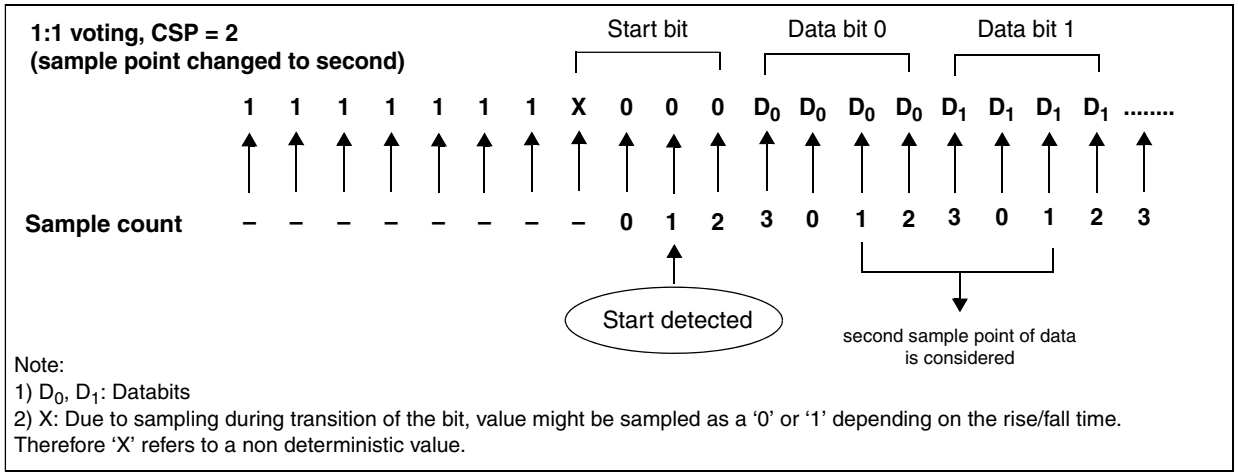


Figure 54-16. Start detection and sampling for over sampling rate = 4 (Case 2b)

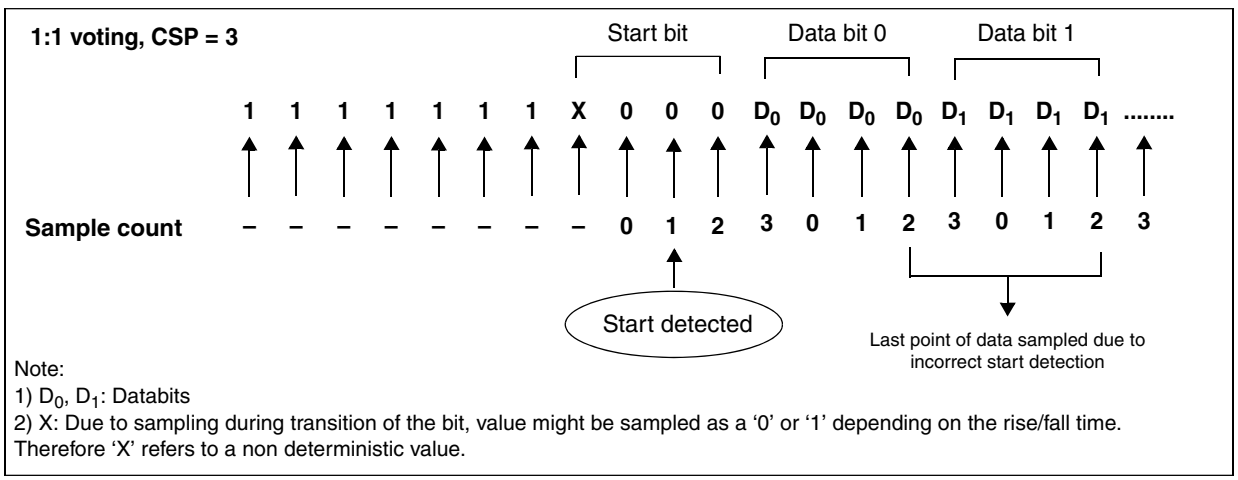


Figure 54-17. Start detection and sampling for over sampling rate = 4 (Case 3)

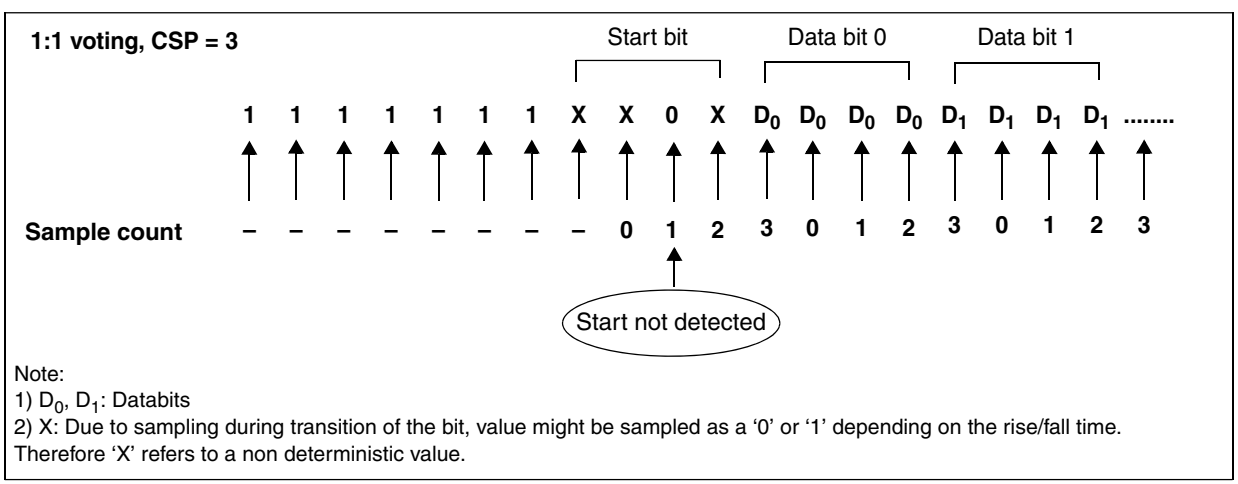


Figure 54-18. Start detection and sampling for over sampling rate = 4 (Case 4)

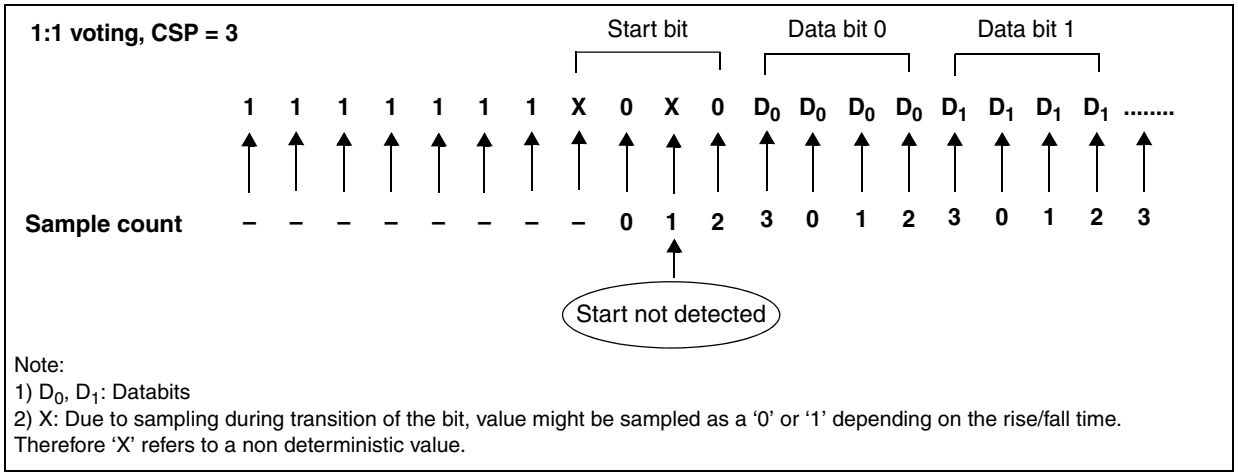


Figure 54-19. Start detection and sampling for over sampling rate = 4 (Case 5)

54.4.2.8 Baud rate generation

The LIN baud rate is programmed in two registers: the LIN Integer Baud Rate Register and the LIN Fraction baud rate register. The Baud Rate Registers can be programmed only during Initialization mode.

Baud rate is calculated with the following formula for both receiver and transmitter.

When ROSE = 1, Tx = Rx = LIN_CLK / (OSR × LDIV). When ROSE = 0, Tx = Rx = LIN_CLK / (16 × LDIV)

Where LIN_CLK is the frequency of the baud clock.

LDIV is an unsigned fixed point number. The mantissa is coded into 20 bits of LINIBRR and the fraction is coded on 4 bits of LINFBRR.

When reduced oversampling is enabled, LINFBRR should not be used and programmed to zero and LDIV contains only the integer part of LINIBRR.

For example: When ROSE = 0 (For LIN and UART mode): LDIV = 468.75 d, LIN_CLK = 36 MHz, LINIBRR = 468 d, LINFBRR = 12 *Baud rate = 36 MHz/ (16 × 468.75) = 4.8 Kbit/s*

For example: When ROSE = 1 (Only for UART mode): LDIV = 10 d, LIN_CLK = 80 MHz, LINIBRR = 4 d, OSR = 4, *Baud rate = LIN_CLK / (OSR × LDIV) = 80 MHz / (4 × 10) = 2 Mbit/s*

54.4.2.9 Automatic resynchronization

To automatically adjust the baud rate based on measurement of the LIN sync field, write the nominal Prescalar value (nominal baud rate) in LINIBRR and LINFBRR, then set the LASE bit in LINCR1 to enable automatic synchronization.

When auto synchronization is enabled, after each LIN Sync Del, the time duration between five falling edges on RDI is sampled on LIN_CLK.

54.4.2.10 Wakeup management

Any node in a sleeping LIN cluster may request a wakeup. The wakeup request is issued by forcing the bus to the dominant state for 250 μs to 5 ms. Every slave node should detect the wakeup request (a dominant pulse longer than 150 μs) and be ready to listen to bus commands within 100 ms, measured from the ending edge of the dominant pulse. The master also wakes on detecting a wakeup request and when the slaves are ready, starts sending frame headers to find out the cause of the wakeup. If the master does not issue a frame header within 150 ms from the wakeup request, then the node issuing a request may try issuing a new wakeup request.

In LINFlexD, a wakeup request can be generated by writing the wakeup character in BDR0 and setting the WURQ bit in LINCR2. On setting the WURQ bit, the character in BDR0 is transmitted. For LIN 2.0, character 0xF0 is sent as the wakeup character.

In LINFlexD, wakeup can be detected in two ways.

1. AUTOWU = 1 On detecting a falling edge in sleep mode, the SLEEP bit is cleared by hardware, the WUF flag is set, and an interrupt is generated if WUPIE is set. LINFlexD is now in normal mode and ready to receive frames.
2. AUTOWU = 0 On detecting a falling edge the WUF flag is set and an interrupt is generated (if WUPIE bit is set). It is then up to the software to clear the SLEEP bit.

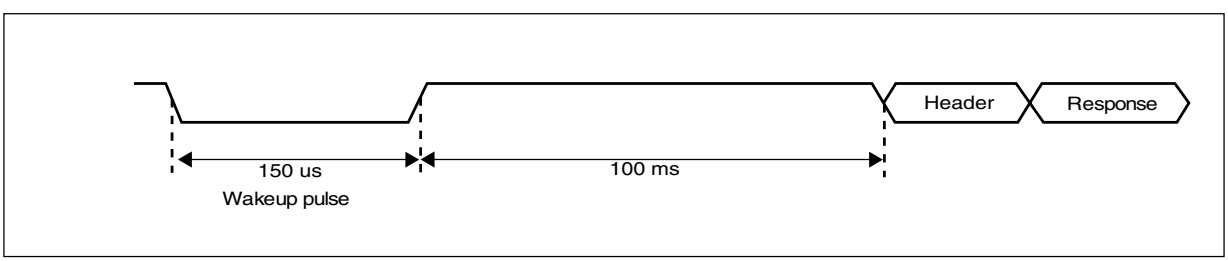


Figure 54-20. Wakeup sequence

54.4.3 Timer

There is an 8-bit counter which has different behavior in different modes as described below:

- **In Output Compare Mode (LINTCSR[MODE]= 1):**

  This counter is running even in Sleep and Initialization mode. The counter value which when matches the two software configurable output compare registers (LINOCR[OC1] or LINOCR[OC2]), generate output compare interrupt (LINESR[OCF]) provided TOCE bit in LINTCSR is set. Once an interrupt occurs subsequent interrupt is generated only when the application has reset the stuck FSM state.
- **In LIN Mode:**

  The software has no control over the TOCE bit and output compare registers are utilized for generation of LIN timeout interrupts (header, frame, response times out). In this case, if LIN moves to Sleep or Init state then this counter remains in Reset state. LIN mode has no meaning if UART is enabled, hence the counter will remain in Reset state.

54.4.4 UART mode

Main features in the UART mode are:

- Full duplex communication
- 8-bit frames, 9-bit frames, 13-bit frames, 16-bit frames, 17-bit frames
- Even/Odd/0/1 Parity
- User programmable over sampling rate to obtain the baud rate of up to 25 Mbit/s

54.4.4.1 8-bit data frames

The eighth bit can be a data or a parity bit. Even/Odd/0/1 parity can be selected by the PC[1:0] bit in the same register. An even parity will be set if the modulo-2 sum of the seven data bits is one. An odd parity will be cleared in this case.

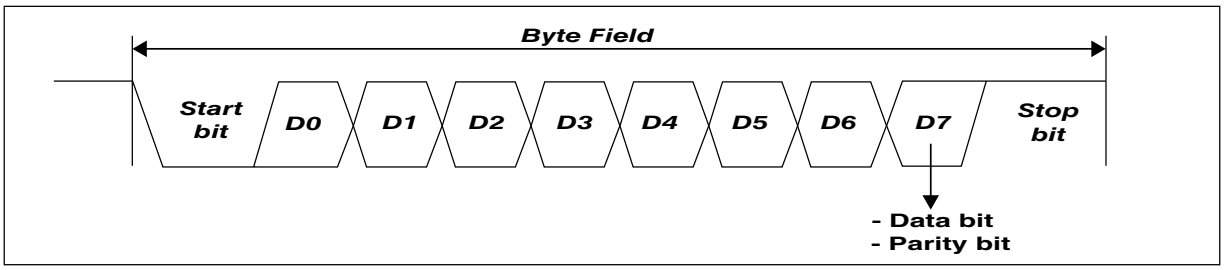


Figure 54-21. UART mode 8-bit data frame

54.4.4.2 9-bit frames

The ninth bit should be a parity bit. Even/Odd/0/1 Parity can be selected by the PC[1:0] field in the same register. An even parity will be set if the modulo-2 sum of the seven data bits is one. An odd parity will be cleared in this case. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

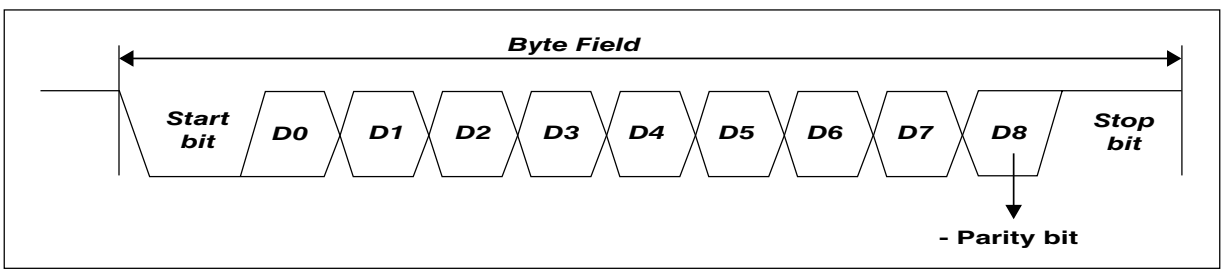


Figure 54-22. UART mode 9-bit data frame

54.4.4.3 16-bit data frames

The sixteenth bit can be a data or a parity bit. Even/Odd/0/1 Parity bit can be selected by the PC[1:0] bit in the same register. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

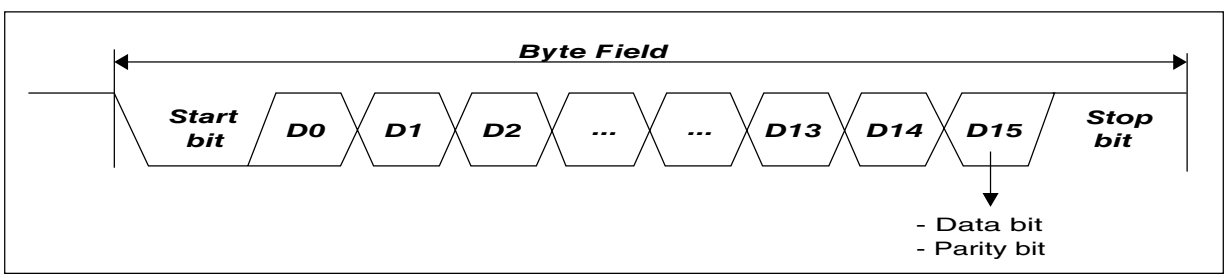


Figure 54-23. UART mode 16-bit data frame

54.4.4.4 17-bit frames

The seventeenth bit is the parity bit. Even/Odd/0/1 Parity bit can be selected by the PC[1:0] bit in the same register. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

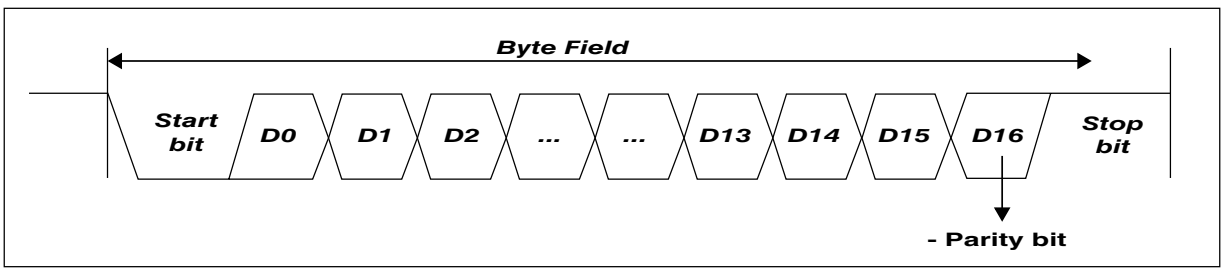


Figure 54-24. UART mode 17-bit data frame

54.4.4.5 13-bit frames

Whenever WLS is one, special word length is selected in UART mode. This bit enables 12-bit + parity bit reception only in FIFO mode.

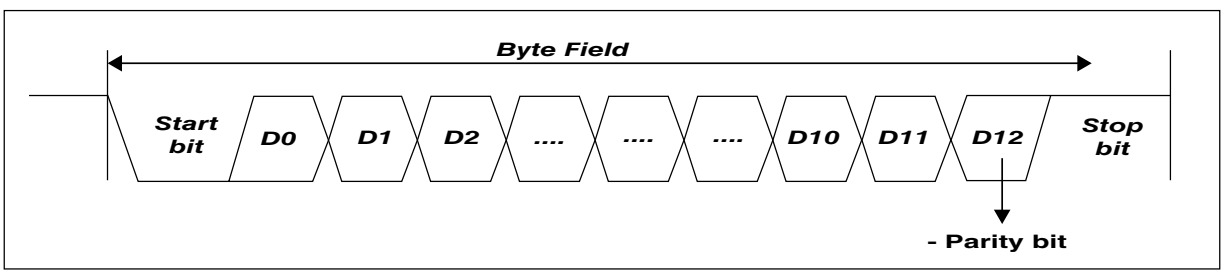


Figure 54-25. UART mode 13-bit data frame

54.4.4.6 Buffer in UART mode

The 8-byte buffer is divided into two parts: one for receiver and one for transmitter, as shown in the following figure:

Tx0	BDR0
Tx1	BDR1
Tx2	BDR2
Tx3	BDR3
Rx0	BDR4
Rx1	BDR5
Rx2	BDR6
Rx3	BDR7

Figure 54-26. Structure of 8-byte buffer

For 16-bit frames, the lower eight bits are written in BDR0 and upper eight bits in BDR1.

The same applies for reception.

54.4.4.7 UART transmitter

UART transmitter

In order to start transmission in the UART mode, UART bit should be set and the transmitter enable bit should be set. Transmission starts when the BDR0 (least significant data byte) is programmed and continues until the number of bytes/halfwords transmitted is equal to the value in the TDFL bits in UARTCR.

The transmit buffer is four bytes (when UARTCR[WL1] = 0) or two halfwords (when UARTCR[WL1] = 1), hence a maximum of four bytes (two halfwords) transmission can be triggered. Once the programmed number of bytes (halfwords) has been transmitted, the DTF flag is set in UARTSR. If the TxEn bit of UART is reset in the middle of a transmission, then the current transmission is completed and no further transmissions can be invoked.

The buffer can be configured in FIFO mode (mandatory when the DMA Tx is enabled) by setting the control bit UARTCR[TFBM].

NOTE

If TFF bit is set and a write is performed to the FIFO, the data transmitted may be erroneous.

Table 54-4. BDRL access in UART mode

IPS operation	Register	Mode (UARTCR[TFBM])	Word length (UARTCR[WL])	IPS operation result
Write Byte0	BDRL	FIFO	Byte	OK
Write Byte1-2-3	BDRL	FIFO	Byte	IPS transfer error
Write Half-word0-1	BDRL	FIFO	Byte	IPS transfer error
Write Word	BDRL	FIFO	Byte	IPS transfer error
Write Byte0-1-2-3	BDRL	FIFO	Half-word	IPS transfer error
Write Half-word0	BDRL	FIFO	Half-word	OK
Write Half-word1	BDRL	FIFO	Half-word	IPS transfer error
Write Word	BDRL	FIFO	Half-word	IPS transfer error
Read Byte0-1-2-3	BDRL	FIFO	Byte/Half-word	IPS transfer error
Read Half-word0-1	BDRL	FIFO	Byte/Half-word	IPS transfer error
Read Word	BDRL	FIFO	Byte/Half-word	IPS transfer error
Write Byte0-1-2-3	BDRL	BUFFER	Byte/Half-word	OK
Write Half-word0-1	BDRL	BUFFER	Byte/Half-word	OK

Table continues on the next page...

Table 54-4. BDRL access in UART mode (continued)

IPS operation	Register	Mode (UARTCR[TFBM])	Word length (UARTCR[WL])	IPS operation result
Write Word	BDRL	BUFFER	Byte/Half-word	OK
Read Byte0-1-2-3	BDRL	BUFFER	Byte/Half-word	OK
Read Half-word0-1	BDRL	BUFFER	Byte/Half-word	OK
Read Word	BDRL	BUFFER	Byte/Half-word	OK

In UART FIFO mode (UARTCR[TFBM] = 1), any read operation causes an IPS transfer error.

54.4.4.8 UART receiver

UART receiver

The reception of a data byte is started as soon as the user exits initialization mode, sets the RxEn bit, and detects start bit. There is a dedicated 4-byte (if UARTCR[WL1] = 0) or 2-half-words (if UARTCR[WL1] = 1) data buffer for received data. Once the programmed number (RDFL bits) of bytes have been received, the DRF flag is set in UARTSR and the current reception gets completed. RxEn bit needs to be set only to start the reception.The reception is automatically completed as soon as the programmed number (RDFL bits) of bytes have been received.

The buffer can be configured in FIFO mode (mandatory when the DMA Rx is enabled) by setting the control bit UARTCR[RFBM].

Table 54-5. BDRM access in UART mode

IPS operation	Register	Mode (UARTCR[RFBM])	Word length (UARTCR:WL)	IPS operation result
Read Byte4	BDRM	FIFO	Byte	OK
Read Byte5-6-7	BDRM	FIFO	Byte	IPS transfer error
Read Half-word2-3	BDRM	FIFO	Byte	IPS transfer error
Read Word	BDRM	FIFO	Byte	IPS transfer error
Read Byte4-5-6-7	BDRM	FIFO	Half-word	IPS transfer error
Read Half-word2	BDRM	FIFO	Half-word	OK
Read Half-word3	BDRM	FIFO	Half-word	IPS transfer error
Read Word	BDRM	FIFO	Half-word	IPS transfer error
Write Byte4-5-6-7	BDRM	FIFO	Byte/Half-word	IPS transfer error
Write Half-word2-3	BDRM	FIFO	Byte/Half-word	IPS transfer error
Write Word	BDRM	FIFO	Byte/Half-word	IPS transfer error
Read Byte4-5-6-7	BDRM	BUFFER	Byte/Half-word	OK

Table continues on the next page...

Table 54-5. BDRM access in UART mode (continued)

IPS operation	Register	Mode (UARTCR[RFBM])	Word length (UARTCR:WL)	IPS operation result
Read Half-word2-3	BDRM	BUFFER	Byte/Half-word	OK
Read Word	BDRM	BUFFER	Byte/Half-word	OK
Write Byte4-5-6-7	BDRM	BUFFER	Byte/Half-word	IPS transfer error
Write Half-word2-3	BDRM	BUFFER	Byte/Half-word	IPS transfer error
Write Word	BDRM	BUFFER	Byte/Half-word	IPS transfer error

Note

Refer to the layout of the registers BDRL and BDRM to identify the mapping between byte x and data bits of the registers BDRL/BDRM.

Note

- If the user does not know in advance how many bytes are to be received, RDFL should not be programmed in advance. The reset value of RDFL is zero. This will ensure that the reception will happen byte by byte. The state machine will move to the Idle state after each byte reception.
- If RDFL is programmed for a certain value but that number of bytes are not received, then reception will hang. In that case, software needs to take care of timeout by seeing the flag. Software has to set the sleep bit to move to Idle state.
- If a STOP request arrives in the middle of one reception, it is only acknowledged after all the programmed number of data bytes have received; it is not served immediately. If the programmed number of data bytes are not received, then software has to take care of timeout. When the state machine moves to Idle state, then only a stop request is served.
- If during reception of any byte a parity error occurs, then the corresponding PEx bit in UARTSR is set. No interrupt is generated in this case. If a framing error occurs in any byte (FE bit in UARTSR is set) then an interrupt is generated if FEIE bit in LINIER is set. As there is only one register bit for framing error, this interrupt will be helpful in identifying which byte has framing error.

- If the last received frame has not been read from the buffer (in other words, RMB is not reset by the user), then upon reception of the next byte, an overrun error occurs (BOF bit in UARTSR will be set) and one message is depending on RBLM bit of LINCR1. An interrupt is generated if the BOIE bit in LINIER is set.
- When WLS bit = 1, BDRM access depends on WL bit setting.
- When WLS bit = 1, WL = 0 or 1 should not be used, since this will lead to incorrect reception of data.
- When IPG_STOP is requested, for UART in FIFO mode, SW has to set the INIT bit in order to send the receiver to idle. This ensures that IPG_STOP_ACK is generated and IP enters STOP mode.

54.4.5 DMA interface

54.4.5.1 Main features

The LINFlexD DMA offers a parametric and programmable solution with the following distinctive features:

- LIN Master node, TX mode: single DMA channel
- LIN Master node, RX mode: single DMA channel
- LIN Slave node, TX mode: 1 to N DMA channel where N = max number of ID filters
- LIN Slave node, RX mode: 1 to N DMA channel where N = max number of ID filters
- UART node, TX mode: single DMA channel
- UART node, RX mode: single DMA channel + time-out

54.4.5.2 Definitions

Control/status register fields of the LINFlexD macrocell are described in Table 54-6 and Table 54-7.

Table 54-6. LINFlexD control/status fields description

Register	Field	Level	Description
LINCR2	DDRQ	High	Data discard request in reception mode
	DTRQ	High	Data transmission request (slave mode) of the LIN data field stored in BDRL/BDRM
	HTRQ	High	Header transmission request (master mode)
BIDR	DFL [5:0]	—	Data field length: • 1 to 8 bytes: normal frames • 1 to 64 bytes: extended frames
	DIR	—	Direction of the data field: • 0 => reception • 1 => transmission
	CCS	—	Checksum type: • 0 enhanced • 1 classic
	ID[5:0]	—	Identifier.
LINSR	RMB	—	Buffer data ready to be read via CPU/DMA. • 0 buffer data is free • 1 buffer data is ready
	DBFF	High	Data buffer full (8 bytes received) and DFL > 7. Used for the extended frames.
	DBEF	High	Data buffer empty (8 bytes have been transmitted) and DFL > 7. Used for the extended frames.
	DRF	High	Data reception completed
	DTF	High	Data transmission completed
	HRF	High	Header received. Used in slave mode in case of Tx filter match.
IFMI	IFMIx	!= zero	Filter match index (slave mode).
UARTCR	FBM	—	FIFO/Buffer mode • 0 Buffer mode • 1 FIFO mode (mandatory in DMA mode)

Table continues on the next page...

Table 54-6. LINFlexD control/status fields description (continued)

Register	Field	Level	Description
UARTSR	RFE	—	Rx FIFO empty • 0 Rx FIFO not empty • 1 Rx FIFO empty
	TFF	—	Tx FIFO full • 0 Tx FIFO not full • 1 Tx FIFO full

DMATXE[2**TX_CH_NUM][1]		High	DMA Tx channel enable (read/write). In UART or LIN master mode only the channel 0 can be programmed.
			In LIN slave mode: # DMA TX channel = IFMI[3:0] -1.
			DMATXE[x] = 0b => TX channel x disabled
			DMATXE[x] = 1b => TX channel x enabled

DMARXE[2**RX_CH_NUM][2]		High	DMA Rx channel enable (read/write). In UART or LIN master mode only the channel 0 can be programmed.
			In LIN slave mode: # DMA RX channel = IFMI[3:0] -1.
			DMARXE[x] = 0b => RX channel x disabled
			DMARXE[x] = 1b => RX channel x enabled

1. ** stands for exponentiation.
2. ** stands for exponentiation

Table 54-7. LINFlexD DMA control fields description

Field	Mode	Value	Level	Description
DMA_TEN	LIN master Tx or UART Tx	DMATXE[0]	High	Logical AND between the 2 DMA enable bits (LINFlexD and eDMA).
DMA_TEN	LIN slave Tx	DMATXE[x] x = IFMI — 1	High	
DMA_REN	LIN master Rx or UART Rx	DMARXE[0]	High	
DMA_REN	LIN slave Rx	DMARXE[x] x = IFMI — 1	High	

Control/status fields of the TCD descriptors referred to in this document are described in Table 54-8.

Table 54-8. TCD control fields description

TCD Field	Level	Description
CITER[14:0]	—	Current "major" iteration count
BITER[14:0]	—	Beginning "major" iteration count
NBYTES[31:0]	—	Inner "minor" byte transfer count. Number of bytes to be transferred in each service request of the channel.
SADDR[31:0]	—	Source address
SOFF[15:0]	—	Source address signed offset applied to the current source address as each source read is completed.
SSIZE[2:0]	—	Source data transfer size 000 => 8-bit 001 => 16-bit 010 => 32-bit 011 => 64-bit
SLAST[31:0]	—	Last source address adjustment
DADDR[31:0]	—	Destination address
DOFF[15:0]	—	Destination address signed offset applied to the current destination address as each destination write is completed.
DSIZE[2:0]	—	Destination data transfer size 000 => 8-bit 001 => 16-bit 010 => 32-bit 011 => 64-bit
DLAST_SGA[31:0]	—	Last destination address adjustment or the memory address for the next TCD to be loaded into this channel (scatter/gather)
INT_MAJ	High	Enable an interrupt when major iteration count completes
START	High	The channel is explicitly started via a software initiated service request.
DONE	High	Channel done (the DMA has completed the outer major loop).
D_REQ	High	Disable request. If this flag is set the DMA hardware automatically clears the corresponding DMAERQ bit when the current major iteration count reaches zero.

54.4.5.3 Master node –TX mode

On a master node, in TX mode the DMA interface requires a single TX channel. Each TCD controls a single frame except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in the following figure.

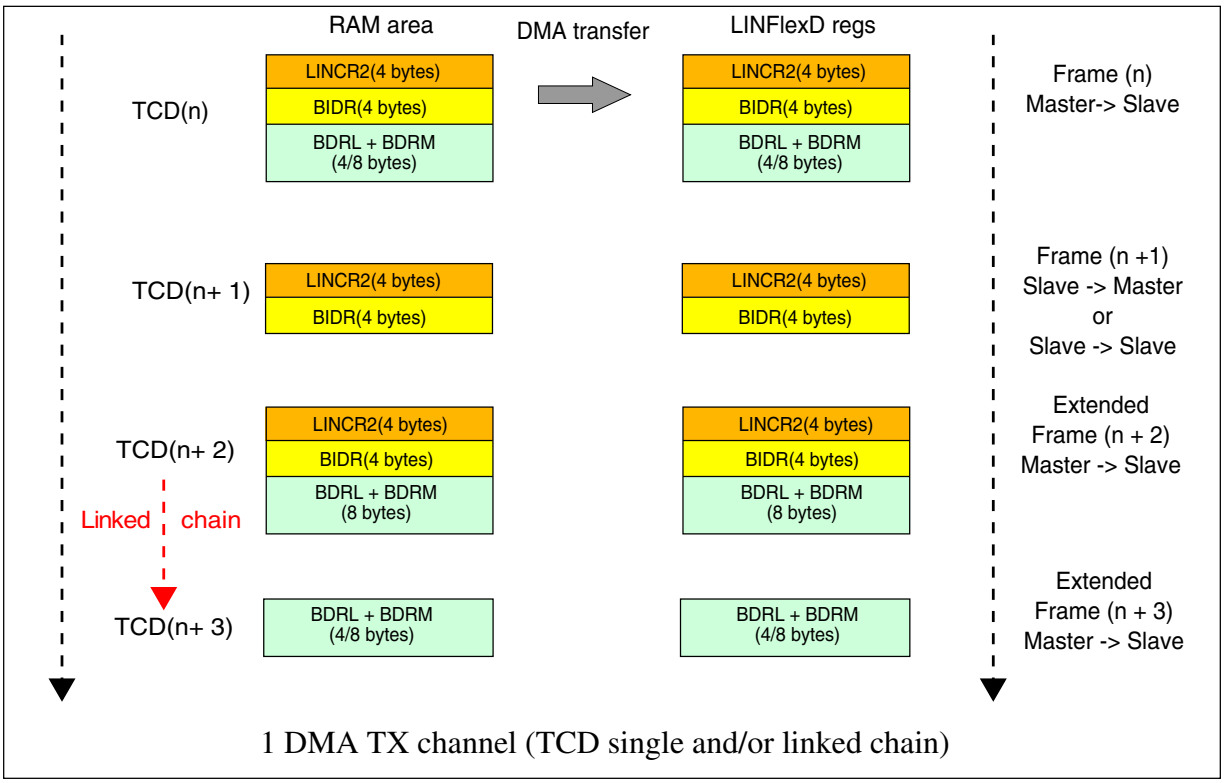


Figure 54-27. Master node –TX memory map

The TCD chain of the DMA Tx channel on a master node supports:

- Master to Slave: transmission of the entire frame (header + data)
- Slave to Master: transmission of the header; the data reception is controlled by the Rx channel on the master node
- Slave to Slave: transmission of the header

The following table provides the register settings of the LINCR2 and BIDR for each class of LIN frame.

Table 54-9. Master node – Tx mode – Register setting

LIN frame	LINCR2	BIDR
Master to Slave	DDRQ = 1	DFL = payload size
	DTRQ = 0	ID = address
	HTRQ = 0	CCS = checksum
		DIR = 1 (TX)

Table continues on the next page...

Table 54-9. Master node – Tx mode – Register setting (continued)

LIN frame	LINCR2	BIDR
Slave to Master	DDRQ = 0 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)
Slave to Slave	DDRQ = 1 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)

The concept FSM to control the DMA Tx interface is given in the following figure. DMA TX FSM moves to Idle state immediately at next clock edge if DMATXE[0] = 0. The TCD setting (word transfer) is given in Table 54-10. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 54-10. TCD setting – Master node – Tx mode

TCD Field	Value	Description
CITER[14:0]	1	Single iteration for the "major" loop
BITER[14:0]	1	Single iteration for the "major" loop
NBYTES[31:0]	[4 + 4] + 0/4/8 = N	Data buffer is stuffed with dummy bytes if the length is not word aligned. LINCR2 + BIDR + BDRL + BDRM
SADDR[31:0]	-	RAM address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	
DADDR[31:0]	-	LINCR2 address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No SW request

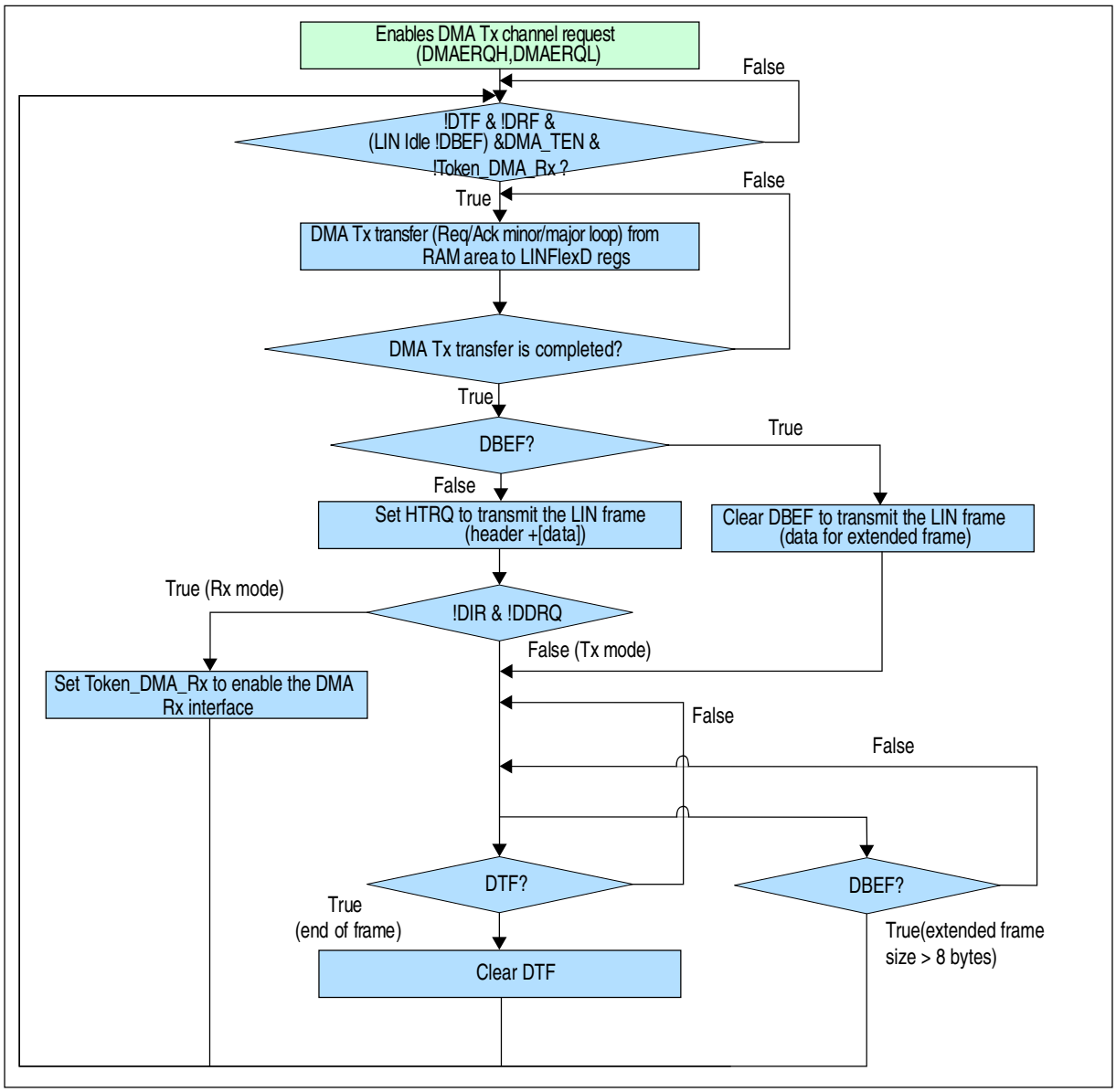


Figure 54-28. Master node – DMA Tx FSM (concept scheme)

54.4.5.4 Master node - RX mode

On a master node in RX mode, the DMA interface requires a single RX channel. Each TCD controls a single frame , except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in the following figure.

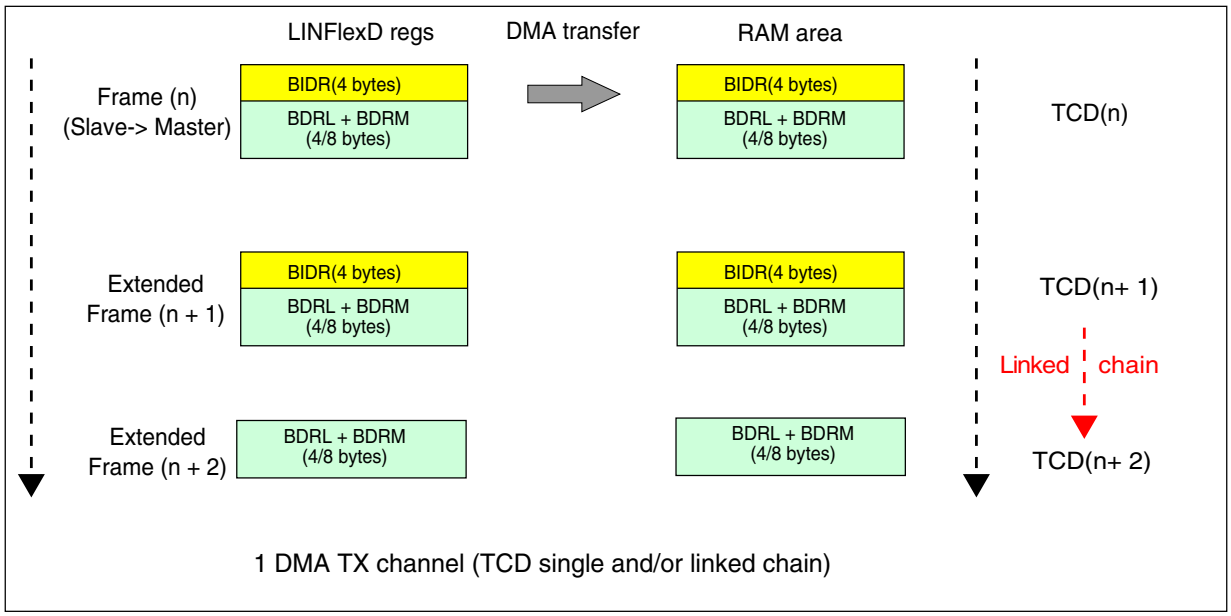


Figure 54-29. Master node – RX memory map

The TCD chain of the DMA Rx channel on a master node supports Slave to Master: reception of the data field of the header

The BIDR register is optionally copied into the RAM area. This BIDR field (part of FIFO data) contains the ID of each message, which only allows the CPU to figure out which ID the LINFlexD DMA received if the single DMA channel setup is used.

The concept FSM to control the DMA Rx interface is given in the following figure. DMA RX FSM moves to Idle state immediately at the next clock edge if DMARXE[0] = 0. The TCD setting (word transfer) is given in the next table. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 54-11. TCD setting – Master node – Rx mode

TCD Field	Value	Description
CITER[14:0]	1	Single iteration for the "major" loop
BITER[14:0]	1	Single iteration for the "major" loop
NBYTES[31:0]	[4] +4/8 = N	Data buffer is stuffed with dummy bytes if the length is not word aligned. BIDR + BDRL + BDRM
SADDR[31:0]	—	BIDR address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	RAM address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer

Table continues on the next page...

Table 54-11. TCD setting – Master node – Rx mode (continued)

TCD Field	Value	Description
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No SW request

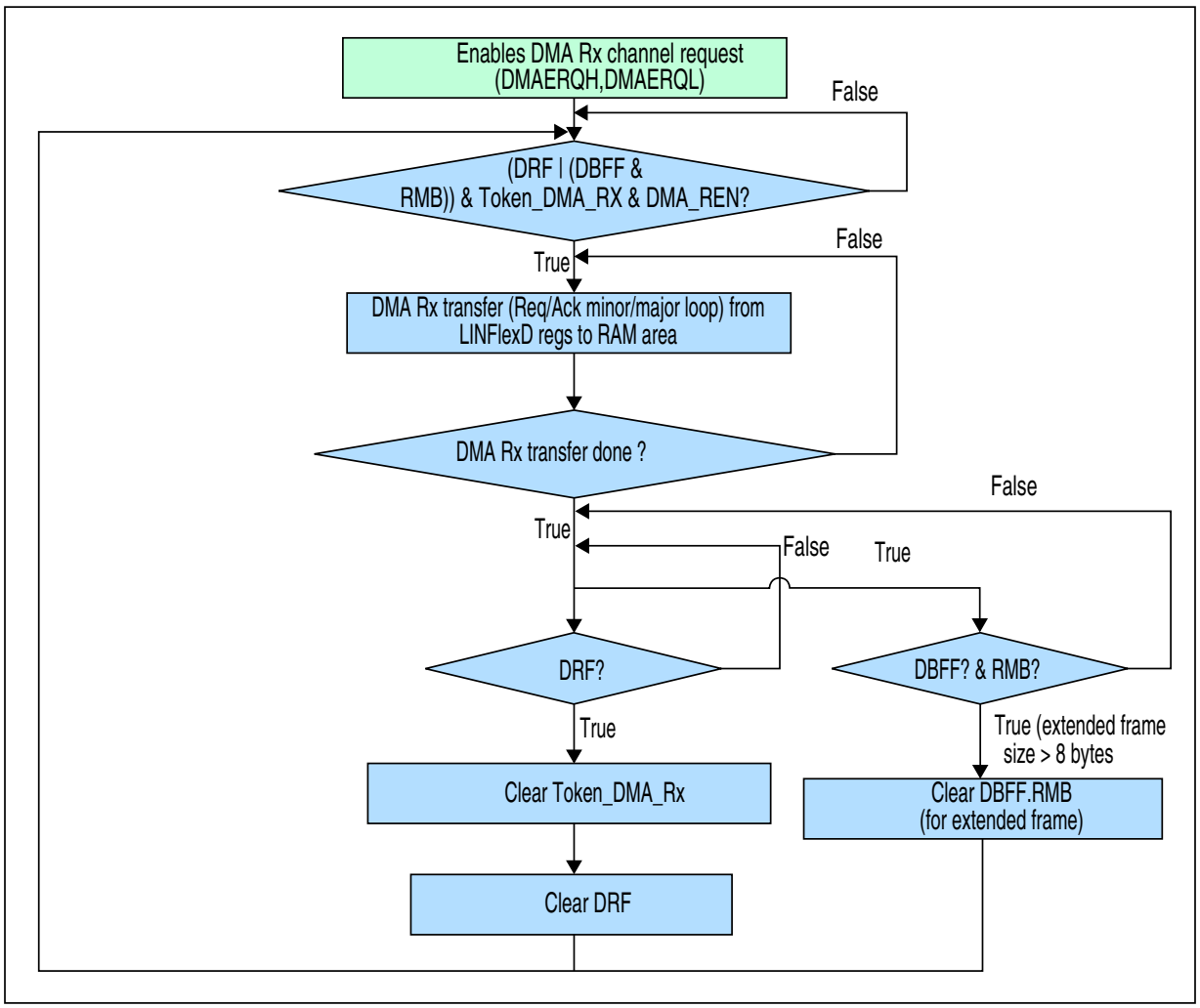


Figure 54-30. Master node – DMA Rx FSM (concept scheme)

54.4.5.5 Slave node – TX mode

On a slave node in TX mode, the DMA interface requires a DMA TX channel for each ID filter programmed in TX mode. In case a single DMA TX channel is available, a single ID field filter must be programmed in TX mode. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in the following figure.

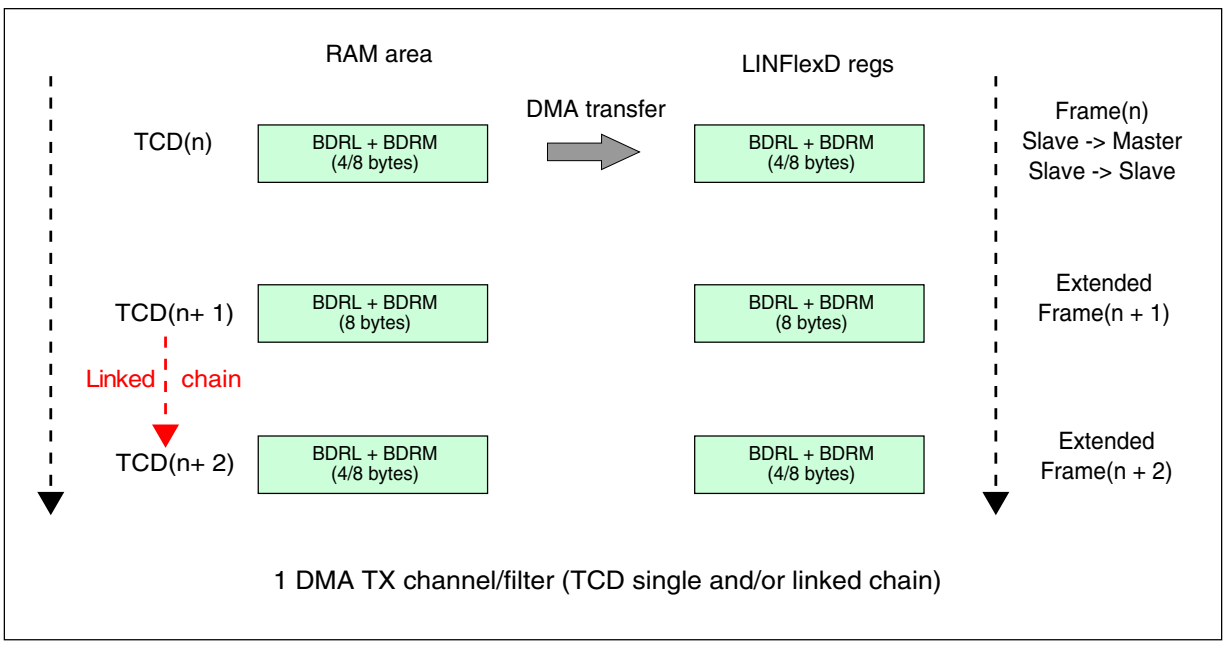


Figure 54-31. Slave node – TX memory map

The TCD chain of the DMA Tx channel on a slave node supports:

- Slave to Master: transmission of the data field
- Slave to Slave: transmission of the data field

The register setting of the LINCR2, IFER, IFMR, and IFCR registers is given in Table 54-12.

Table 54-12. Slave node – Tx mode – Register setting

LIN frame	LINCR2	IFER	IFMR	IFCR
Slave to Master or Slave to Slave	DDRQ = 0 DTRQ = 0 HTRQ = 0	To enable an ID filter (Tx mode) for each DMA TX channel	• Identifier list mode • Identifier mask mode	DFL = payload size ID = address CCS = checksum DIR = 1 (TX)

The concept FSM to control the DMA Tx interface is given in the following figure. DMA TX FSM moves to Idle state if DMATXE[x] = 0 where x = IFMI – 1.The TCD setting (word transfer) is given in Table 54-13. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 54-13. TCD setting – Slave node – Tx mode

TCD Field	Value	Description
CITER[14:0]	1	Single iteration for the "major" loop
BITER[14:0]	1	Single iteration for the "major" loop
NBYTES[31:0]	4/8 = N	Data buffer is stuffed with dummy bytes if the length is not word aligned. BDRL + BDRM
SADDR[31:0]	—	RAM address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	BDRL address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain.
START	0	No SW request

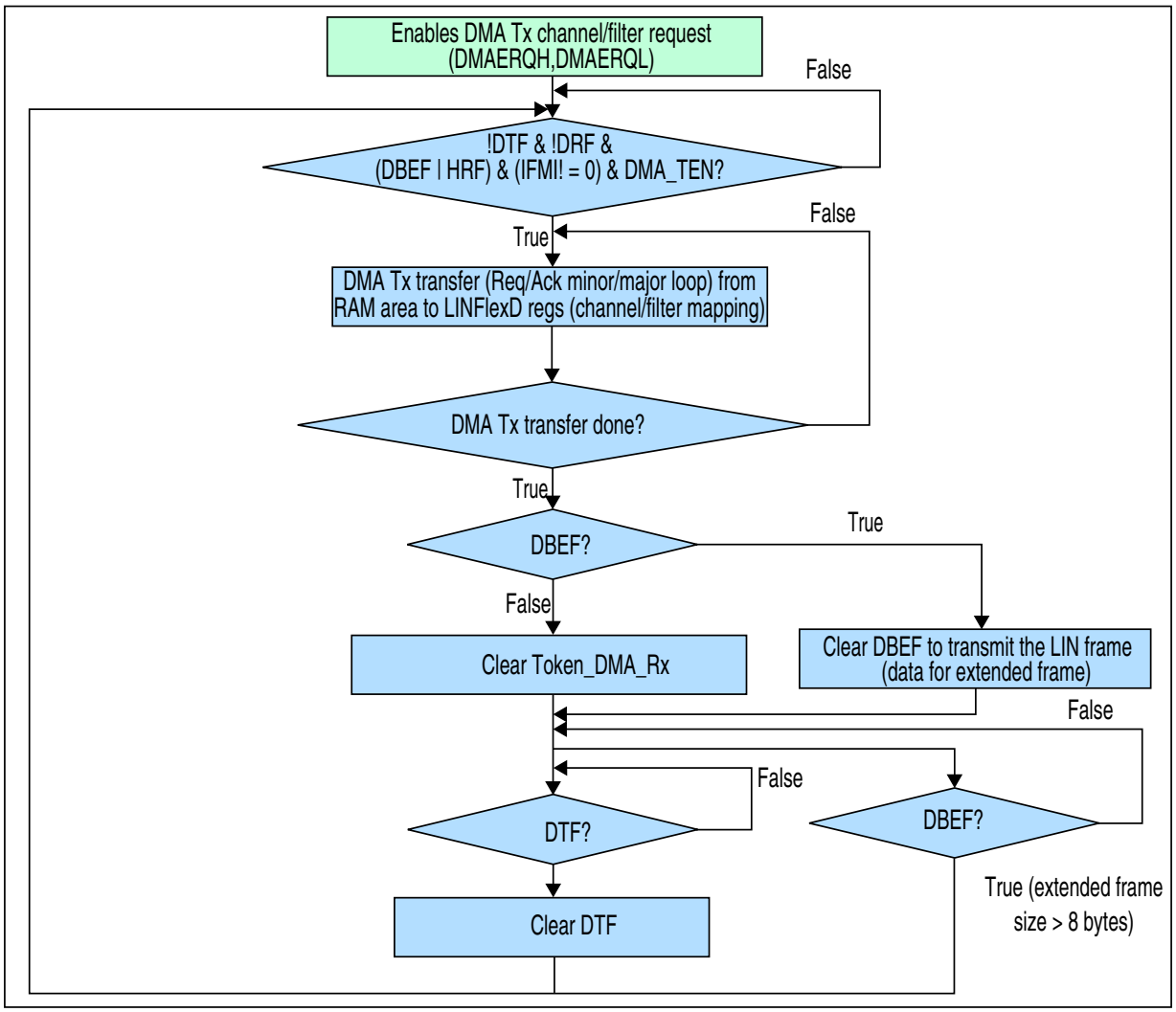


Figure 54-32. Slave node – DMA Tx FSM (concept scheme)

54.4.5.6 Slave node – RX mode

On a slave node in RX mode, the DMA interface requires a DMA RX channel for each ID filter programmed in RX mode. In case a single DMA RX channel is available, a single ID field filter must be programmed in RX mode. Each TCD controls a single frame, except for the extended frames (multiple TCDs). The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in the following figure.

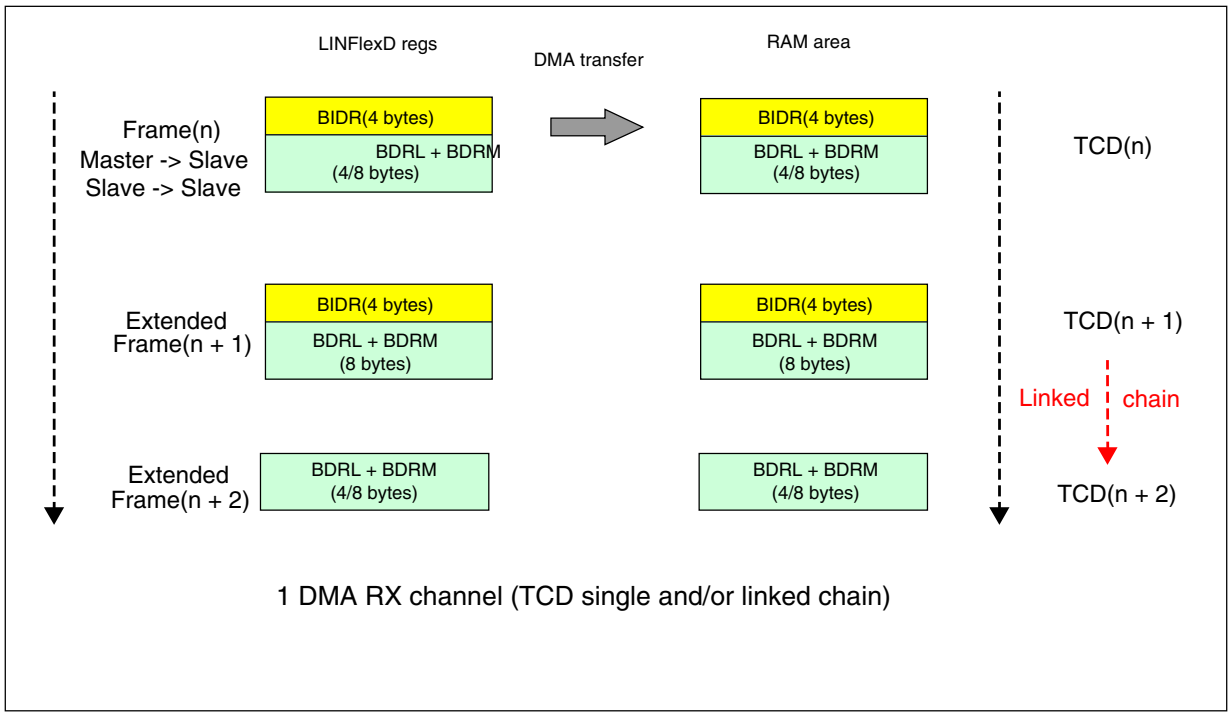


Figure 54-33. Slave node – RX memory map

The TCD chain of the DMA Rx channel on a slave node supports:

- Master to Slave: reception of the data field
- Slave to Slave: reception of the data field

The register setting of the LINCR2, IFER, IFMR, IFCR registers is given in Table 54-14.

Table 54-14. Slave node – Rx mode – Register setting

LIN frame	LINCR2	IFER	IFMR	IFCR
Master to Slave or Slave to Slave	DDRQ = 0 DTRQ = 0 HTRQ = 0	To enable an ID filter (Rx mode) for each DMA RX channel	• Identifier list mode • Identifier mask mode	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)

The concept FSM to control the DMA Rx interface is given in the following figure. DMA Rx FSM moves to Idle state if DMARXE[x] = 0 where x = IFMI – 1. The TCD setting (word transfer) is given in Table 54-15. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 54-15. TCD setting – Slave node – Rx mode

TCD Field	Value	Description
CITER[14:0]	1	Single iteration for the "major" loop
BITER[14:0]	1	Single iteration for the "major" loop
NBYTES[31:0]	[4] + 4/8 = N	Data buffer is stuffed with dummy bytes if the length is not word aligned. BIDR + BDRL + BDRM
SADDR[31:0]	—	BIDR address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	RAM address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain.
START	0	No SW request

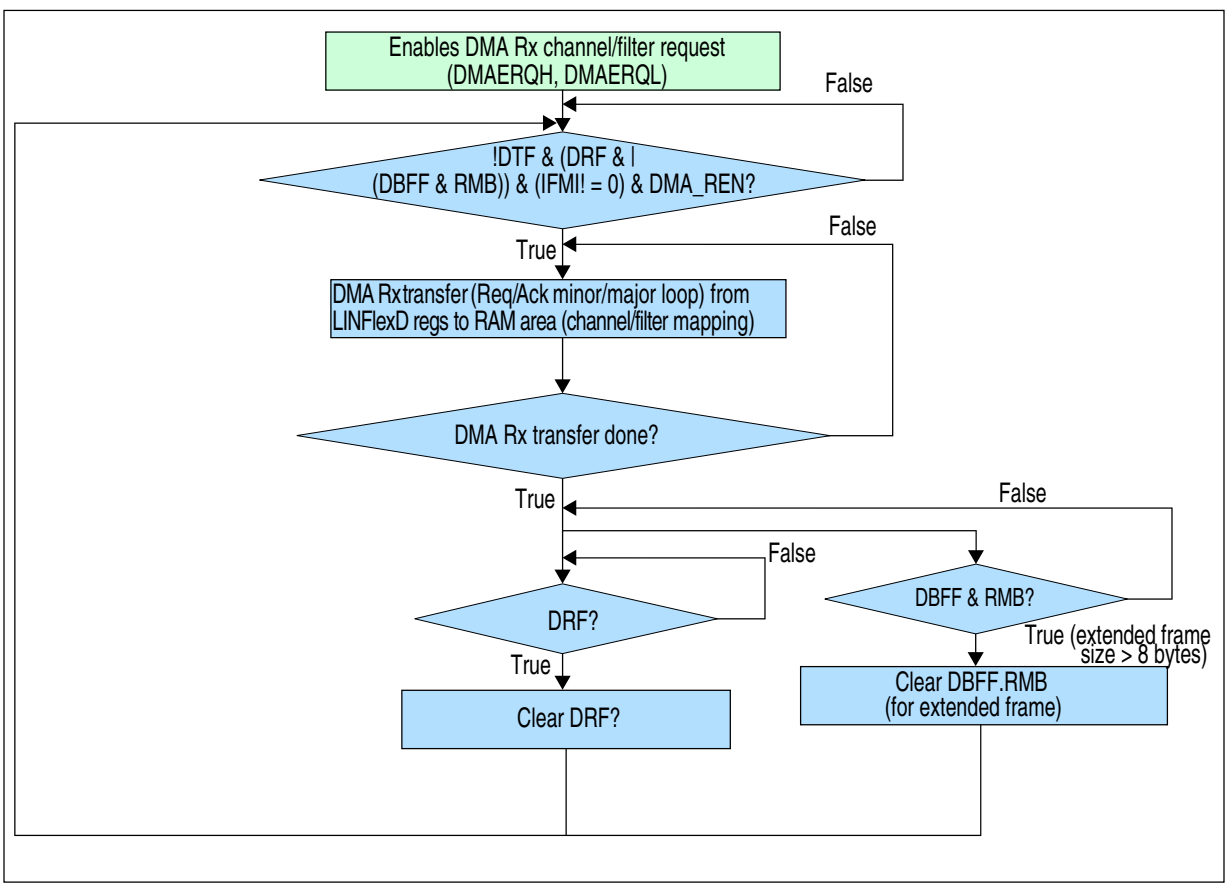


Figure 54-34. Slave node – DMA Rx FSM (concept scheme)

54.4.5.7 UART – TX mode

In UART TX mode, the DMA interface requires a DMA TX channel. A single TCD can control the transmission of an entire Tx buffer. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in Figure 54-35.

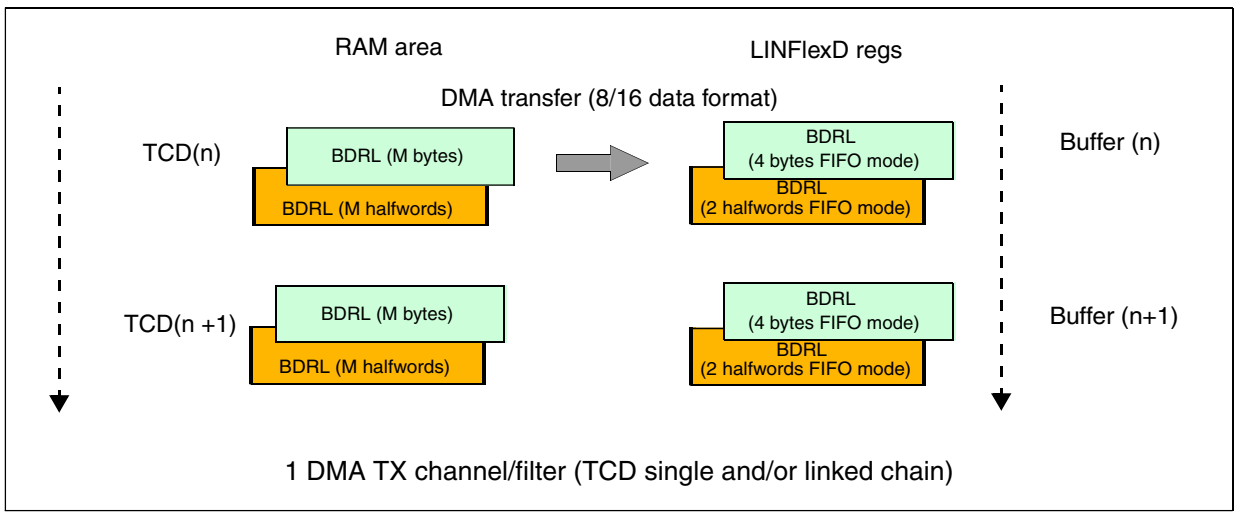


Figure 54-35. UART – TX memory map

The UART TX buffer must be configured in FIFO mode in order to:

- Allow the transfer of large data buffer by a single TCD
- Adsorb the latency, following a DMA request (due to the DMA arbitration), to move data from the RAM to the FIFO
- Use low priority DMA channels

The Tx FIFO size is:

- 4 bytes in 8 bit data format
- 2 halfwords in 16-bit data format

A DMA request is triggered by FIFO-not-full (TX) status signals.

The concept FSM to control the DMA Tx interface is given in Figure 54-36. DMA Tx FSM will move to Idle state if DMATXE[0] = 0.The TCD setting (typical case) is given in Table 54-16. All other TCD fields = 0. The minor loop transfers a single byte/halfword as soon as a free entry is available in the Tx FIFO.

Table 54-16. TCD setting – UART – Tx mode

TCD Field	Value		Description
	8 bits data	16 bits data	
CITER[14:0]	M		Multiple iterations for the "major" loop
BITER[14:0]	M		Multiple iterations for the "major" loop
NBYTES[31:0]	1	2	Minor loop transfer = 1 or 2 bytes
SADDR[31:0]	-		RAM address
SOFF[15:0]	1	2	Byte/Half-word increment
SSIZE[2:0]	0	1	Byte/Half-word transfer
SLAST[31:0]	–M	–M × 2	-
DADDR[31:0]	-		BDRL address
DOFF[15:0]	0		No increment (FIFO)
DSIZE[2:0]	0	1	Byte/Half-word transfer
DLAST_SGA[31:0]	0		No scatter/gather processing
INT_MAJ	0/1		Interrupt disabled/enabled
D_REQ	1		Only on the last TCD of the chain.
START	0		No SW request

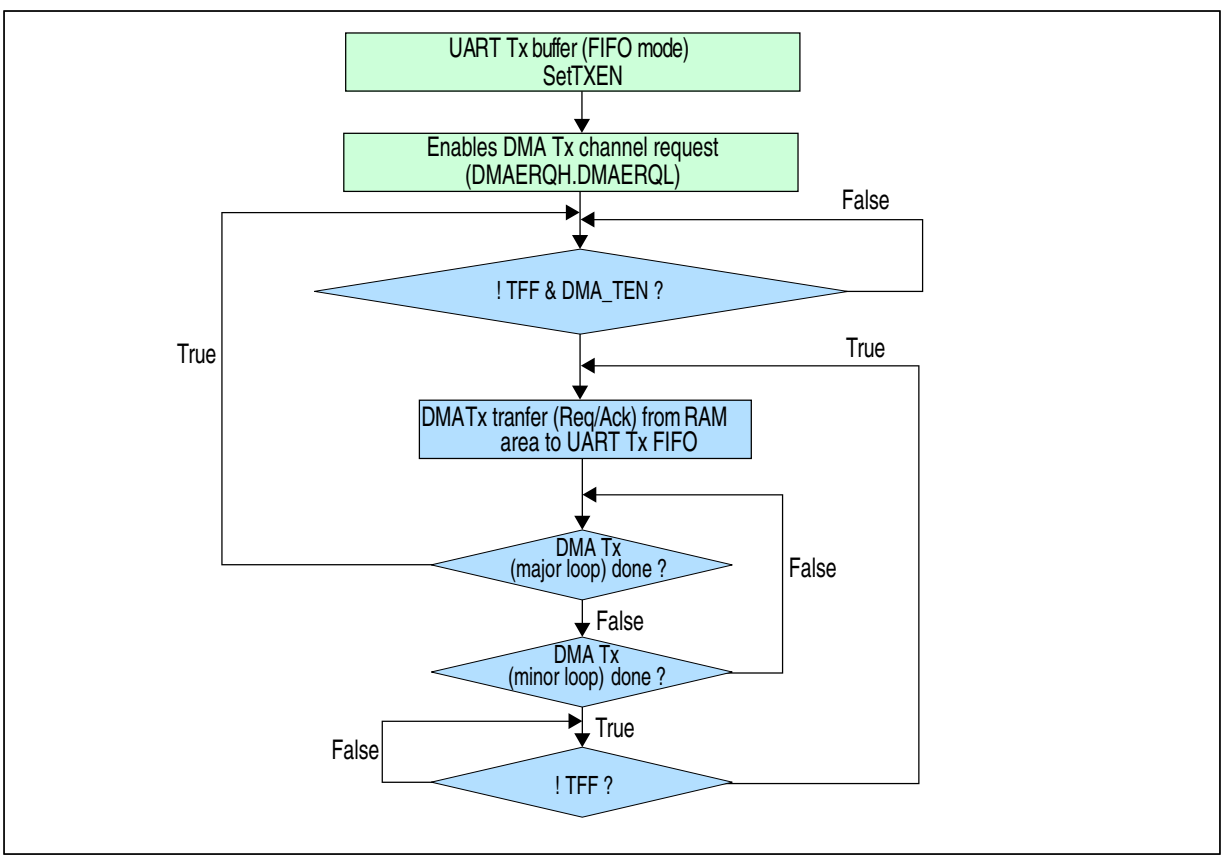


Figure 54-36. UART – DMA Tx FSM (concept scheme)

54.4.5.8 UART – RX mode

In UART RX mode, the DMA interface requires a DMA RX channel. A single TCD can control the reception of an entire Rx buffer. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in Table 54-17.

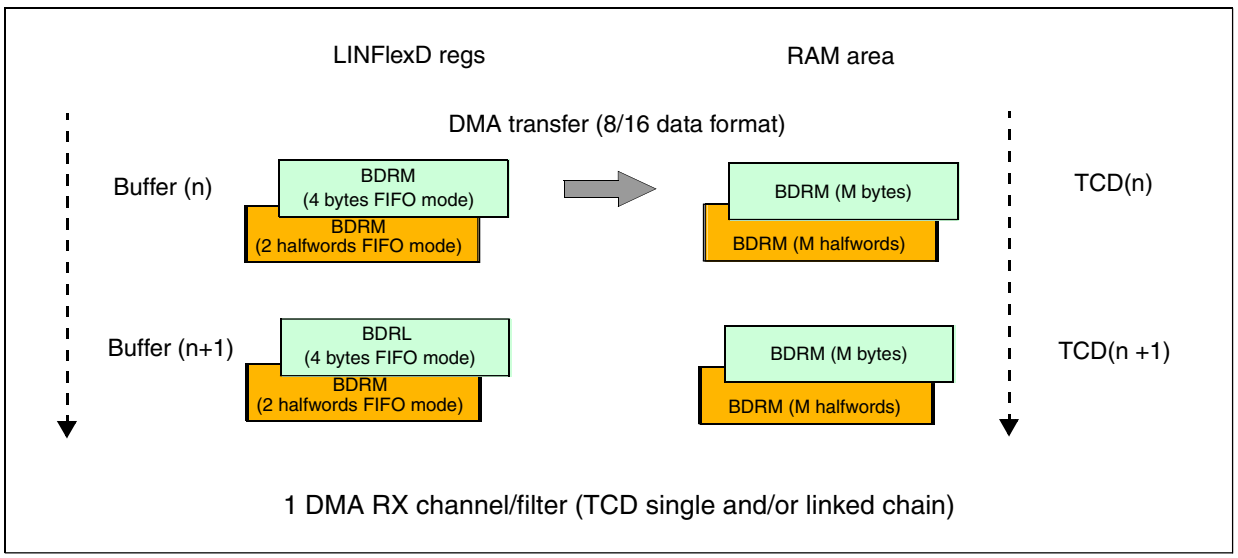


Figure 54-37. UART – RX memory map

The UART RX buffer must be configured in FIFO mode in order to:

- Allow the transfer of large data buffer by a single TCD
- Adsorb the latency, following a DMA request (due to the DMA arbitration), to move data from the FIFO to the RAM
- Use low priority DMA channels

The Rx FIFO size is:

- 4 bytes in 8-bit data format

This will sufficient because just one byte allows a reaction time of about 3.8 µs (at 2 Mbit/s) (~450 clock cycles at 120 MHz) before the transmission is affected. A DMA request is triggered by FIFO and not by empty (Rx) status signals.

The concept FSM to control the DMA Rx interface is given in Figure 54-38. DMA Rx FSM will move to Idle state if DMARXE[0] = 0.The TCD setting (typical case) is given in the next table. All other TCD fields equal zero. The minor loop transfers a single byte/ halfword as soon an entry is available in the Rx FIFO. A new software reset bit is

required that allows the LINFlexD FSMs to be reset in case this timeout state is reached or in any other case. The timeout counter can be re-written by software at any time to extend the timeout period.

Table 54-17. TCD setting – UART – Rx mode

TCD Field	Value		Description
	8 bits data	16 bits data	
CITER[14:0]	M		Multiple iterations for the "major" loop
BITER[14:0]	M		Multiple iterations for the "major" loop
NBYTES[31:0]	1	2	Minor loop transfer = 1 or 2 bytes
SADDR[31:0]	-		BDRM address
SOFF[15:0]	0		No increment (FIFO)
SSIZE[2:0]	0	1	Byte/Half-word transfer
SLAST[31:0]	0		
DADDR[31:0]	-		RAM address
DOFF[15:0]	1	2	Byte/Half-word increment
DSIZE[2:0]	0	1	Byte/Half-word transfer
DLAST_SGA[31:0]	–M	–M × 2	No scatter/gather processing
INT_MAJ	0/1		Interrupt disabled/enabled
D_REQ	1		Only on the last TCD of the chain.
START	0		No SW request

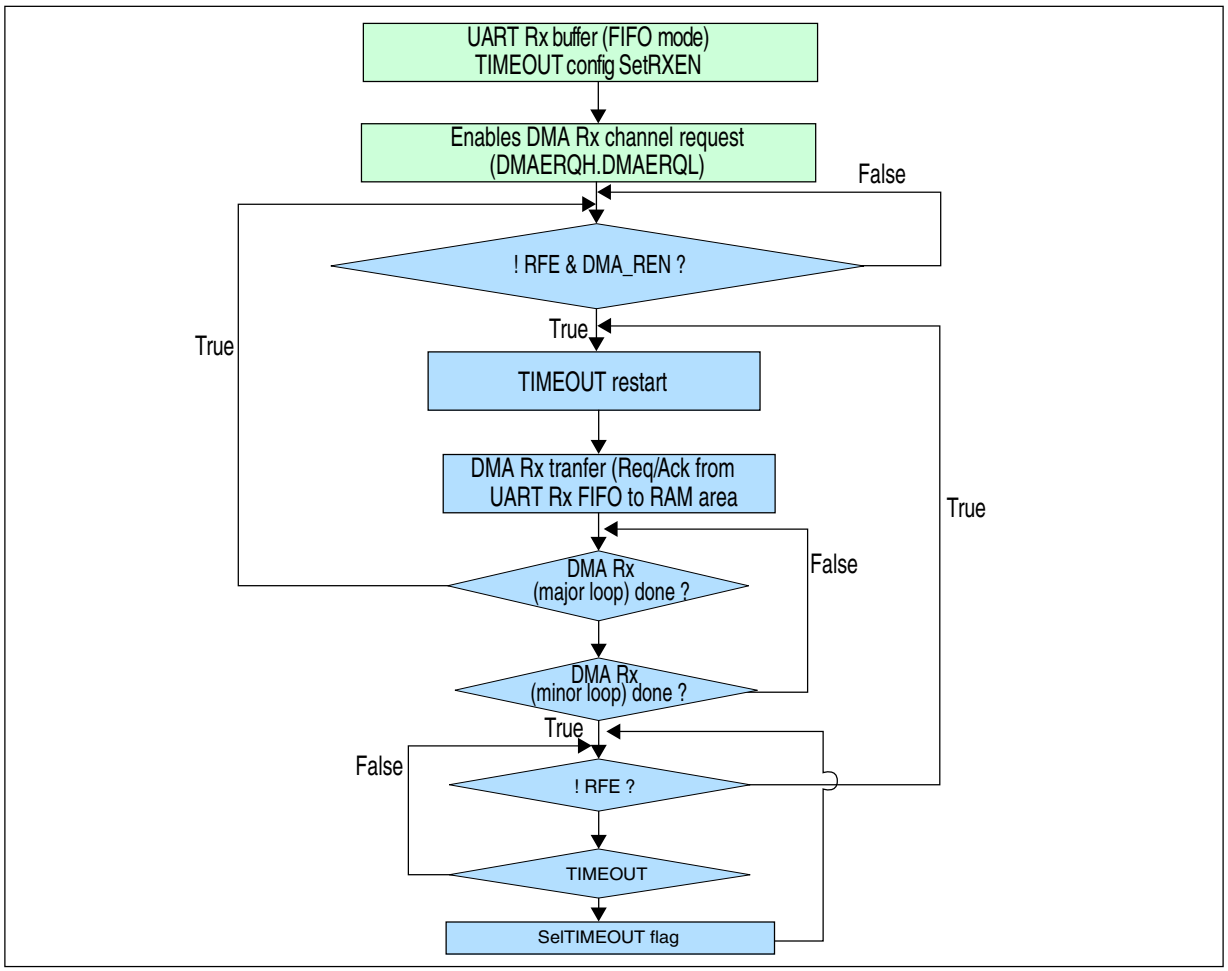


Figure 54-38. UART – DMA Rx FSM (concept scheme)

54.4.5.9 Use cases and limitations

- In LIN slave mode, the DMA capability can be used only if the ID filtering mode is activated. The number of ID filters enabled must be equal to the number of DMA channels enabled. The correspondence between channel number and ID filter is based on IFMI (identifier filter match index)
- In LIN master mode both the DMA channels (TX and RX) must be enabled in case the DMA capability is required
- In UART mode the DMA capability can be used only if the UART Tx/Rx buffers are configured as FIFOs
- DMA and CPU operating modes are mutually exclusive for the data/frame transfer on a UART or LIN node. Once a DMA transfer is finished the CPU can manage subsequent accesses

- Error management must always be executed via the CPU enabling the related error interrupt sources. DMA capability does not provide support for error management. Error management means checking status bits, handling IRQs, and potentially canceling DMA transfers
- The DMA programming model must be coherent with the TCD setting defined in this document
- When IPG_STOP is requested, SW has to first disable DMATXE/DMARXE channel registers, after the current minor loop finishes (indicated by the clearing of ACTIVE bit in DMA TCD register). This ensures that IPG_STOP_ACK is generated and IP enters STOP mode.

54.5 Memory map and register description

The following points should be considered for the below mentioned LINFlexD registers:

- Reset values are with-slave/without-slave (master only) format
- If not specified, then the bit can be configured for both master only and master/slave format; in other words, for both values of generic slave = 0 and slave = 1.

NOTE

In Master mode, the registers IFCR0-IFCR15 are not present and the corresponding absolute address of the below register changes as shown in the following table.

Table 54-19. Offsets of the register from offsets 8C to 9C

Register name	Master/slave mode	Master only mode
LINFlexD_GCR	0x8C	0x4C
LINFlexD_UARTPTO	0x90	0x50
LINFlexD_UARTCTO	0x94	0x54
LINFlexD_DMATXE	0x98	0x58
LINFlexD_DMARXE	0x9C	0x5C

NOTE

In Master Mode, read access to IFMI register and read/write access to IFER and IFMR registers would result in transfer error.

NOTE

Any access to A0 location will not generate a transfer error.

LINFlexD memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	LIN Control Register 1 (LINFlexD_LINCR1)	32	R/W	See section	54.5.1/2425
4	LIN Interrupt enable register (LINFlexD_LINIER)	32	R/W	0000_0000h	54.5.2/2428
8	LIN Status Register (LINFlexD_LINSR)	32	R/W	See section	54.5.3/2430
C	LIN Error Status Register (LINFlexD_LINESR)	32	w1c	0000_0000h	54.5.4/2434
10	UART Mode Control Register (LINFlexD_UARTCR)	32	R/W	0000_0000h	54.5.5/2435
14	UART Mode Status Register (LINFlexD_UARTSR)	32	R/W	0000_0000h	54.5.6/2441
18	LIN Time-Out Control Status Register (LINFlexD_LINTCSR)	32	R/W	0000_0200h	54.5.7/2444
1C	LIN Output Compare Register (LINFlexD_LINOCR)	32	R/W	0000_FFFFh	54.5.8/2445
20	LIN Time-Out Control Register (LINFlexD_LINTOCR)	32	R/W	See section	54.5.9/2446
24	LIN Fractional Baud Rate Register (LINFlexD_LINFBRR)	32	R/W	0000_0000h	54.5.10/ 2446
28	LIN Integer Baud Rate Register (LINFlexD_LINIBRR)	32	R/W	0000_0000h	54.5.11/ 2447
2C	LIN Checksum Field Register (LINFlexD_LINCFR)	32	R/W	0000_0000h	54.5.12/ 2448
30	LIN Control Register 2 (LINFlexD_LINCR2)	32	R/W	See section	54.5.13/ 2449
34	Buffer Identifier Register (LINFlexD_BIDR)	32	R/W	0000_0000h	54.5.14/ 2451
38	Buffer Data Register Least Significant (LINFlexD_BDRL)	32	R/W	0000_0000h	54.5.15/ 2452
3C	Buffer Data Register Most Significant (LINFlexD_BDRM)	32	R/W	0000_0000h	54.5.16/ 2453
40	Identifier Filter Enable Register (LINFlexD_IFER)	32	R/W	0000_0000h	54.5.17/ 2453
44	Identifier Filter Match Index (LINFlexD_IFMI)	32	R	0000_0000h	54.5.18/ 2454
48	Identifier Filter Mode Register (LINFlexD_IFMR)	32	R/W	0000_0000h	54.5.19/ 2454
4C	Identifier Filter Control Register (LINFlexD_IFCR0)	32	R/W	0000_0000h	54.5.20/ 2455
50	Identifier Filter Control Register (LINFlexD_IFCR1)	32	R/W	0000_0000h	54.5.20/ 2455
54	Identifier Filter Control Register (LINFlexD_IFCR2)	32	R/W	0000_0000h	54.5.20/ 2455
58	Identifier Filter Control Register (LINFlexD_IFCR3)	32	R/W	0000_0000h	54.5.20/ 2455
5C	Identifier Filter Control Register (LINFlexD_IFCR4)	32	R/W	0000_0000h	54.5.20/ 2455
60	Identifier Filter Control Register (LINFlexD_IFCR5)	32	R/W	0000_0000h	54.5.20/ 2455
64	Identifier Filter Control Register (LINFlexD_IFCR6)	32	R/W	0000_0000h	54.5.20/ 2455

Table continues on the next page...

LINFlexD memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
68	Identifier Filter Control Register (LINFlexD_IFCR7)	32	R/W	0000_0000h	54.5.20/ 2455
6C	Identifier Filter Control Register (LINFlexD_IFCR8)	32	R/W	0000_0000h	54.5.20/ 2455
70	Identifier Filter Control Register (LINFlexD_IFCR9)	32	R/W	0000_0000h	54.5.20/ 2455
74	Identifier Filter Control Register (LINFlexD_IFCR10)	32	R/W	0000_0000h	54.5.20/ 2455
78	Identifier Filter Control Register (LINFlexD_IFCR11)	32	R/W	0000_0000h	54.5.20/ 2455
7C	Identifier Filter Control Register (LINFlexD_IFCR12)	32	R/W	0000_0000h	54.5.20/ 2455
80	Identifier Filter Control Register (LINFlexD_IFCR13)	32	R/W	0000_0000h	54.5.20/ 2455
84	Identifier Filter Control Register (LINFlexD_IFCR14)	32	R/W	0000_0000h	54.5.20/ 2455
88	Identifier Filter Control Register (LINFlexD_IFCR15)	32	R/W	0000_0000h	54.5.20/ 2455
8C	Global Control Register (LINFlexD_GCR)	32	R/W	0000_0000h	54.5.21/ 2456
90	UART Preset Timeout Register (LINFlexD_UARTPTO)	32	R/W	0000_0FFFh	54.5.22/ 2458
94	UART Current Timeout Register (LINFlexD_UARTCTO)	32	R	0000_0000h	54.5.23/ 2459
98	DMA Tx Enable Register (LINFlexD_DMATXE)	32	R/W	0000_0000h	54.5.24/ 2460
9C	DMA Rx Enable Register (LINFlexD_DMARXE)	32	R/W	0000_0000h	54.5.25/ 2460

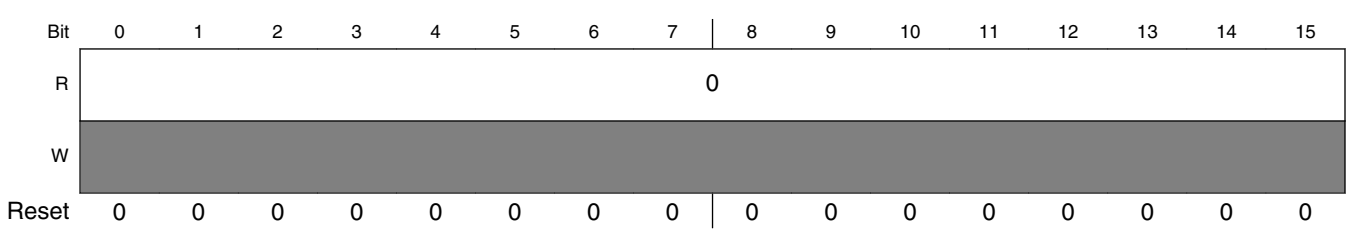
54.5.1 LIN Control Register 1 (LINFlexD_LINCR1)

LINCR1 consists of control bits used to configure features of the LINFlexD.

NOTE

When accessing the LINCR1 register, each reserved bit should be written to its original reset value.

Address: 0h base + 0h offset = 0h



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CCD	CFD	LASE	AUTOWU	MBL				BF	0	LBKM	MME	SSBL	RBLM	SLEEP	INIT
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	*	0	0	1	0

* Notes:
- MME field: If slave = 0, then MME bit is 1, else MME bit is 0

LINFlexD_LINCR1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 CCD	Checksum Calculation disable This bit can be written during Initialization mode only. It is read-only in Normal mode. 0 Checksum calculation is done by hardware. When this bit is reset the LINCFR register is read only. 1 Checksum calculation is disabled. When this bit is set the LINCFR register is read/write. User can program this register to sent a software calculated checksum/CRC (provided CFD is reset).
17 CFD	Checksum field disable This bit can be configured during Initialization mode only. It is read-only in Normal mode. 0 Checksum field is sent after the required number of data bytes are sent 1 No checksum field is sent in the frame
18 LASE	LIN Autosynchronization Enable This bit can be programmed in Initialization mode only. It is read-only in Normal mode. (Note: If generic auto_sync = 0, then this bit will always read a 0). 0 Autosynchronization disabled 1 Autosynchronization enabled
19 AUTOWU	Auto Wakeup This bit can be configured during Initialization mode only. This bit is utilized in UART mode also. 0 Sleep bit is cleared by software only 1 Sleep bit gets cleared by hardware whenever WUF bit of LINSR is set
20–23 MBL	Master Break Length These bits choose the length of the Sync break to be generated by the master. These bits can be programmed in Initialization mode only. They are read-only in Normal mode. 0000 10-bit break length 0001 11-bit break length 0010 12-bit break length 0011 13-bit break length 0100 14-bit break length 0101 15-bit break length 0110 16-bit break length

Table continues on the next page...

LINFlexD_LINCR1 field descriptions (continued)

Field	Description
	0111 17-bit break length 1000 18-bit break length 1001 19-bit break length 1010 20-bit break length 1011 21-bit break length 1100 22-bit break length 1101 23-bit break length 1110 36-bit (cooling) break length 1111 50-bit break length
24 BF	By-pass filter This bit can be programmed during Initialization mode only. **NOTE:** If generic slave = 0 or no_of_filters = 0, then this bit will always read a 1, and cannot be programmed. 0 No IRQ if ID does not match any filter 1 A RX IRQ is generated on ID not matching any filter
25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 LBKM	Loop Back mode Refer to "Loop back mode" in Test mode . Note that this bit can be programmed only in Initialization mode. This bit is utilized in UART mode also. 0 Loop Back Mode disabled 1 Loop Back mode enabled
27 MME	Master mode enable This bit can be programmed in Initialization mode only. It is read-only in Normal mode. **NOTE:** If generic slave = 0, then this bit will read a '1' always, and cannot be programmed. 0 Slave Mode 1 Master Mode
28 SSBL	Slave Mode Sync Break Length This bit can be programmed in Initialization mode only. It is read only in Normal mode. 0 11 bit break length 1 10 bit break length
29 RBLM	Receiver Buffer Locked mode This bit can be programmed in Initialization mode only. It is read-only in Normal mode. This bit is utilized in UART mode also. 0 Receiver Buffer not locked, next incoming message will overwrite the old one 1 Receiver buffer locked against overrun. Once the buffer is full the next incoming message will be discarded if buffer is not released — in other words, RMB is not reset by software.
30 SLEEP	Sleep Mode Request This bit is set by software to request LINFlexD to enter Sleep mode. This bit is cleared by software or hardware (if AUTOWU bit in LINCR1 and WUF bit in LINSR are set) to exit sleep mode. This bit is utilized in UART mode also.

Table continues on the next page...

LINFlexD_LINCR1 field descriptions (continued)

Field	Description
31 INIT	Initialization Mode Request The software sets this bit to switch the hardware into Initialization mode. On clearing this bit (and if SLEEP bit is also zero) LINFlexD enters normal mode. This bit is utilized in UART mode also.

54.5.2 LIN Interrupt enable register (LINFlexD_LINIER)

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZIE	OCIE	BEIE	CEIE	HEIE	0		FEIE	BOIE	LSIE	WUIE	DBFIE	DBEIETOIE	DRIE	DTIE	HRIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_LINIER field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SZIE	Stuck at zero Interrupt Enable An interrupt is generated if this bit is set and the Stuck at Zero Flag (SZF) in LINESR or UARTSR is set. 0 No interrupt 1 Interrupt enabled
17 OCIE	Output Compare Interrupt Enable 0 No interrupt 1 Interrupt generated when OCF bit in LINESR or UARTSR is set
18 BEIE	Bit Error Interrupt Enable 0 No interrupt 1 Interrupt generated when BEF bit in LINESR is set
19 CEIE	Checksum Error Interrupt Enable An interrupt is generated if this bit is set and the Checksum Error Flag (CEF) is set in LINESR.

Table continues on the next page...

LINFlexD_LINIER field descriptions (continued)

Field	Description
	0 No interrupt 1 Interrupt enabled
20 HEIE	Header Error Interrupt Enable An interrupt is generated when this bit is set and either of the following flags are set SFEF, SDEF, IDPEF in LINESR are set. **NOTE:** If generic slave = 0, then this bit will always read a 0 and cannot be programmed. 0 No interrupt 1 Interrupt enabled
21–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 FEIE	Frame Error Interrupt Enable 0 No interrupt 1 Interrupt generated if Frame Error Flag (FEF) bit is set in LINESR or UARTSR
24 BOIE	Buffer Overrun Error Interrupt Enable An interrupt is generated if this bit is set and the Buffer Overrun Flag (BOF) is set in LINESR or UARTSR. 0 No interrupt 1 Interrupt enabled
25 LSIE	LIN state Interrupt enable Interrupt is generated only when entering the above fields. This interrupt can mainly be used for debugging purposes. The interrupt has no status flag. 0 No interrupt 1 Interrupt generated on entering the following states: Sync Del, Sync Field, Identifier field, Checksum
26 WUIE	Wakeup interrupt enable If this bit is set and the WUF in LINSR or UARTSR is set then an interrupt is generated. 0 No interrupt 1 Interrupt enabled
27 DBFIE	Data Buffer Full Interrupt enable An interrupt is generated if this bit is set and the DBFF bit in LINSR is also set. 0 No interrupt 1 Interrupt enabled
28 DBEIETOIE	Data Buffer Empty Interrupt enable/Timeout Interrupt Enable An interrupt is generated if this bit is set and LINSR[DBEF] status bit is set (in LIN mode) or if this bit is set and UARTSR[TO] status bit is set (in UART mode). 0 No interrupt 1 Interrupt enabled
29 DRIE	Data Reception complete Interrupt enable An interrupt is generated when this bit is set and Data Received flag (DRF) in LINSR or UARTSR is set. 0 No interrupt 1 Interrupt enabled

Table continues on the next page...

LINFlexD_LINIER field descriptions (continued)

Field	Description
30 DTIE	Data Transmitted Interrupt enable An interrupt is generated when this bit is set and Data Transmitted flag (DTF) in LINSR or UARTSR is set. 0 No interrupt 1 Interrupt enabled
31 HRIE	Header Received Interrupt An interrupt is generated when this bit is set and the Header Received flag (HRF) in LINSR is set. **NOTE:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed. 0 No interrupt 1 Interrupt enabled

54.5.3 LIN Status Register (LINFlexD_LINSR)

This register consists of status bits indicating the state of the LINFlexD hardware.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												AUTOSYNC_COMP	RDC		
W													w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LINS				0		RMB	DRBNE	RXbusy	RDI	WUF	DBFF	DBEF	DRF	DTF	HRF
W	1						w1c	w1c			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	*	0	0	0	0	0	0

* Notes:
- RDI field: Reset value of RDI reflects the RX pin state

LINFlexD_LINSR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 AUTOSYNC_COMP	AUTOSYNC_COMP This bit is set after autosynchronization is complete and when the LASE bit in LINCR1 register is enabled. Only after this bit is set, the contents of LINIBRR and LINFBRR registers can be read in autosynchronization mode. **NOTE:** If autosynchronization is disabled, this bit is reserved.
13–15 RDC	Receive Data Byte Count RDC contains the number of entries (bytes) in the Receive data buffer in LIN mode. RDCx is a read-only field available for debug purposes. 000 1 byte 001 2 bytes 010 3 bytes 011 4 bytes 100 5 bytes 101 6 bytes 110 7 bytes 111 8 bytes

Table continues on the next page...

LINFlexD_LINSR field descriptions (continued)

Field	Description
16–19 LINS	LIN state In UART mode Idle, Init, Sleep, and Data Transmission/Reception states are flagged by the LIN status bits. **NOTE:** The value of this bit field doesn't change by any write operation to it. Writing 0xF to this bit field clears the Rx interrupt, only when set due to the LIN state event. 0000 Sleep mode: LINFlexD is in Sleep mode, to save power consumption. 0001 Init mode: This mode is entered when: SLEEP bit and INIT bit are reset by software; Wakeup pulse has been received on RX pin (AUTOWU set); Previous frame transmission/reception has been completed/aborted 0010 Idle mode: This mode is entered when: SLEEP bit and INIT bit are reset by software; Wakeup pulse has been received on RX pin (AUTOWU set); Previous frame transmission/reception has been completed/aborted 0011 Sync break: In slave mode, a falling edge followed by a dominant state has been detected. Receiving sync break. In slave mode, a falling edge followed by a dominant state has been detected. Receiving sync break. In master mode, sync break transmission ongoing. Note: In slave mode upon any error LIN state could be either Idle or Rec Break, depending on last bit detected on LIN_RX. If last bit detected is dominant then Rec_Break, otherwise Idle. 0100 Sync Del: In Slave mode, valid Sync break has been detected (10 bit or 11 bit). Waiting for a rising edge. In Master mode, Sync break transmission has been completed, sync delimiter transmission is ongoing. 0101 Sync Field: In Slave mode, a valid sync Del has been detected (recessive state for at least one bit time). Receiving sync field. In Master mode, sync field transmission ongoing. 0110 Identifier Field: In Slave mode, a valid sync field has been received. Receiving ID field. In Master mode, identifier transmission is ongoing. 0111 Header Reception/Transmission: In Slave mode, a valid header has been received and Identifier field is available in the BIDR. In Master mode, header transmitted. 1000 Data Reception/Data Transmission: In Receiver mode, reception ongoing. In Transmitter mode, response transmission ongoing. 1001 Checksum: Data transmission/reception completed, checksum transmission/reception ongoing.
20–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 RMB	Release Message Buffer 0 Buffer data is free. Reset by hardware in when in Initialization mode 1 Buffer data ready to be read by software. This bit should be cleared by software after reading the data received in the buffer.
23 DRBNE	Data Reception Buffer Not Empty Flag This bit is set by hardware as soon as the first byte of response has been received and stored in BDRL (when there is at least one data byte in reception buffer). Software should clear it after reading all the buffers. This bit is also reset by hardware in Initialization mode. This flag could be checked by software in case of a response timeout event.
24 RXbusy	Receiver Busy flag In Slave mode after header reception, if DIR bit is reset and reception starts, then this bit is set. In this case user cannot set the DTRQ bit. 0 Receiver is idle 1 Reception ongoing

Table continues on the next page...

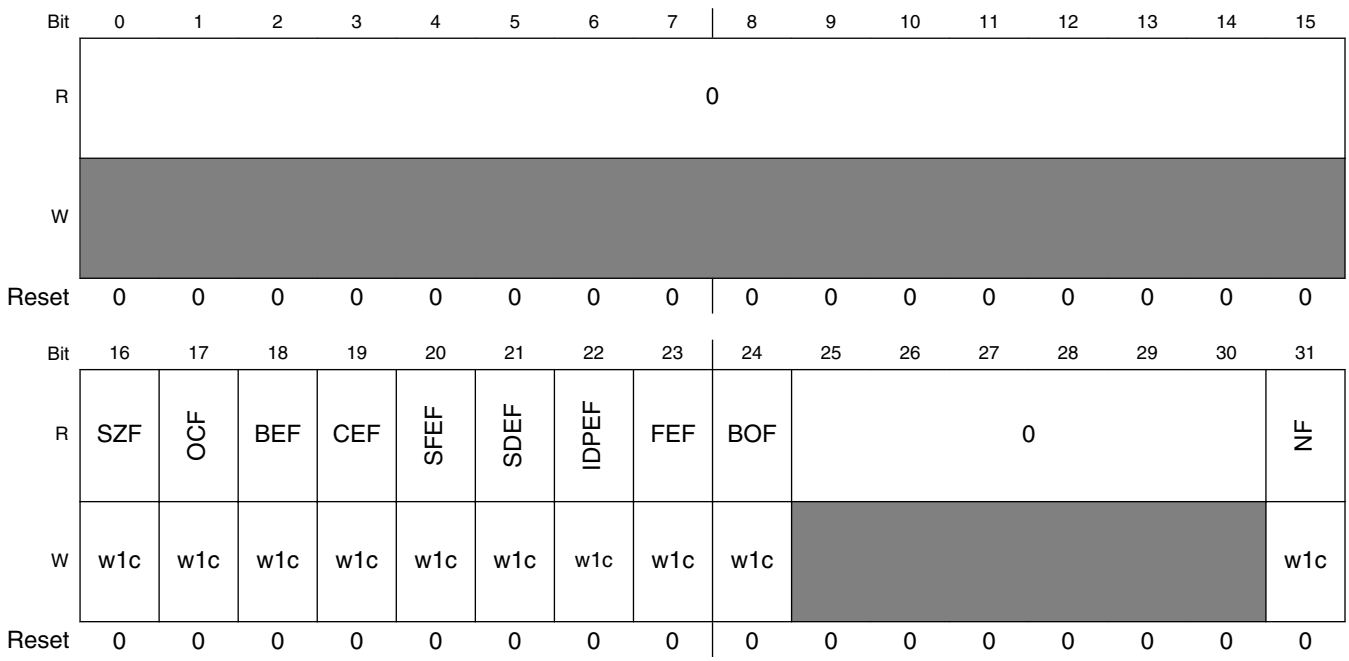
LINFlexD_LINSR field descriptions (continued)

Field	Description
25 RDI	LIN Receive signal This bit reflects the current status of the Rx pin. **NOTE:** After reset is released, RDI reflects the actual value of Rx pin.
26 WUF	Wakeup flag This bit is set by hardware when a falling edge is detected on the Rx pin. 1. When slave is in Sleep mode 2. When master is in Sleep mode or Idle mode It can be cleared by software. It gets reset by hardware in Initialization mode.
27 DBFF	Data Buffer full flag This bit is set by hardware when the buffer is full (8 bytes received) and DFL > 7. It should be cleared by software. This bit is also reset by hardware in Initialization mode.
28 DBEF	Data Buffer empty flag This bit is set by hardware when the buffer is empty (8 bytes have been transmitted) and DFL > 7. It should be cleared by software after data buffer is refilled by software. Transmission of next 8 bytes will not start unless this bit is reset by software. This bit is also reset by hardware in Initialization mode.
29 DRF	Data Reception Completed flag This bit is set by hardware and indicates that data reception has been completed. This flag should be cleared by software. This bit is also reset by hardware in Initialization mode. **NOTE:** In case framing error or checksum error occurs then this flag is not set.
30 DTF	Data Transmission Completed flag This bit is set by hardware and indicates that data transmission is completed. This flag should be cleared by software. This bit is also reset by hardware in Initialization mode. **NOTE:** For LIN mode, in case a bit error occurs (and IOBE is reset) then this flag is not set.
31 HRF	Header Received flag This bit is set when the header reception is completed. It should be cleared by software. This bit is set only when: a) All filters are inactive and Bypass filter (BF) bit is set b) No match in any filter and Bypass filter (BF) bit is set c) TX filter match At end of frame or frame aborted, if HRF is still set, it gets reset by hardware. This bit is also reset by hardware in Initialization mode. **NOTE:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.

54.5.4 LIN Error Status Register (LINFlexD_LINESR)

Refer to Errors for detailed description of all errors.

Address: 0h base + Ch offset = Ch



LINFlexD_LINESR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SZF	Stuck at Zero flag This bit is set when there is a stuck-at-zero timeout error. It should be reset by software.
17 OCF	Output Compare Flag 0: No output compare event occurred 1: In master mode, LINESR[OCF] flag is set when counter LINTCSR[CNT] has matched the content of LINOCR[OC2]. In slave mode, LINESR[OCF] is set when the content of the counter LINTCSR[CNT] has matched the content of LINOCR[OC1] or LINOCR[OC2]. This bit should be cleared by software. If this bit is set, MODE bit in LINTCSR is cleared, and the IOT bit of LINTCSR is set, then LINFlexD moves to Idle state. If the MODE bit in LINTCSR is clear, then OCF gets reset by hardware in Initialization mode; if the MODE bit is set, then OCF maintains its status irrespective of the LIN state.
18 BEF	Bit Error flag This bit is set by hardware when there is a bit error. It should be cleared by software. Reset by hardware in Initialization mode.

Table continues on the next page...

LINFlexD_LINESR field descriptions (continued)

Field	Description
19 CEF	Checksum Error flag This bit is set by hardware if the received checksum does not match the hardware-calculated checksum. It should be cleared by software. This error will never occur if CCD or CFD bit of LINCR1 is set. Reset by hardware in Initialization mode.
20 SFEF	Sync Field Error flag This bit is set by hardware when the received Sync Field is inconsistent. It should be cleared by software. Reset by hardware in Initialization mode. If generic slave = 0, then this bit will always read a 0, and cannot be programmed.
21 SDEF	Sync Delimiter Error flag This bit is set by hardware when the delimiter is too short (in other words, less than one bit time). It should be cleared by software. Reset by hardware in Initialization mode. **NOTE:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.
22 IDPEF	ID Parity Error flag This bit is set by hardware when there is an error in the ID parity. It should be cleared by software. **NOTE:** Header Error Interrupt is triggered for SFEF or SDEF or IDPEF flag is set and HEIE is set. If generic slave = 0, then this bit will always read a 0, and cannot be programmed. For generic slave = 1,this bit is reset by hardware in Initialization mode.
23 FEF	Framing Error flag This bit is set by hardware when there is a framing error (invalid stop bit). It should be cleared by software. Reset by hardware in initialization mode.
24 BOF	Buffer overrun flag This bit is set by hardware when there is a new byte received and RMB bit is not cleared. It can be cleared by software. Reset by hardware in initialization mode.
25–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 NF	Noise flag This bit is set by hardware when noise is detected in the received character. It should be cleared by software. Reset by hardware in Initialization mode.

54.5.5 UART Mode Control Register (LINFlexD_UARTCR)

NOTE

When accessing the UARTCR register, reserved bits should always be written to zero.

NOTE

- The LINFlexD module in UART mode does not support communication with Special Word Length (UARTCR[WLS] = 1) in buffer mode.
- The LINFlexD module in UART mode supports communication with Special Word Length (UARTCR[WLS] = 1) in FIFO mode only with UARTCR[WL1, WL0] = 1,1.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MIS	CSP			OSR				ROSE	NEF			DTU_PCETX	SBUR		WLS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TDFL_TFC			RDFL_RFC			RFBM	TFBM	WL1	PC1	RxEn	TxEn	PC0	PCE	WL0	UART
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_UARTCR field descriptions

Field	Description
0 MIS	Monitor Idle State 0 UARTCTO monitors the number of bits to be received. 1 UARTCTO monitors the idle state of the reception line.
1–3 CSP	Configurable Sample Point (i) These bits will decide the sample point during reduced over sampling. CSP can take the following range of values for a certain over sampling rate:

OSR	CSP
4	2,3
5	2, 3, 4
6	3, 4, 5
8	NA

4–7 OSR	Over Sampling Rate These bits are programmable by the user to configure the number of samples taken for a bit when reduced over sampling is enabled.

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

Field	Description
	Allowed values are: 4, 5, 6 and 84 and 8 .
8 ROSE	Reduced Over Sampling Enable 0 Each bit is over sampled sixteen times. 1 OSR bits decide the over sampling rate.
9–11 NEF	Number of expected frame These bits are used to configure the number of expected frames in UART reception mode. If the DTU bit is set, then the UART timeout counter will be reset after the configured number of frames have been received.
12 DTU_PCETX	Disable Timeout in UART mode This bit can be programmed in Initialization mode only when the UART bit is set. There is an internal counter that gives the number of frames received. **In Buffer mode:** This timer gets reset whenever the number of bytes received (rec_byte_cnt) is equal to RDFL. At this point the DRF bit is also set and num_of_frames (number of frames received), rec_byte_cnt are reset. When num_of_frames equals NEF, Disable Timeout is generated. But this clears num_of_frames also. Thus the timer restarts again. The value of counter is 0 when it restarts. **In FIFO mode:** When num_of_frames equals NEF, Disable Timeout is generated which clears num_of_frames and causes the timer to restart again. The value of counter is 0 when it restarts. Notes: • Disable Timeout causes only the Timer to restart which enables Timeout again. • Timer reset means it resets and starts counting again and the Timeout is enabled. 0 Timeout has to be handled by software 1 Timeout in UART mode is disabled after the configured number of data frames are received
13–14 SBUR	Stop bits in UART reception mode When the UART is used for transmission and reception we have to set the same number of stop bits in GCR and SBUR. When the UART is used only as receiver, it is enough to set SBUR bits only. This bit can be programmed in the initialization mode only, when the UART bit is set. 00 1 stop bit 01 2 stop bits 10 3 stop bits 11 reserved
15 WLS	Special Word Length in UART mode This bit can be programmed in initialization mode only when the UART bit is set. If this bit is set, setting WL and PCE bits has no effect in UART reception although parity check is enabled by default in this mode and PC0/1 bits can be used to select the parity control. This bit is given the highest priority in UART reception mode. 0 This bit is disabled. 1 This bit enables 12-bit + parity bit in reception (UART mode) for MSC (Micro Second Channel upstream frame) support.
16–18 TDFL_TFC	Transmitter Data Field Length/TX FIFO Counter

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

Field	Description
	TDFL defines the number of bytes to be transmitted in UART buffer mode (TFBM = 0). TDFL is a read/write field. Bit 15 is reserved and not implemented. When the UART data length is configured as halfword (WL = 10 or WL = 11), only the configurations TDFL = x01 or TDFL = x11 are allowed. x00 : 1 byte x01 : 2 bytes x10 : 3 bytes x11 : 4 bytes Register bit is a read/write field. TFC contains the number of entries (bytes) of the Tx FIFO in UART FIFO mode (TFBM = 1). TFCx is a read-only field available for debug purposes. 000 : empty 001 : 1 byte 010 : 2 bytes 011 : 3 bytes 100 : 4 bytes Others : reserved TDFLTFC can be programmed and are significative only when the UART bit is set. Register bit can be read in any mode, written only in Initialization mode. **NOTE:** When a debugger is connected, this counter will decrement if TxFIFO is being read through the debugger. **NOTE:** In case of data path is functioning normally, RFC/TFC counters will be cleared internally by HW.
19–21 RDFL_RFC	Reception Data Field Length /RX FIFO Counter RDFL defines the number of bytes to be received in UART buffer mode (RFBM = 0). RDFL is a read/write field. Bit 19 is reserved and not implemented. When the UART data length is configured as halfword (WL = 10 or WL = 11), only the configurations RDFL = x01 or RDFL = x11 are allowed. x00 : 1 byte x01 : 2 bytes x10 : 3 bytes x11 : 4 bytes RFC contains the number of entries (bytes) of the Rx FIFO in UART FIFO mode (RFBM = 1). RFCx is a read-only field available for debug purposes. 000 : Empty 001 : 1 byte 010 : 2 bytes/1.5 byte in case of WLS bit setting 011 : 3 bytes 100 : 4 bytes/2x1.5 byte in case of WLS bit setting Others - Reserved RDFLRFC can be programmed and are significative only when the UART bit is set.

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>NOTE: In buffer mode, RDFL should be programmed to be greater than or equal to NEF (number of expected frames). In FIFO mode, there is no such constraint.
NOTE: When a debugger is connected, this counter will decrement if RxFIFO is being read through the debugger.
NOTE: In case of data path is functioning normally, RFC/TFC counters will be cleared internally by HW.</td></tr>
<tr><td>22
RFBM</td><td>RFBM Rx Fifo/Buffer mode
This bit can be programmed in Initialization mode, only when the UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
0 Rx Buffer mode enabled
1 Rx Fifo mode enabled (mandatory in DMA Rx mode)</td></tr>
<tr><td>23
TFBM</td><td>Tx Fifo/Buffer mode
This bit can be programmed in initialization mode, only when the UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
0 Tx Buffer mode enabled
1 Tx Fifo mode enabled (mandatory in DMA Tx mode)</td></tr>
<tr><td>24
WL1</td><td>Word Length in UART mode
This bit can be programmed in Initialization mode only, when the UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
<table>
<tr><th>WL1</th><th>WL0</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>7 bits data + parity</td></tr>
<tr><td>0</td><td>1</td><td>8 bits data when PCE = 0 or 8 bits data + parity when PCE = 1</td></tr>
<tr><td>1</td><td>0</td><td>15 bits data + parity</td></tr>
<tr><td>1</td><td>1</td><td>16 bits data when PCE = 0 or 16 bits data + parity when PCE = 1</td></tr>
</table></td></tr>
<tr><td>25
PC1</td><td>Parity Control
This bit can be programmed in Initialization mode, only when UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
<table>
<tr><th>PC1</th><th>PC0</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>Parity sent is Even</td></tr>
<tr><td>0</td><td>1</td><td>Parity sent is Odd</td></tr>
<tr><td>1</td><td>0</td><td>A logical 0 is always transmitted/ checked as parity bit</td></tr>
<tr><td>1</td><td>1</td><td>A logical 1 is always transmitted/ checked as parity bit</td></tr>
</table></td></tr>
</table>

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>26
RxEn</td><td>Receiver Enable
This bit can be programmed only when the UART bit is set.
0 Receiver disabled
1 Receiver enabled</td></tr>
<tr><td>27
TxEn</td><td>Transmitter Enable
This bit can be programmed only when UART bit is set.
0 Transmitter disabled
1 Transmitter enabled, transmission starts only when this bit is set and Data byte 0 (DATA0) is programmed</td></tr>
<tr><td>28
PC0</td><td>Parity Control
This bit can be programmed in Initialization mode, only when UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
<table><tr><th>PC1</th><th>PC0</th><th>Description</th></tr><tr><td>0</td><td>0</td><td>Parity sent is Even</td></tr><tr><td>0</td><td>1</td><td>Parity sent is Odd</td></tr><tr><td>1</td><td>0</td><td>A logical 0 is always transmitted/ checked as parity bit</td></tr><tr><td>1</td><td>1</td><td>A logical 1 is always transmitted/ checked as parity bit</td></tr></table></td></tr>
<tr><td>29
PCE</td><td>Parity Control Enable
This bit can be programmed in Initialization mode only, when the UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
0 Parity transmit/check Disable
1 Parity transmit/check Enable</td></tr>
<tr><td>30
WL0</td><td>Word Length in UART mode
This bit can be programmed in Initialization mode only, when UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
<table><tr><th>WL1</th><th>WL0</th><th>Description</th></tr><tr><td>0</td><td>0</td><td>7 bits data + parity</td></tr><tr><td>0</td><td>1</td><td>8 bits data when PCE = 0 or 8 bits data + parity when PCE = 1</td></tr><tr><td>1</td><td>0</td><td>15 bits data + parity</td></tr><tr><td>1</td><td>1</td><td>16 bits data when PCE = 0 or 16 bits data + parity when PCE = 1</td></tr></table></td></tr>
<tr><td>31
UART</td><td>UART Mode
This bit can be programmed in Initialization mode only.</td></tr>
</table>

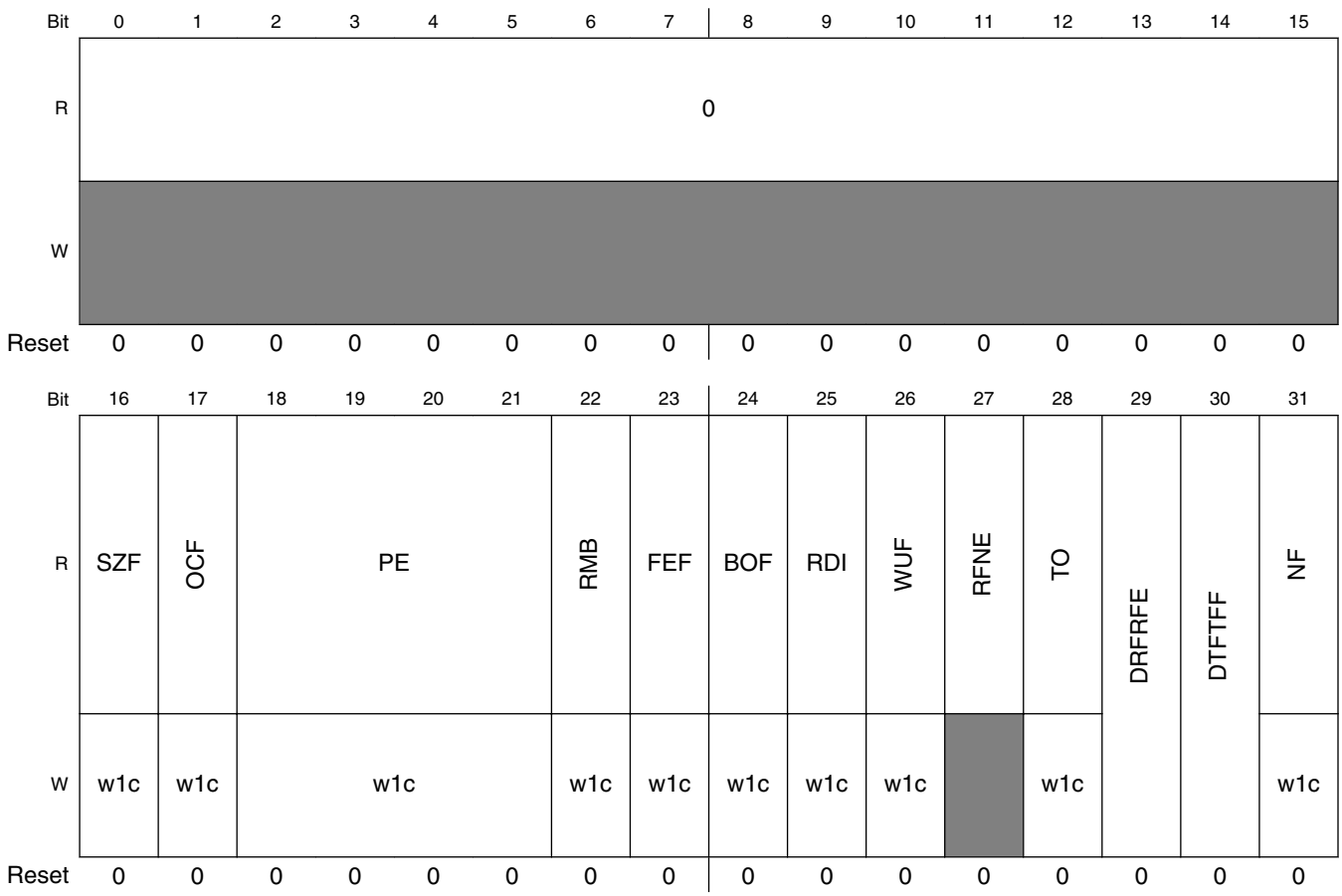
Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

Field	Description
	Register bit can be read in any mode, written only in initialization mode. 0 LIN mode 1 UART mode

54.5.6 UART Mode Status Register (LINFlexD_UARTSR)

Address: 0h base + 14h offset = 14h



LINFlexD_UARTSR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SZF	Stuck at Zero flag This bit is set by hardware when 100 dominant bits are detected. It should be cleared by software. An interrupt will be generated if the SZIE bit in LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.

Table continues on the next page...

LINFlexD_UARTSR field descriptions (continued)

Field	Description
17 OCF	Output Compare Flag An interrupt will be generated if the OCIE bit in LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set. **NOTE:** For Baud rate above 1 Mbit/s, this flag is not usable. 0 No output compare event occurred 1 The content of the counter has matched the content of LINOCR
18–21 PE	Parity Error flag These bits indicate if there is a Parity Error in the corresponding byte. No interrupt is generated if this error occurs. This bit will reflect the same value as in LINESR, when in initialization mode and UART bit set. **NOTE:** Parity is checked only after complete frame is received. Following are the conditions when WL bits = 01 or 11 (either PE0 or PE2 is only set). • When PE0 is set, it indicates Parity error in either first or second byte received • When PE2 is set, it indicates Parity error in either third or fourth byte received 0 No parity error 1 Parity error in the corresponding received byte
22 RMB	Release Message Buffer This bit should be cleared by software. This bit will reflect the same value as in LINSR, when in Initialization mode and the UART bit is set. 0 Buffer data is free 1 Buffer data ready to be read by software
23 FEF	Framing Error flag This bit is set by hardware when there is a framing error (invalid stop bit). It should be cleared by software. It will generate an interrupt if the FEIE bit of LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
24 BOF	FIFO/Buffer overrun flag This bit is set by hardware when there is a new byte received and the RMB bit is not cleared in UART buffer mode. In UART FIFO mode this bit is set when there is a new byte and the Rx FIFO is full. In UART FIFO mode, once Rx FIFO is full, the new received message will be discarded irrespective of the new value of the RBLM bit. In UART Rx Buffer mode, if RBLM is set then the new message received will be discarded; if RBLM is reset then the new message will overwrite the buffer. It can be cleared by software writing a 1. An interrupt is generated if the BOIE bit of LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
25 RDI	Receiver Data Input signal This bit reflects the current status of the RX pin.
26 WUF	Wakeup flag This bit is set by hardware when a falling edge is detected on the RX pin in sleep mode. It should be cleared by software. An interrupt will be generated if the WUIE bit in LINIER is set. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set.
27 RFNE	Receive FIFO Not Empty

Table continues on the next page...

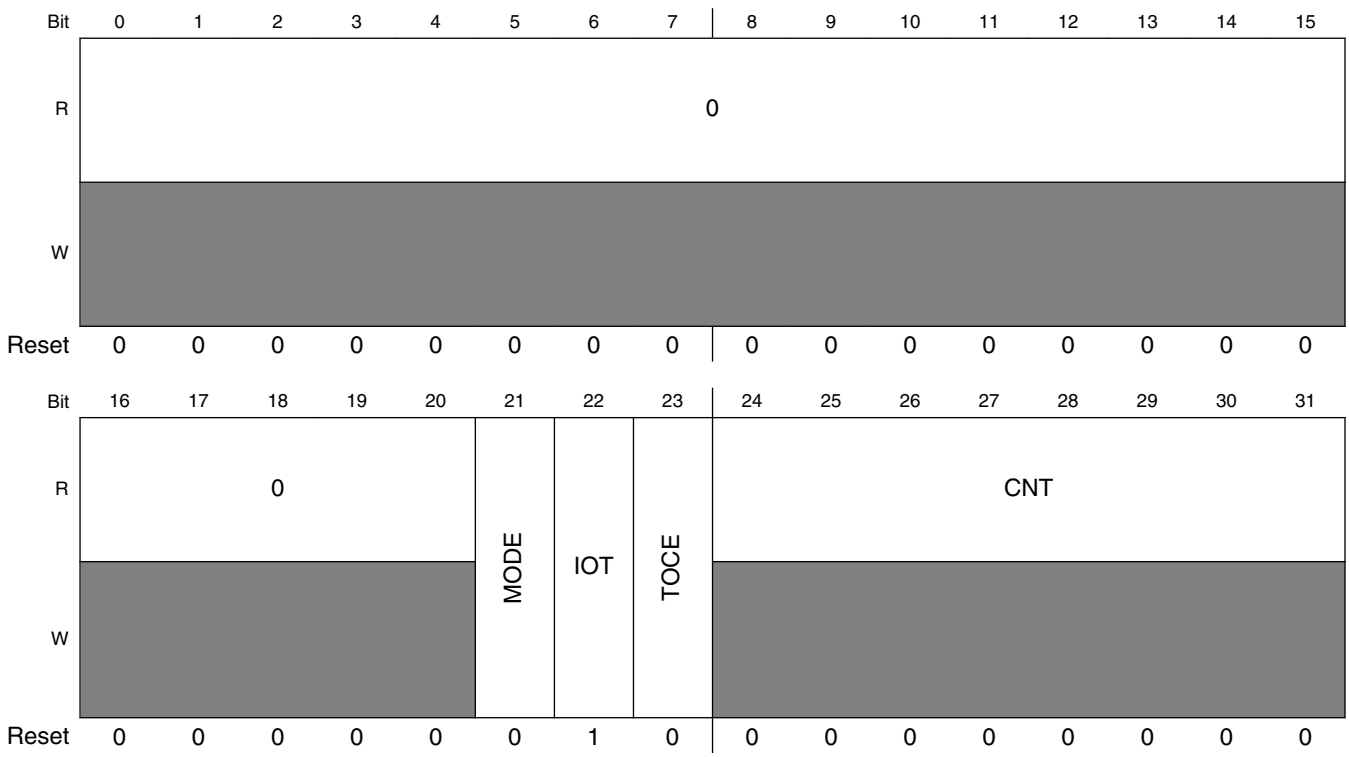
LINFlexD_UARTSR field descriptions (continued)

Field	Description
	RFNE bit is set by hardware in UART FIFO mode (RFBM = 1), when there is at least one data byte present in the receive FIFO. RFNE is a read-only bit for debugging purposes. This flag can be used by software in case of a timeout event.
28 TO	Timeout This bit is set by hardware when a UART timeout occurs — in other words, the value of UARTCTO becomes equal to the preset value of the timeout (UARTPTO register setting). TO should be cleared by software. The SR bit should be used to reset the receiver fsm to Idle state in case of UART TIMEOUT for UART reception, depending on the application, in both buffer and FIFO mode. An interrupt will be generated when LINIER.DBEIE/TOIE bit is set on the Error interrupt line in UART mode.
29 DRFRFE	Data Reception Completed Flag /Rx FIFO Empty Flag DRF is set by hardware in UART buffer mode (RFBM = 0) and indicates that the number of bytes programmed in RDFL have been received. DRF should be cleared by software. An interrupt will be generated if the DRIE bit in LINIER is set. DRF bit is set when the configured number of valid stop bits are received for the last frame (number of frames is configurable by RDFL bits). DRF is set irrespective of framing error in case framing error is in the last STOP bit configured, parity error or overrun error. This bit will reflect the same value as in LINSR when in Initialization mode and the UART bit is set. Register bit can be read/cleared by software. Writing 1 clears contents. RFE is set by hardware in UART FIFO mode (RFBM = 1) when the RX FIFO is empty. RFE is a read-only bit for debugging purposes. It is internally used by the DMA RX interface.
30 DTFTFF	Data Transmission Completed Flag/ TX FIFO Full Flag DTF is set by hardware in UART buffer mode (TFBM = 0) and indicates that data transmission is completed. DTF should be cleared by software. An interrupt will be generated if the DTIE bit in LINIER is set. This bit will reflect the same value as in LINSR when in Initialization mode and the UART bit is set. Register bit can be read/cleared by software. Writing 1 clears contents. TFF is set by hardware in UART FIFO mode (TFBM = 1) when TX FIFO is full. TFF is a read-only bit for debugging purposes. It is internally used by the DMA TX interface.
31 NF	Noise flag This bit is set by hardware when noise is detected in the received character. It should be cleared by software. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set. During reduced oversampling (ROSE bit = 1), it is enabled only when OSR = 8.

54.5.7 LIN Time-Out Control Status Register (LINFlexD_LINTCSR)

This register contains control and status bits for timeout feature.

Address: 0h base + 18h offset = 18h



LINFlexD_LINTCSR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 MODE	Time-out counter mode This bit can be configured only during initialization. Register bit can be read in any mode, written only in initialization mode. 0 LIN mode 1 Output compare mode
22 IOT	Idle on timeout Register bit can be read in any mode, written only in initialization mode. This feature is applicable only when MODE bit in LINTCSR is cleared. 0 LIN state machine does not reset to Idle on timeout 1 LIN state machine resets to Idle on timeout event

Table continues on the next page...

LINFlexD_LINTCSR field descriptions (continued)

Field	Description
23 TOCE	Time-out counter enable TOCE is always configurable by software in Initialization mode. If LIN state is other than INIT and if timer is configured in LIN mode, then hardware takes control of TOCE. 0 Time-out counter disable. OCF flag is not set on an output compare event. 1 Time-out counter enable. OCF flag is set if an output compare event occurs.
24–31 CNT	Counter Value These bits reflect the value of a counter used for timeout. **NOTE:** For proper functionality of this counter, LINIBRR should be >= 5.

54.5.8 LIN Output Compare Register (LINFlexD_LINOCR)

This register contains the value to be compared to the LINTCSR:CNT value. This register is writable by software only in Output Compare Mode.

Address: 0h base + 1Ch offset = 1Ch

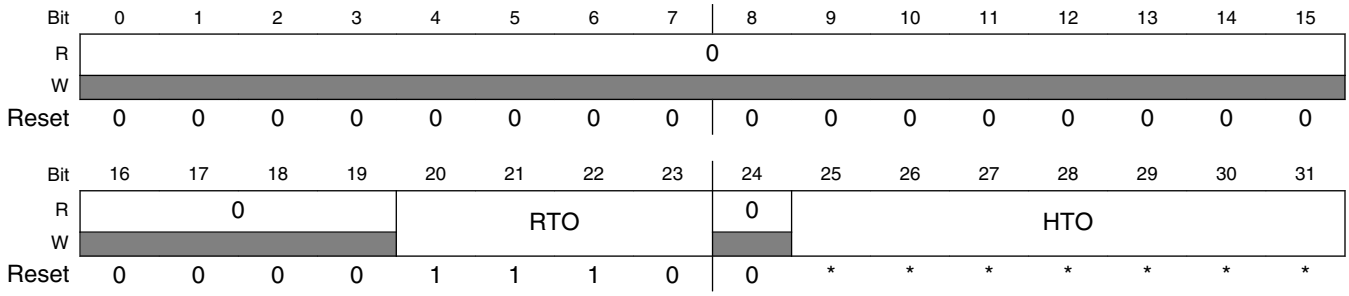
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																OC2								OC1							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

LINFlexD_LINOCR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 OC2	Output compare value 2
24–31 OC1	Output compare value 1

54.5.9 LIN Time-Out Control Register (LINFlexD_LINTOCR)

Address: 0h base + 20h offset = 20h



* Notes:

- HTO field: HTO reset values:

  HTO resets to 0011100b in Master only mode, if generic slave = 0.

  HTO resets to 0101100b in Master/Slave mode, if generic slave = 1.

LINFlexD_LINTOCR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 RTO	Response timeout value This is the response timeout duration (in bit time) for 1 byte. The reset value is 0Eh = 14, corresponding to $T_{Response_Maximum} = 1.4 \times T_{Response_Nominal}$
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 HTO	Header timeout value This register contains the header timeout duration (in bit time). This register can be written only for Slave mode. If Master mode is enabled then these bits will always reflect 28 (1.4 x 10 bits of sync + 1.4 x 10 bits of ID). For slave, these bits should be programmed without considering 11 bits of break.

54.5.10 LIN Fractional Baud Rate Register (LINFlexD_LINFBRR)

This register consists of bits that decide the fractional part of the LIN Baud Rate. It can be programmed only in Initialization mode.

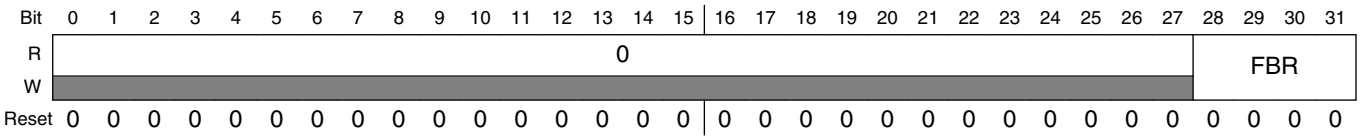
NOTE

When LASE bit is set, this register should be read only after AUTOSYNC_COMP bit in LINSR register is set to obtain the correct value.

NOTE

This register cannot be used when reduced oversampling is enabled (ROSE bit = 1)

Address: 0h base + 24h offset = 24h



LINFlexD_LINFBRR field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 FBR	Fractional Baud rates Register bit can be read in any mode, written only in initialization mode. 0000 Fraction(LDIV) = 0 0001 Fraction(LDIV) = 1/16 0010 Fraction(LDIV) = 2/16 0011 Fraction(LDIV) = 3/16 0100 Fraction(LDIV) = 4/16 0101 Fraction(LDIV) = 5/16 0110 Fraction(LDIV) = 6/16 0111 Fraction(LDIV) = 7/16 1000 Fraction(LDIV) = 8/16 1001 Fraction(LDIV) = 9/16 1010 Fraction(LDIV) = 10/16 1011 Fraction(LDIV) = 11/16 1100 Fraction(LDIV) = 12/16 1101 Fraction(LDIV) = 13/16 1110 Fraction(LDIV) = 14/16 1111 Fraction(LDIV) = 15/16

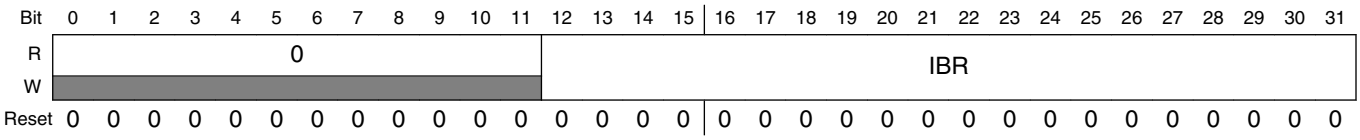
54.5.11 LIN Integer Baud Rate Register (LINFlexD_LINIBRR)

This register consists of control bits that decide the baud rate along with the LINFBRR. It can be programmed only in Initialization mode.

NOTE

When LASE bit is set, this register should be read only after AUTOSYNC_COMP bit in LINSR register is set to obtain the correct value.

Address: 0h base + 28h offset = 28h



LINFlexD_LINIBRR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 IBR	Integer Baud rates These bits along with the fractional baud rate bits decide the LIN baud rate. IBR = 0h: LIN clock disabled IBR = 1h: Mantissa (LDIV) = 1 ... IBR = FFFFEh: Mantissa (LDIV) = 1048574 IBR = FFFFFh: Mantissa (LDIV) = 1048575 Register bit can be read in any mode, written only in initialization mode.

54.5.12 LIN Checksum Field Register (LINFlexD_LINCFR)

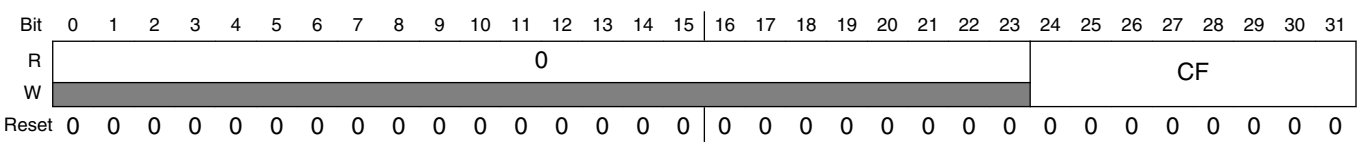
NOTE

There is a delay between 4 to 6 clock cycles of PBRIDGEx_CLK for the internal checksum's value (which is clocked with LIN_CLK/16 * LDIV) to reflect on LINCFR.

This register consists of checksum bits.

CFD	CCD	LINCHKSUM Read/write	Checksum sent
1	1	read/write	None
1	0	read-only	None
0	1	read/write	Programmed checksum
0	0	read-only	Calculated checksum

Address: 0h base + 2Ch offset = 2Ch



LINFlexD_LINCFR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 CF	Checksum bits When the CCD bit is reset these bits are read-only and are calculated by hardware. When the CCD bit is set, these bits can be written by software.

54.5.13 LIN Control Register 2 (LINFlexD_LINCR2)

This register includes control status bits related to buffer operations.

NOTE

When accessing the LINCR2 register, each reserved bit should be written to its original reset value.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TBDE	IOBE	IOPE	WURQ	DDRQ	DTRQ	ABRQ	HTRQ	0							
W																
Reset	0	1	*	0	0	0	0	0	0	0	0	0	0	0	0	0

* Notes:

- IOPE field: When slave = 0, this field always reads ‘0’ and cannot be programmed else this bit is programmable and reset value is 1.

LINFlexD_LINCR2 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 TBDE	Two Bit delimiter bit This bit can be set in Initialization mode only. Register bit can be read in any mode, written only in initialization mode.

Table continues on the next page...

LINFlexD_LINCR2 field descriptions (continued)

Field	Description
	0 Delimiter length in break field is 1 bit 1 Delimiter length in break field is 2 bits
17 IOBE	Idle on Bit Error This bit can be set in Initialization mode only. Register bit can be read in any mode, written only in initialization mode. 0 Bit Error does not reset LIN state machine 1 Bit Error resets LIN state machine
18 IOPE	Idle on Identifier Parity Error This bit can be set in Initialization mode only. Register bit can be read in any mode, written only in initialization mode. If generic slave = 0, then this bit will always read a 1 else 0, and cannot be programmed. 0 Parity Error does not reset LIN state machine 1 Parity Error resets LIN state machine
19 WURQ	Wakeup Generate Request Setting this bit will generate a wakeup pulse. It is reset by hardware when the wakeup character has been transmitted. The character sent during wakeup is copied from BDRL (DATA0). Note that this bit cannot be set in Sleep mode — software has to exit Sleep mode before setting this bit. Bit Error is not checked when transmitting the wakeup request. Register bit can be read/set by software
20 DDRQ	Data Discard request Set by software to stop data reception if the frame does not concern the node. This bit is reset by hardware once LINFlexD ignores the response and moves to Idle state. For LIN slave this bit can be set only when HRF bit is set and Identifier is software-filtered. Register bit can be read/set by software
21 DTRQ	Data Transmission Request Set by software in slave mode to request the transmission of the LIN Data field stored in the Buffer data register. This bit can be set only when the HRF bit is set (to ensure that data transmission is requested only after a header reception). Cleared by hardware when the request has been completed, or on abort request or error condition. In Master mode, this bit is set by hardware when the DIR bit is set and header transmission is complete. Register bit can be read/set by software
22 ABRQ	Abort Request Set by software to abort the current transmission. Cleared by hardware when the transmission has been aborted. LINFlexD aborts the transmission at the end of the current bit. This bit can abort a wakeup request also and can be used in UART mode also. Register bit can be read/set by software Register bit can be read/set by software
23 HTRQ	Header Transmission Request Set by software to request the transmission of the LIN Header.

Table continues on the next page...

LINFlexD_LINCR2 field descriptions (continued)

Field	Description
	Cleared by hardware when the request has been completed or on abort request. This bit has no effect in UART mode. **NOTE:** In master mode, if both HTRQ and ABRQ are set at the same time then ABRQ has no effect. Similarly, in slave mode after header reception, if DTRQ and ABRQ are simultaneously set then ABRQ has no effect.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

54.5.14 Buffer Identifier Register (LINFlexD_BIDR)

This register contains the bits which provide information about the identifier of the transaction and other related information.

NOTE

All the fields (ID, CSS, DIR, DFL) of the BIDR register must be updated when an ID filter (enabled) in Slave mode (Tx or Rx) matches the ID received.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DFL						DIR	CCS	0		ID					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_BIDR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–21 DFL	Data Field Length Number of data bytes in the response part of the frame. DFL = Number of data bytes - 1 **NOTE:** DFL[5:3] is provided for extended frames. Normally LIN mode will use DFL[2:0]. Identifier filters are now compatible with DFL[5:0] also.
22 DIR	Direction This bit controls the direction of the data field. 0 LINFlexD receives the data and copy them in the BDR registers 1 LINFlexD transmits the data from the BDR registers

Table continues on the next page...

LINFlexD_BIDR field descriptions (continued)

Field	Description
23 CCS	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification rev. 2.0 and higher. 1 Classic Checksum covering Data filed only. This is compatible with LIN specification 1.3 and lower.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 ID	Identifier Identifier part of the identifier field without the identifier parity. This field can be written only in Master mode (MME = '1').

54.5.15 Buffer Data Register Least Significant (LINFlexD_BDRL)

This register is a part of an 8-byte data buffer.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA3								DATA2								DATA1								DATA0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_BDRL field descriptions

Field	Description
0–7 DATA3	Data Byte 3 Data byte 3 of the data field.
8–15 DATA2	Data Byte 2 Data byte 2 of the data field.
16–23 DATA1	Data Byte 1 Data byte 1of the data field.
24–31 DATA0	Data Byte 0 Data byte 0 of the data field.

54.5.16 Buffer Data Register Most Significant (LINFlexD_BDRM)

This register is a part of an 8-byte data buffer.

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA7								DATA6								DATA5								DATA4							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_BDRM field descriptions

Field	Description
0–7 DATA7	Data Byte 7 Data byte 7 of the data field.
8–15 DATA6	Data Byte 6 Data byte 6 of the data field.
16–23 DATA5	Data Byte 5 Data byte 5of the data field.
24–31 DATA4	Data Byte 4 Data byte 4 of the data field.

54.5.17 Identifier Filter Enable Register (LINFlexD_IFER)

This register enables/disables a particular filter.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																FACT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_IFER field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 FACT	Filter active The software sets the bit FACT[x] to activate the filter x in Identifier list mode. Register bit can be read in any mode, written only in initialization mode.

Table continues on the next page...

LINFlexD_IFER field descriptions (continued)

Field	Description
	1. In Identifier mask mode, FACT (2n+1) have no effect on the corresponding filters as they act as mask for the Identifier 2n. 2. The length of this field depends on the value of no_of_filters. 3. The register diagram depicts the maximum no_of_filters, which can be up to 16. **NOTE:** Refer to chip configuration details to see the number of filters used in the device.

54.5.18 Identifier Filter Match Index (LINFlexD_IFMI)

This register contains the index corresponding to the received ID. It can be used to read or write the data directly in RAM.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																											IFMI				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_IFMI field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 IFMI	Filter match index Upon a filter match with xth filter – IFMI[4:0] = x+1. On no match IFMI is equal to 00h. This register contains the index corresponding to the received identifier. It can be used to directly write or read the data in SRAM (see Slave mode for more details).

54.5.19 Identifier Filter Mode Register (LINFlexD_IFMR)

This register configures the modes of filters.

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																											IFM				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_IFMR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 IFM	Filter mode Register bit can be read in any mode, written only in initialization mode. 0 Filters 2n and 2n+1 are in identifier list mode 1 Filters 2n and 2n+1 are in mask mode, where filter 2n+1 is the mask for filter 2n

54.5.20 Identifier Filter Control Register (LINFlexD_IFCR*n*)

This register is read-only in normal mode and can be programmed only in Initialization mode.

Even-numbered instances (IFCR0, IFCR2, IFCR4, ...) of this register act as filters in both Identifier list and Identifier mask modes.

Odd-numbered instances (IFCR1, IFCR3, IFCR5, ...) of this register act:

- As filters in Identifier list mode
- As a mask for the preceding-numbered register in identifier mask mode

Refer to chip configuration details to see the number of filters used in the device.

Address: 0h base + 4Ch offset + (4d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DFL						DIR	CCS	0		ID					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_IFCR*n* field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–21 DFL	Data Field Length Number of data bytes in the response part of the frame. DFL[5:0] = Number of data bytes - 1 Register bit can be read in any mode, written only in initialization mode.

Table continues on the next page...

LINFlexD_IFCR*n* field descriptions (continued)

Field	Description
22 DIR	Direction This bit controls the direction of the data field. Register bit can be read in any mode, written only in initialization mode. 0 LINFlexD receives data and copies to the BDR registers 1 LINFlexD transmits data from the BDR registers
23 CCS	Classic Checksum This bit controls the type of checksum applied on the current message. Register bit can be read in any mode, written only in initialization mode. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification rev. 2.0 and higher. 1 Classic Checksum covering Data field only. This is compatible with LIN specification 1.3 and lower.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 ID	Identifier Identifier part of the identifier field without the identifier parity. Register bit can be read in any mode, written only in initialization mode.

54.5.21 Global Control Register (LINFlexD_GCR)

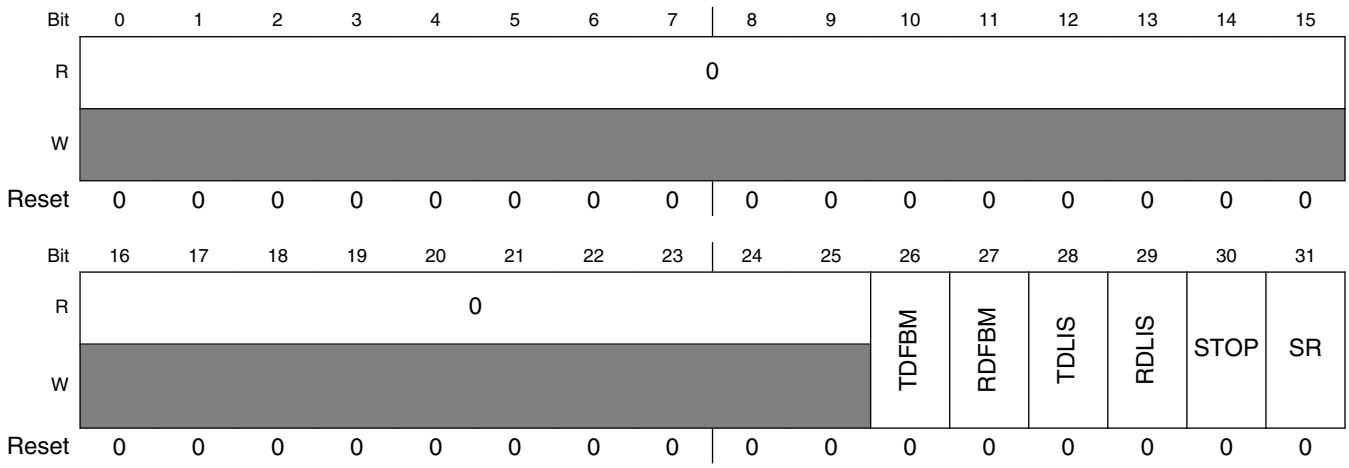
This register is read-only in Normal mode and can be programmed only in Initialization mode. This is a global control register — in other words, the register configuration will be applied in LIN mode as well as in UART mode.

The address offset depends on the no_of_filters. Refer to the chip configuration details for the number of filters used in this device.

Table 54-57. Register fields reset by SR

Register	Comment
LINSR	All fields except RXbusy and AUTOSYNC_COMP are reset
LINESR	All fields are reset
LINTCSR	Only CNT[0:7] is reset
UARTSR	All fields except RFE & TFF are reset
UARTCR	Only TFC & RFC are reset
UARTCTO	All fields are reset

Address: 0h base + 8Ch offset = 8Ch



LINFlexD_GCR field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 TDFBM	Transmit data first bit MSB This bit controls the first bit of transmit data (payload only) as MSB/LSB in both UART and LIN modes. Register bit can be read in any mode, written only in initialization mode. 0 The first bit of transmitted data is LSB — in other words, the first bit transmitted is mapped on LSB bit (BDR (0), BDR (8), BDR (16), BDR (24)) 1 The first bit of transmitted data is MSB — in other words, the first bit transmitted is mapped on MSB bit (BDR (7), BDR (15), BDR (23), BDR (31))
27 RDFBM	Received data first bit MSB This bit controls the first bit of received data (payload only) as MSB/LSB both in UART and LIN modes. Register bit can be read in any mode, written only in initialization mode. 0 The first bit of received data is LSB — in other words, the first bit received is mapped on LSB bit (BDR (0), BDR (8), BDR (16), BDR (24)) 1 The first bit of received data is MSB — in other words, the first bit received is mapped on MSB bit (BDR (7), BDR (15), BDR (23), BDR (31))
28 TDLIS	Transmit data level inversion selection This bit controls the data inversion of transmitted data (payload only) in both UART and LIN modes. Register bit can be read in any mode, written only in initialization mode. 0 Transmitted data is not inverted 1 Transmitted data is inverted
29 RDLIS	Received data level inversion selection This bit controls the data inversion of received data (payload only) in both UART and LIN modes. Register bit can be read in any mode, written only in initialization mode. 0 Received data is not inverted 1 Received data is inverted

Table continues on the next page...

LINFlexD_GCR field descriptions (continued)

Field	Description
30 STOP	1/2 stop bit configuration This bit controls the number of stop bit transmitted data in both UART and LIN modes. The stop bit is configured for all the fields (Delimiter, Sync, ID, Checksum, Payload). Register bit can be read in any mode, written only in initialization mode. 0 1 stop bit 1 2 stop bits
31 SR	Soft reset SR executes a soft reset of the LINFlexD controller (FSMs, FIFO pointers, counters, timers, status and error registers) without modifying the configuration registers when a 1 write operation is performed. This bit should be cleared by software to perform further operations (this bit is not cleared by hardware). Register bit can be written only be software in initialization mode. Bit is always read 0 by software. Table 54-57 describes the register fields reset by SR.

54.5.22 UART Preset Timeout Register (LINFlexD_UARTPTO)

This register contains the preset value of the timeout register in UART mode and is programmed according to the number of bits to be receivedor to monitor the idle state of the reception line. This register can be written by software any time.

The address offset depends on no_of_filters. Refer to the chip configuration details to see the number of filters used in the device.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																PTO															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

LINFlexD_UARTPTO field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 PTO	Preset Timeout PTO defines the preset value of timeout counter. A zero-value is forbidden, otherwise the UARTSR[TO] status bit is immediately set. Refer also to register UARTCTO.

54.5.23 UART Current Timeout Register (LINFlexD_UARTCTO)

This register contains the current timeout value in UART mode, and is used in conjunction with the UARTPTO register (see UART Preset Timeout Register (LINFlexD_UARTPTO) to monitor the number of bits received by UARTor to monitor the idle state of the reception line. UART timeout works in both CPU and DMA modes.

The timeout counter:

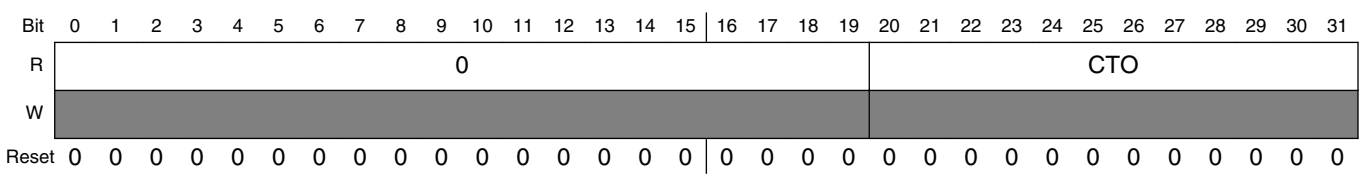
- Starts at zero and counts upward
- Is clocked with LIN_CLK / (16 * LDIV) synchronized to PBRIDGEx_CLK (when ROSE = 0)
- Is clocked with LIN_CLK / (OSR * IBRR) synchronized to PBRIDGEx_CLK (when ROSE = 1)
- Is automatically enabled when UARTCR[RXEN] = 1

It is reset when:

- Number of frames received is equal to NEF bits
- UARTCTO becomes equal to UARTPTO
- Whenever UARTPTO is written
- When DRF is set and DTU bit = 1

The address offset depends on no_of_filters. Refer to chip configuration details to see the number of filters used in the device.

Address: 0h base + 94h offset = 94h



LINFlexD_UARTCTO field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 CTO	Current Timeout

Table continues on the next page...

LINFlexD_UARTCTO field descriptions (continued)

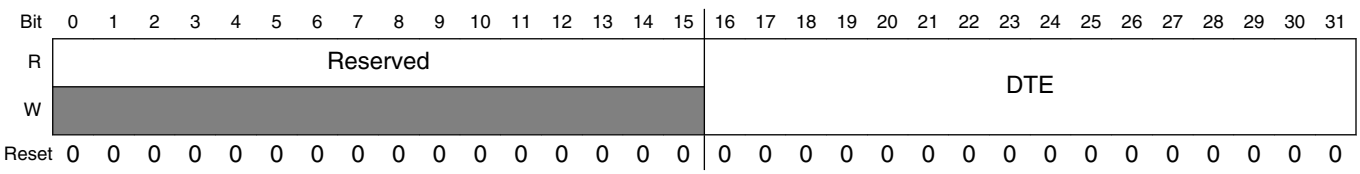
Field	Description
	CTO defines the current value of the timeout counter. CTO is a read-only field. CTO is reset every time UARTPTO is re-initialized, or UARTCTO = UARTPTO, or by hard/soft reset. When the CTO value matches the preset value (UARTPTO), the status bit UARTSR[TO] is set.

54.5.24 DMA Tx Enable Register (LINFlexD_DMATXE)

This register enables the DMA TX interface. This register can be written and read by software anytime.

The address offset depends on no_of_filters. Refer to chip configuration details to see the number of filters used in the device.

Address: 0h base + 98h offset = 98h



LINFlexD_DMATXE field descriptions

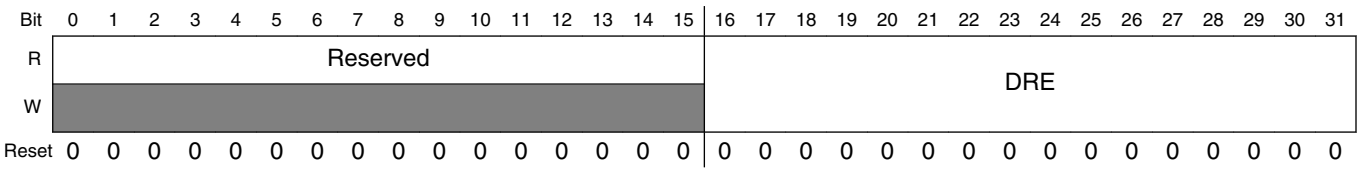
Field	Description
0–15 Reserved	This field is reserved. Reserved **NOTE:** The number of reserved bits varies, and is equal to 32 - $2^{TX_CH_NUM}$. These lies from 0 to (31 - $2^{TX_CH_NUM}$). Refer to the chip configuration details for the value of TX_CH_NUM used in this device.
16–31 DTE	DMA Tx channel Y enable **NOTE:** The actual size of the register DMATXE depends on the value of the static parameter TX_CH_NUM. The size is [$2^{TX_CH_NUM}$ - 1 : 0]. The position of these bits are (32 - $2^{TX_CH_NUM}$) to 31. When DMATXE = 0x00000000, the DMA TX interface FSM is forced (soft reset) in Idle state. **NOTE:** Refer to the chip configuration details for the value of TX_CH_NUM used in this device. 0 DMA Tx channel Y disabled 1 DMA Tx channel Y enabled

54.5.25 DMA Rx Enable Register (LINFlexD_DMARXE)

This register enables the DMA RX interface. This register can be written and read by software any time.

The address offset depends on no_of_filters. Refer to device configuration chapter to see the number of filters used in the device.

Address: 0h base + 9Ch offset = 9Ch



LINFlexD_DMARXE field descriptions

Field	Description
0–15 Reserved	This field is reserved. Reserved **NOTE:** The number of reserved bits varies, and is equal to 32 - $2^{RX_CH_NUM}$. These lies from 0 to (31 - $2^{RX_CH_NUM}$). Refer to the chip configuration details for the value of RX_CH_NUM used in this device.
16–31 DRE	DMA Rx channel Y enable **NOTE:** The actual size of the register DMARXE depends on the value of the static parameter RX_CH_NUM. The size is [$2^{RX_CH_NUM}$ -1 : 0]. The position of these bits are (32 - $2^{RX_CH_NUM}$) to 31. When DMARXE = 0x00000000, the DMA RX interface FSM is forced (soft reset) in Idle state. **NOTE:** Refer to the chip configuration details for the value of RX_CH_NUM used in this device. 0 DMA Rx channel Y disabled 1 DMA Rx channel Y enabled

54.6 Programming considerations

The next subsections describe the various configurations in which the LINFlexD module can be used.

54.6.1 Master node

LINFlexD acts as Master when the MME bit is set.

54.6.1.1 Transmitter

Transmitter Master Node - Transmitter configuration.

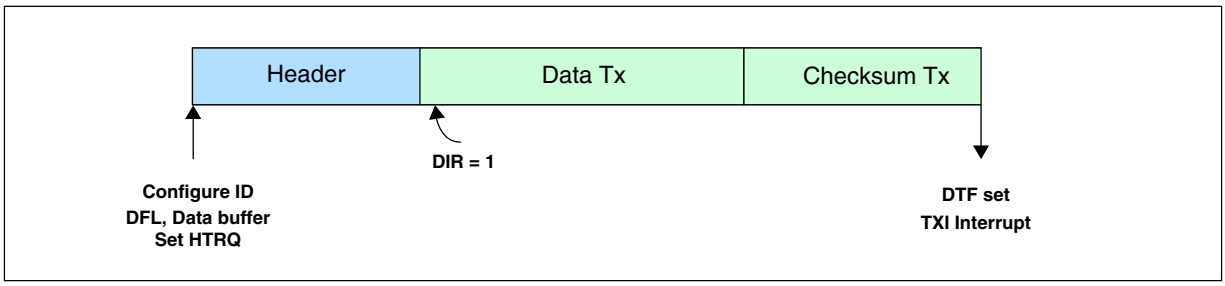


Figure 54-80. Master node – Transmitter configuration

54.6.1.2 Receiver

Receiver Master Mode - RX Configuration

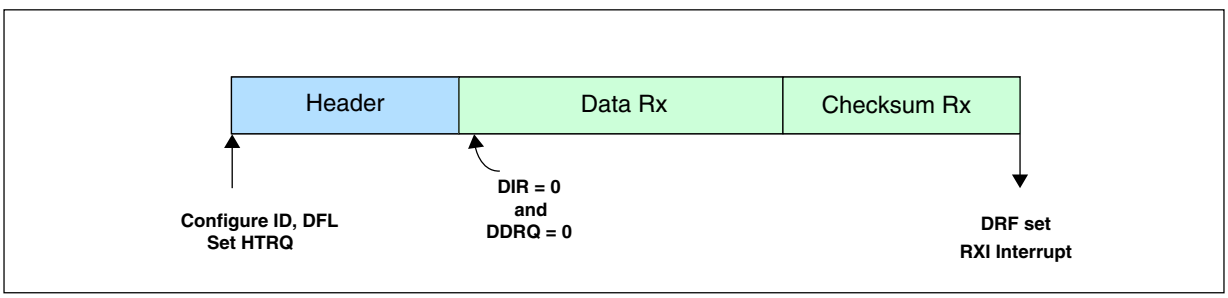


Figure 54-81. Master node – Receiver configuration

54.6.1.3 Transmitter, Bit Error

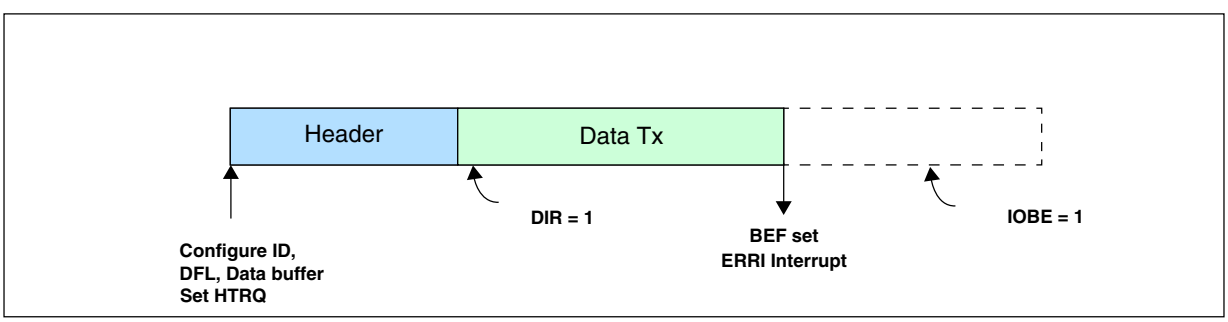


Figure 54-82. Master node – Transmitter, Bit Error configuration

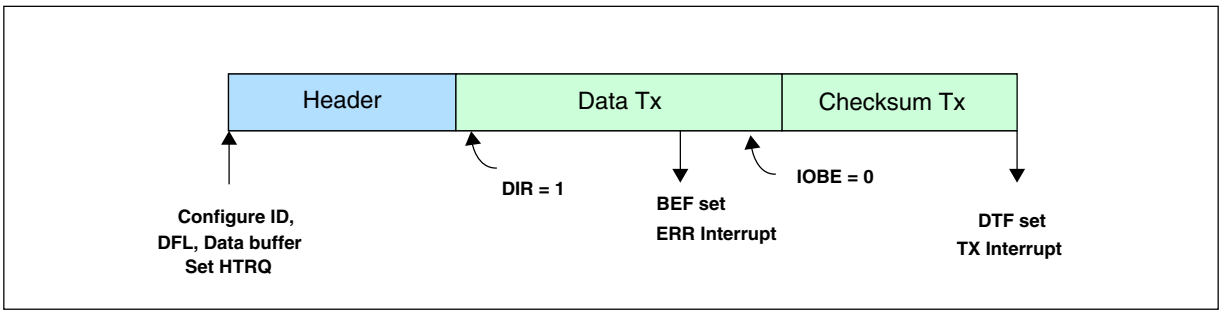


Figure 54-83. Master node – Transmitter, Checksum Error configuration

54.6.1.4 Receiver, Checksum Error

Checksum Error for Receiver.

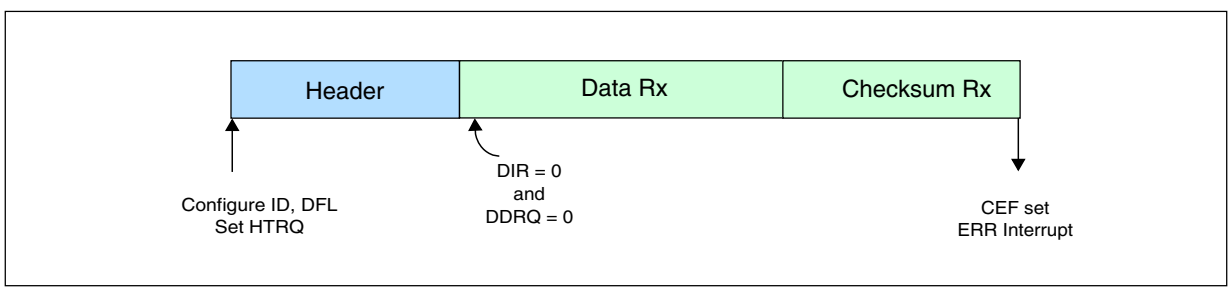


Figure 54-84. Master node – Receiver, Checksum Error configuration

54.6.2 Slave node

Slave node (transmitter).

54.6.2.1 Transmitter (no identifier filters)

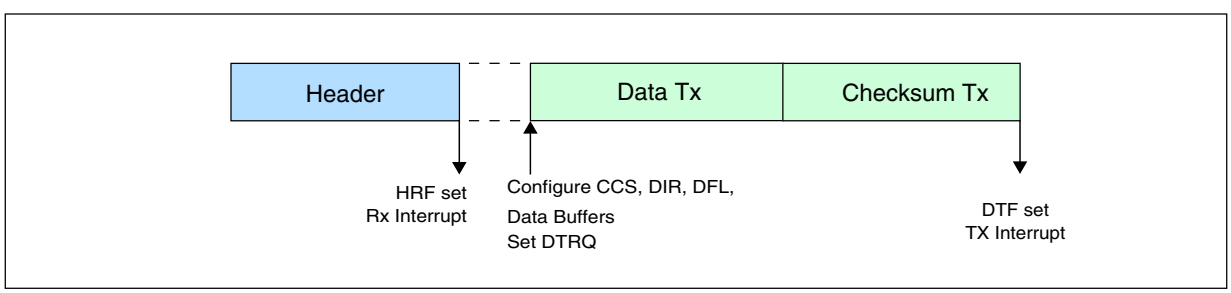


Figure 54-85. Slave node – Transmitter (no identifier filters) configuration

54.6.2.2 Receiver (no identifier filters)

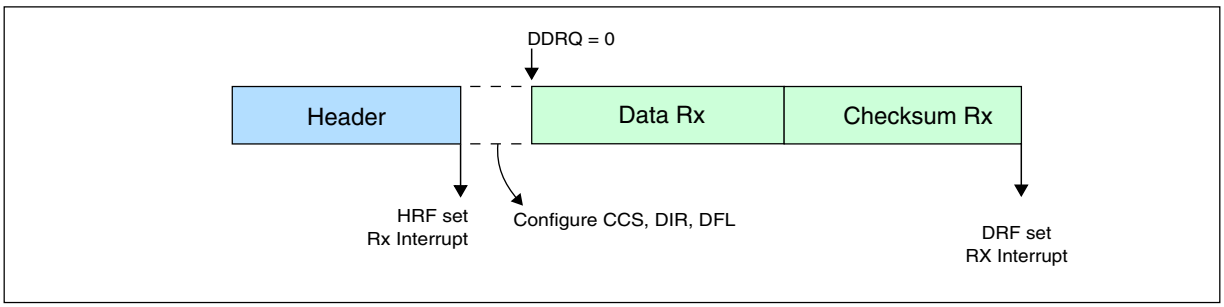


Figure 54-86. Slave node – Receiver (no identifier filters) configuration

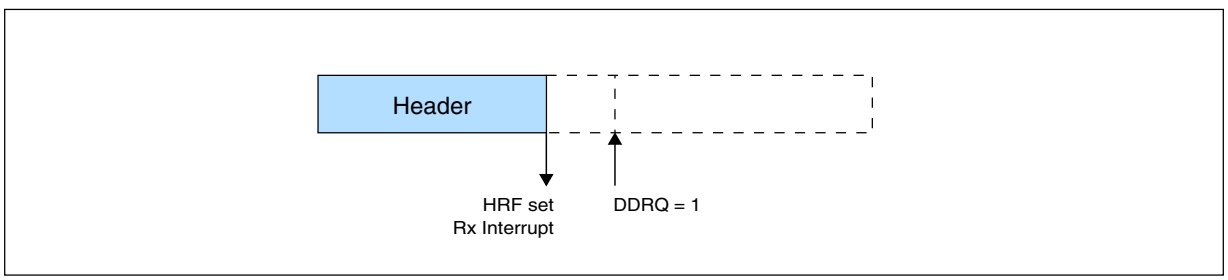


Figure 54-87. Slave node – Receiver (no identifier filters) configuration

54.6.2.3 No filters, Transmitter, Bit error

The following figure shows Slave node - No filters, Transmitter, Bit error configuration.

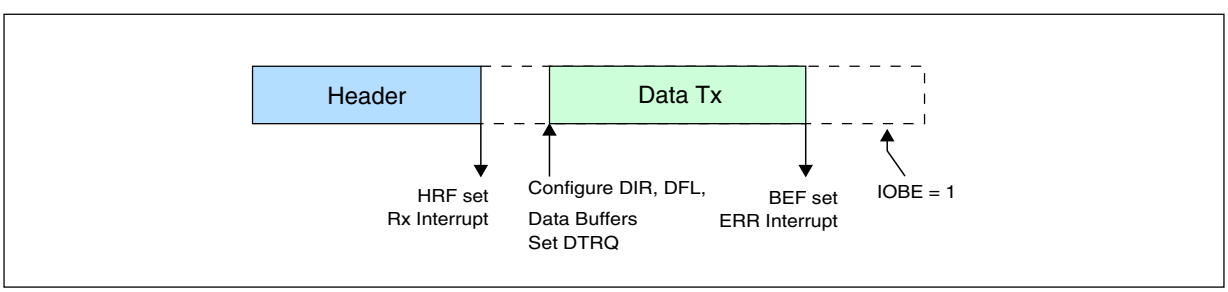


Figure 54-88. Slave node – No filters, Transmitter, Bit error configuration

54.6.2.4 No filters, Receiver, Checksum Error

The following figure shows Slave node - No filters, Receiver, Checksum error configuration.

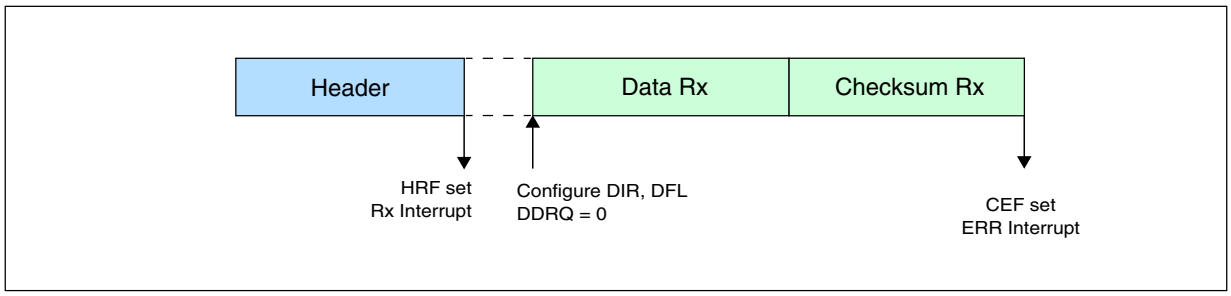


Figure 54-89. Slave node – No filters, Receiver, Checksum error configuration

54.6.2.5 At least one TX filter, BF is reset, ID matches filter

This configuration can be used in case slave never receives data, for example, sensor.

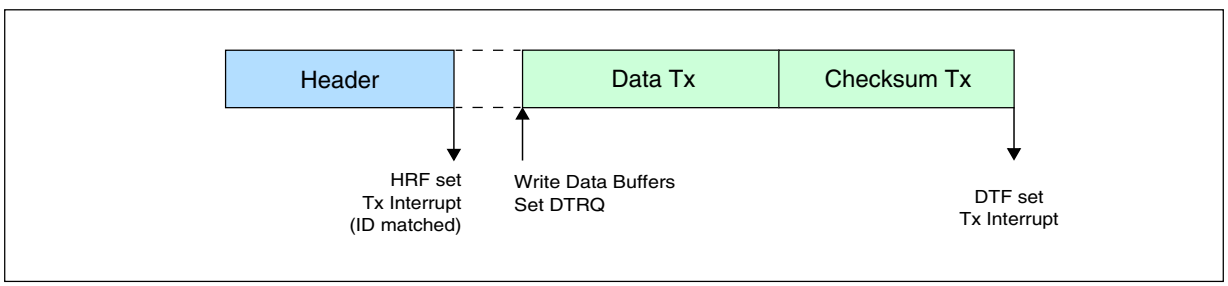


Figure 54-90. Slave node – one TX filter configuration

54.6.2.6 At least one RX filter, BF reset, ID matches filter

The following figure shows Slave node - one RX filter configuration.

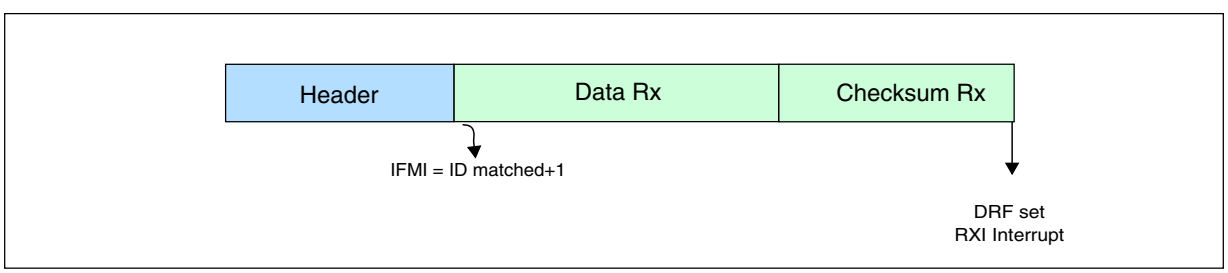


Figure 54-91. Slave node – one RX filter configuration

54.6.2.7 RX only, TX only, RX & TX filters, ID not matching filter, BF reset

The following figure shows RX only, TX only, RX and TX filters configuration.

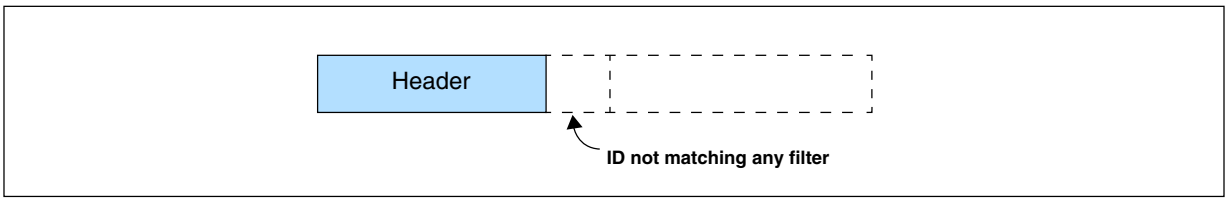


Figure 54-92. RX only, TX only, RX & TX filters configuration

54.6.2.8 TX filter, BF is set

This configuration is used when:

- All TX IDs are handled by filters.
- There are not enough other filters to handle all reception IDs.

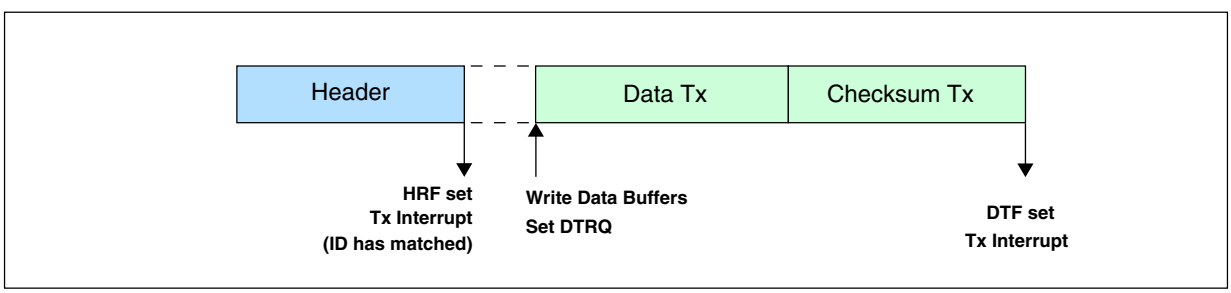


Figure 54-93. TX filter, BF is set configuration – ID has matched

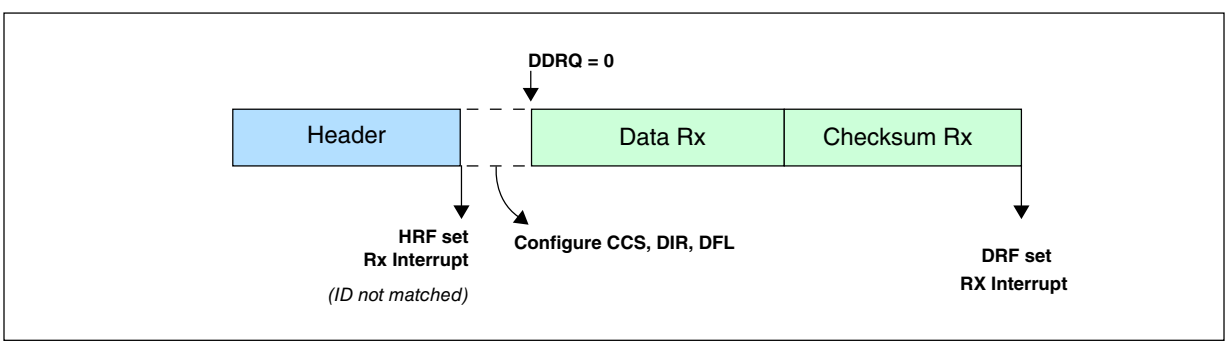


Figure 54-94. TX filter, BF is set configuration – ID not matched

54.6.2.9 RX filter, BF is set

The following figures show RX filter configurations.

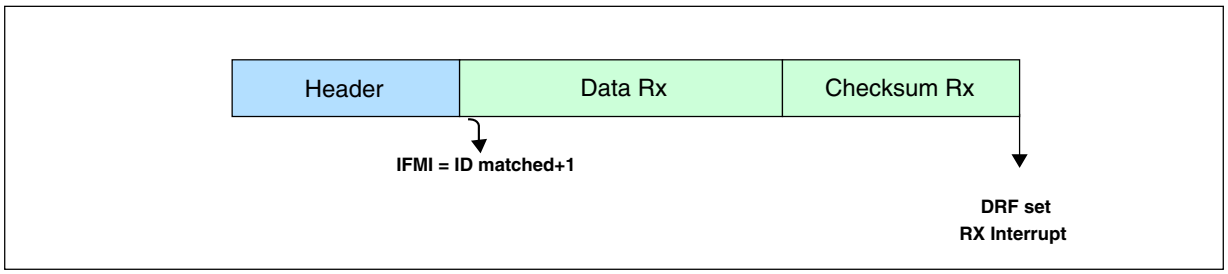


Figure 54-95. RX filter, BF is set configuration – ID matched

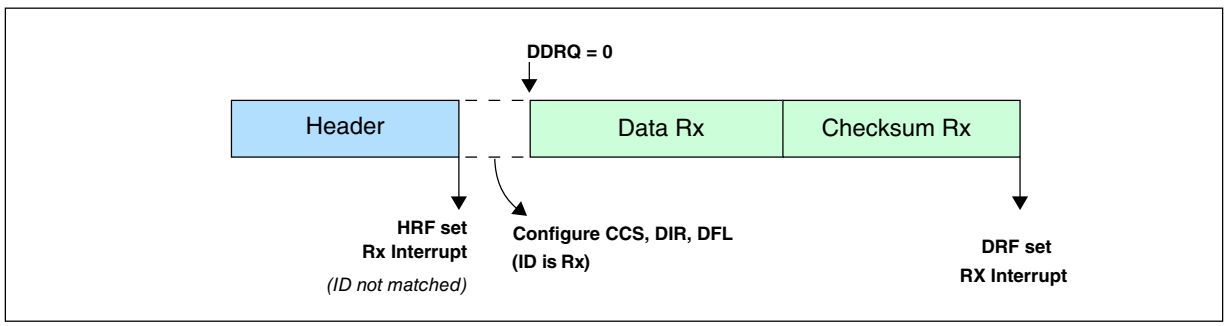


Figure 54-96. RX filter, BF is set configuration – ID not matched (ID is Rx)

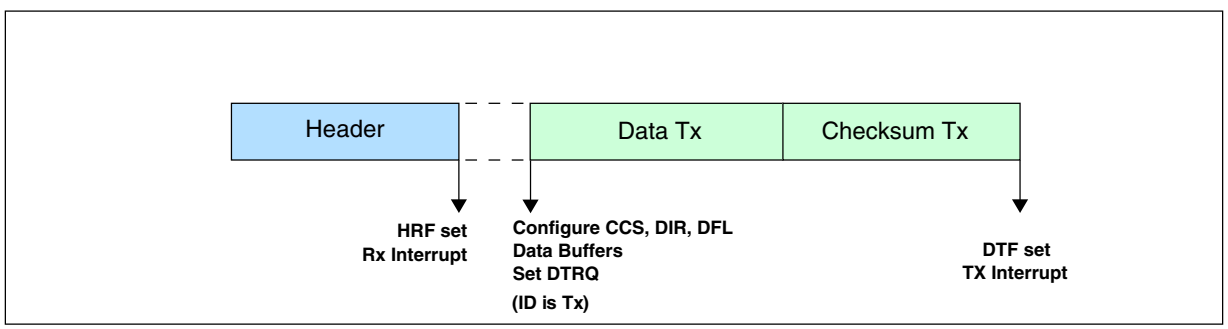


Figure 54-97. RX filter, BF is set configuration – (ID is Tx)

54.6.2.10 TX filter, RX filter, BF set

This configuration is used when:

- There are not enough filters.
- The filters are used for most frequently used IDs to reduce CPU usage.

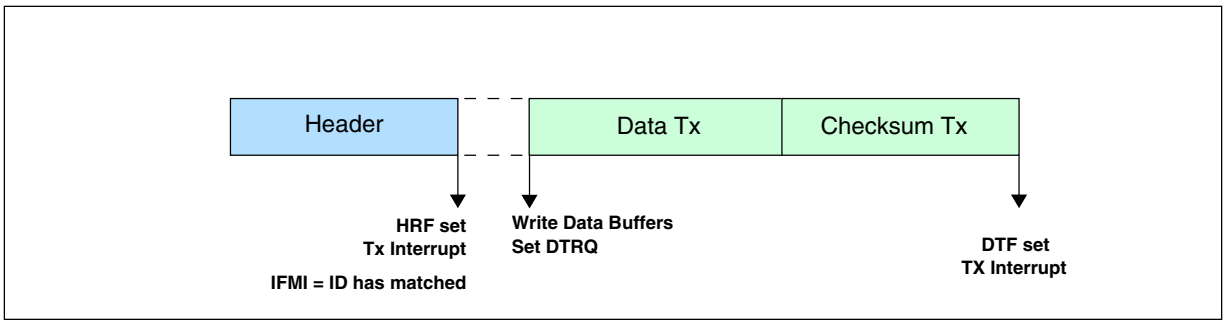


Figure 54-98. TX filter, RX filter, BF set configuration – (ID matched)

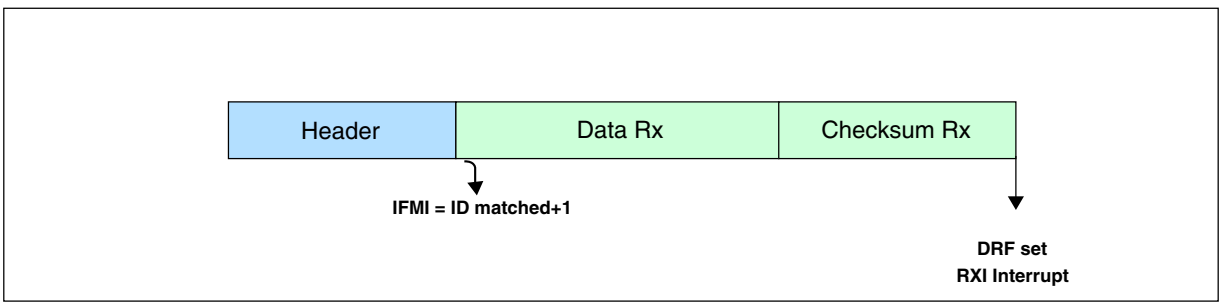


Figure 54-99. TX filter, RX filter, BF set configuration – (ID matched + 1)

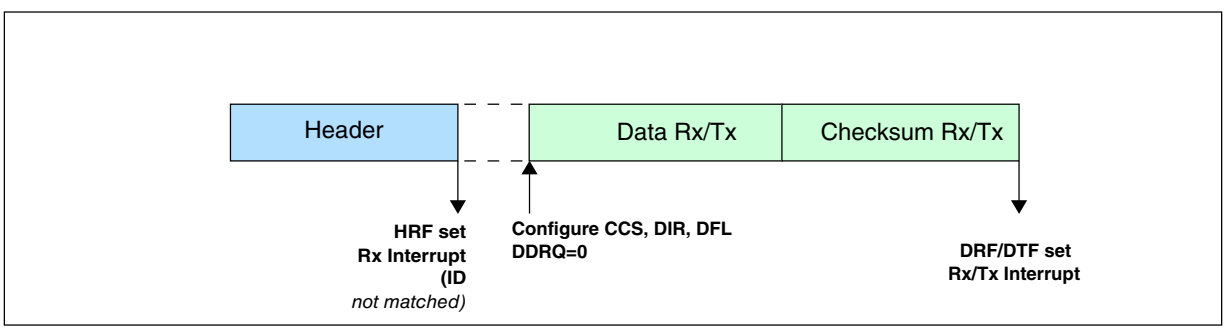


Figure 54-100. TX filter, RX filter, BF set configuration – (ID not matched, RX/TX Interrupt)

54.6.3 Extended frames

The following figure shows Extended frames (RX Interrupt).

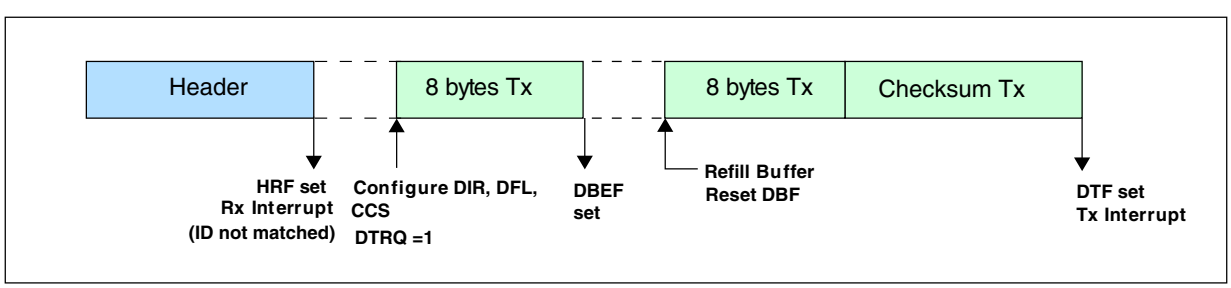


Figure 54-101. Extended frames (TX Interrupt)

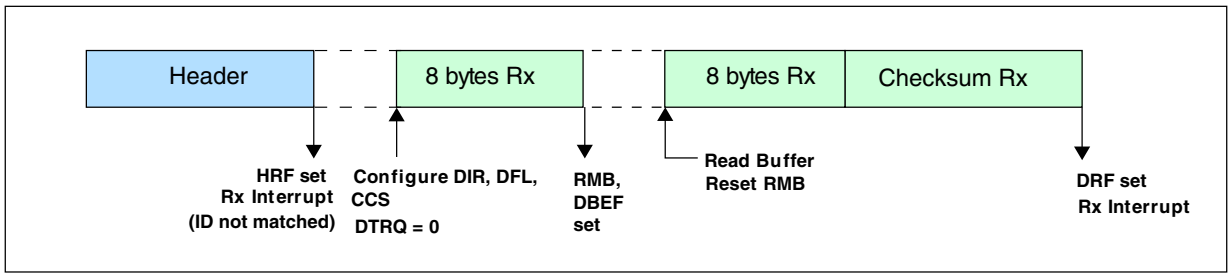


Figure 54-102. Extended frames (RX Interrupt)

54.6.4 Timeout

Master Node: Response (during reception only) and frame timeout are checked.

Slave Node: Header, Response (during reception only), and frame timeout are checked.

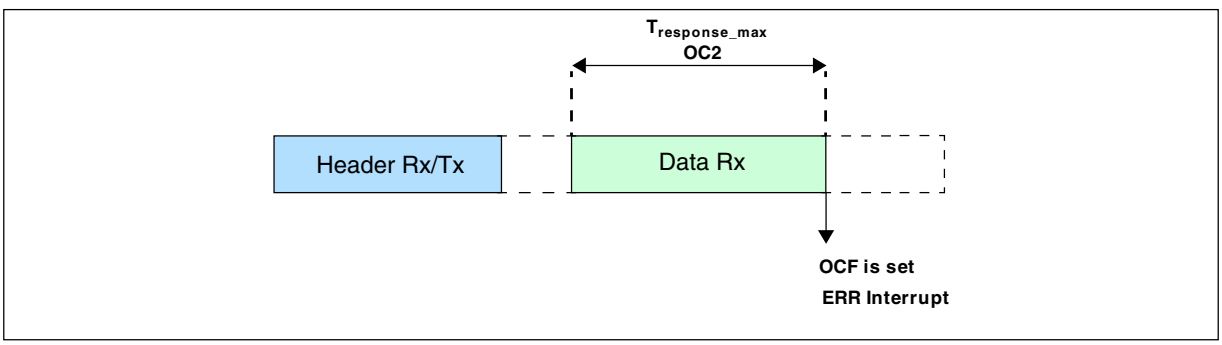


Figure 54-103. Response timeout

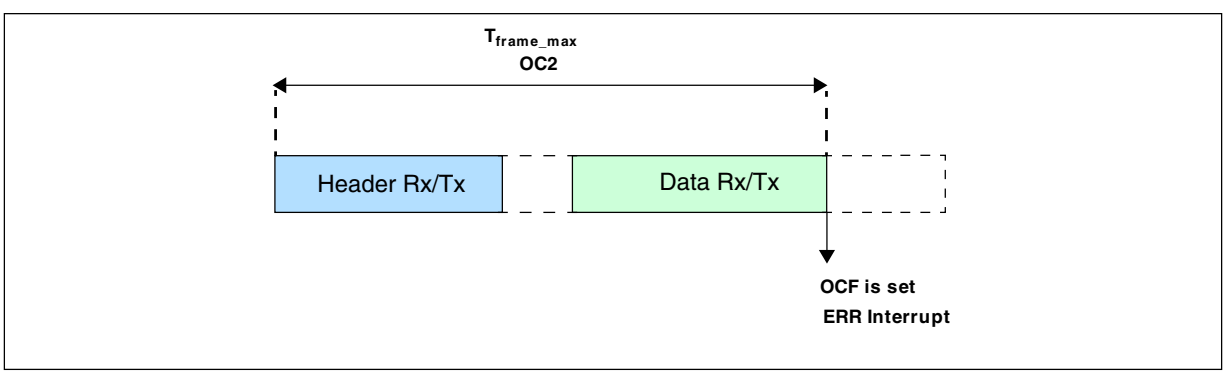


Figure 54-104. Frame timeout

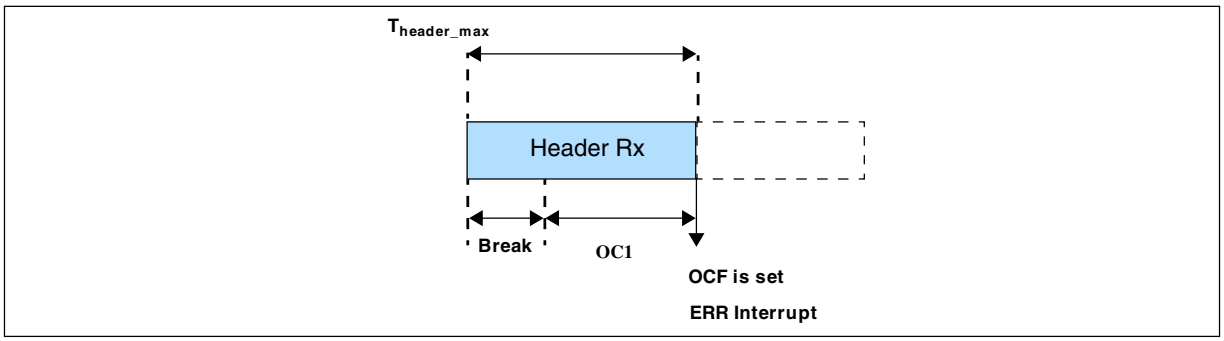


Figure 54-105. Header timeout

54.6.5 UART mode

The following figure shows UART mode configuration.

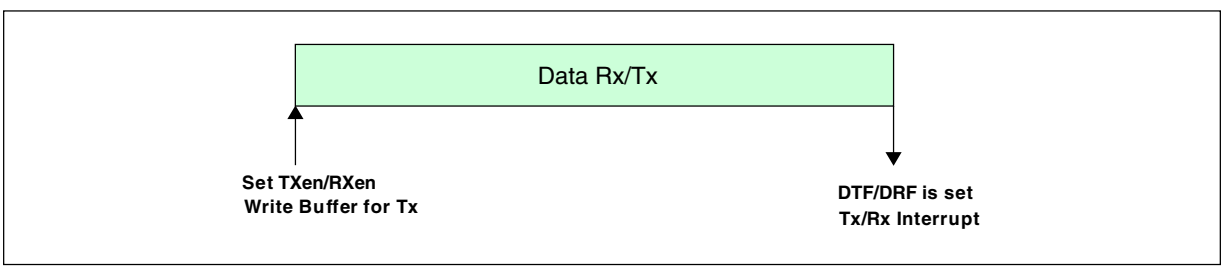


Figure 54-106. UART mode configuration

54.6.6 Interrupts

Interrupt section.

Table 54-62. Interrupts

Interrupt Event	Event flag	Enable control bit	Interrupt vector
Stuck at zero	SZF	SZIE	Error
Output compare	OCF	OCIE	Error
Bit error	BEF	BEIE	Error
Checksum error	CEF	CEIE	Error
Header error	SFEF orSDEF orIDPEF	HEIE	Error
Frame error	FEF	FEIE	Error
Buffer Overrun error	BOF	BOIE	Error
UART Timeout error	TO	TOIE	Error

Table continues on the next page...

Table 54-62. Interrupts (continued)

Interrupt Event	Event flag	Enable control bit	Interrupt vector
Lin state	Sync Del, Sync Field,Identifier field or Checksum Field	LSIE	Rx
Wakeup	WUF	WUIE	Rx
Data buffer full	DBFF	DBFIE	Rx
Data buffer empty	DBEF	DBEIE	Tx
Data Reception Complete	DRF	DRIE	Rx
Data transmitted	DTF	DTIE	Tx
Header received	HRF	HRIE	Rx
Header received	HRF	HRIE	Tx (if there is a filter match for Tx identifier)

The following figure shows an Interrupt Flow Diagram.

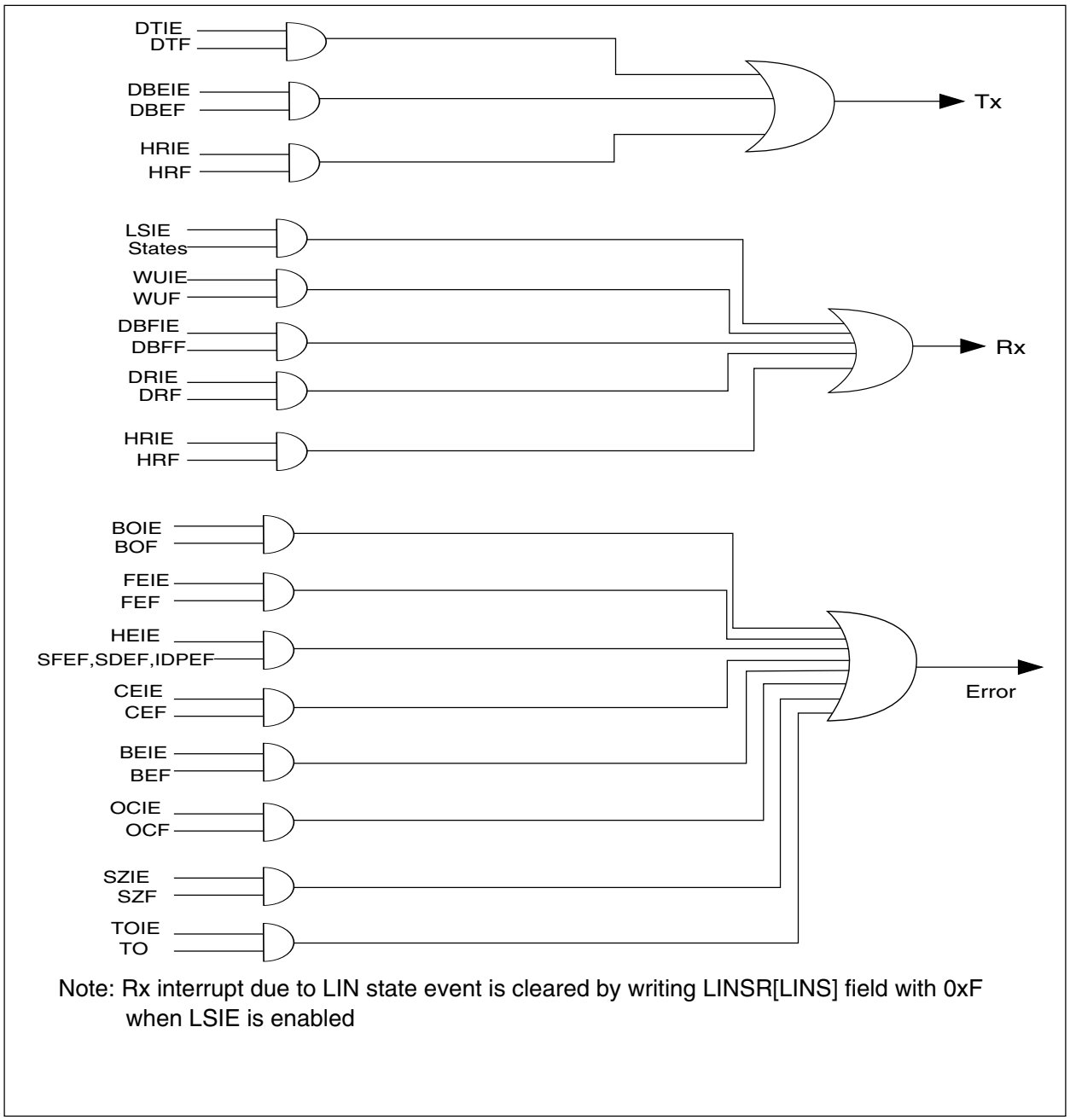


Figure 54-107. Interrupt diagram

54.6.7 LINFlexD Clock Tolerance

1. **Faster receiver tolerance:**
   In this case the receiver has a higher baud rate than the transmitter, thus the stop bit sampling starts already in the last transmitted payload bit. To ensure the correct noise and framing error free reception bit , the samples RS8, RS9, and RS10 must be located in the transmitted stop bit as shown in the following figure.

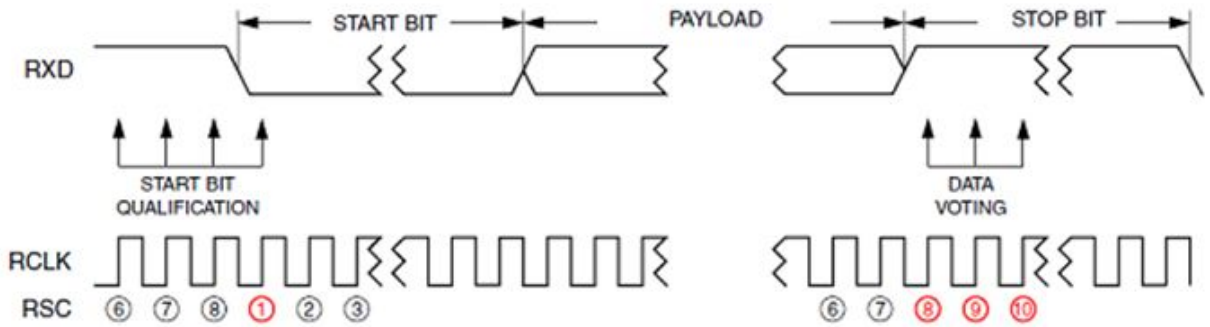


Figure 54-108. Faster receiver

2. **Slower receiver tolerance:**
   In this case the receiver has a slower baud rate than the transmitter, thus the stop bit sampling is still running while the next start bit is already transmitted. To ensure the correct noise and framing error free reception bit , the samples RS8, RS9, and RS10 must be located in the transmitted stop bit as shown in the following figure.

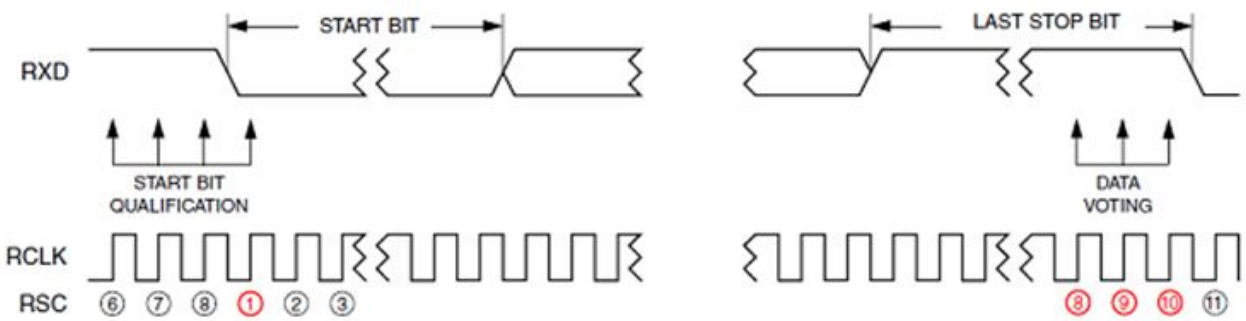


Figure 54-109. Slower receiver

Chapter 55
Reset and Boot

55.1 Introduction

The reset, boot and initialization sequences for this device provides significant new safety and security features for automotive microcontrollers and provides additional flexibility in controlling the initial configuration of the device after reset. The device must be brought out of reset with each of the system components configured for correct operation.

- Multiple processing cores, configuration and startup
- Lock step core preparation and startup
- Use of the Device Configuration Format (DCF) records
- Clock tree and PLL configuration and startup
- Configuring and entering the correct run mode

55.2 Flash memory boot overview

The device boot process uses several different blocks as it progresses. These blocks are the Power Management Controller (PMC), the Reset Generation Module (RGM), the Mode Entry Module (MC_ME), the System Status and Configuration Module (SSCM), the Boot Assist Flash (BAF), and the Self Test Configuration Unit (STCU). The reset sequence and interaction of the various blocks is illustrated at a very high level in the following figure.

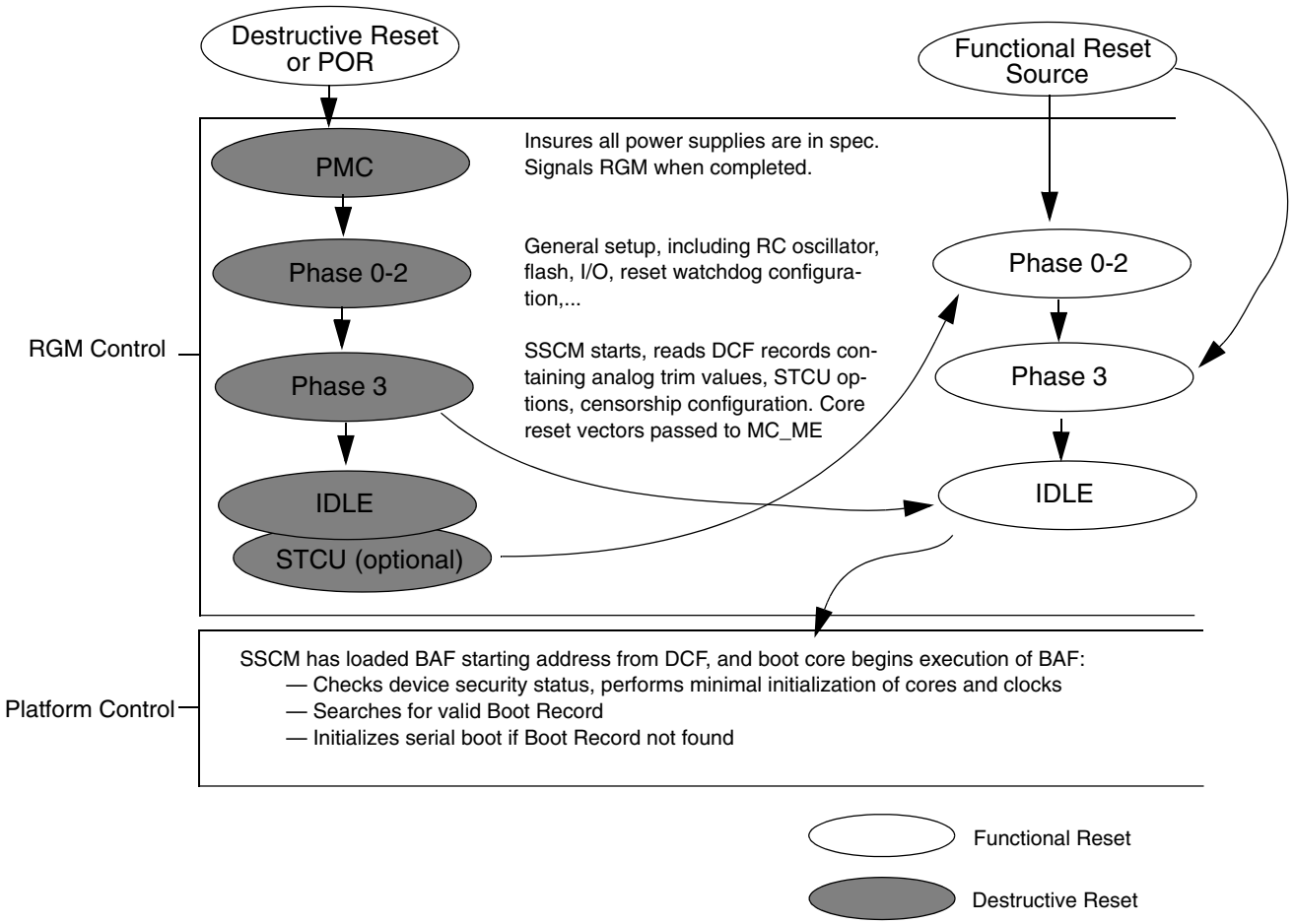


Figure 55-1. Boot sequence (high level)

See the chapter for each module for detailed descriptions of each module.

Note that the device has both destructive and functional resets. This feature allows a full system reset that is typical of a power on reset (POR), or a less destructive functional reset. The functional reset allows reset and initialization of a subset of the system resources, as opposed to a full system reset.

55.3 Device Configuration Format (DCF) records

A special area of flash memory contains programmed information in Device Configuration Format (DCF) records, which is used to control certain options at boot time. During the boot process the BAF and SSCM modules use these pre-programmed values to control certain options for device configuration. Some of these DCFs are preprogrammed at the factory and may not be modified, while others can be programmed by the end user.

The region of flash used for storing these records is called UTEST. See the DCF records chapter for details.

55.4 Reset sequence

Following section is giving an overview of the power-up sequence from power rising to the release of reset to the internal modules.

When a reset is triggered, the MC_RGM state machine is activated and proceeds through the different phases (i.e. ***PHASEn_X*** states). Each phase is associated with a particular device reset being provided to the system. A phase is completed when all corresponding phase completion gates from either the system or internal to the MC_RGM are acknowledged. The device reset associated with the phase is then released, and the state machine proceeds to the next phase up to entering the ***IDLE*** phase. During this entire process, the MC_ME state machine is held in ***RESET*** mode. Only at the end of the reset sequence, when the ***IDLE*** phase is reached, does the MC_ME enter the ***DRUN*** mode.

Alternatively, it is possible for software to configure some reset source events to be converted from a reset to either a ***SAFE*** mode request issued to the MC_ME or to an interrupt issued to the core.

The reset sequence can be observed by monitoring the *PORST* and *RESET* pins status.

The *PORST* pin is driven low by a strong internal pull-down during the ***POWERUP*** phase. It is released on ***PHASE0*** exit when all supplies have reached their specification values. After release, the PORST remains driven by a weak pull-down for safety reason, ensuring test of *PORST* pin connection. Device will not exit ***PHASE3[FUNC]*** until *PORST* is driven high.

The *RESET* pin is driven low by a strong internal pull-down during the full power-up/ reset sequence, from the ***POWERUP*** phase up to the beginning of ***PHASE3[FUNC]***. It is released during ***PHASE3[FUNC]*** when internal reset sequence is completed. Device will not exit ***PHASE3[FUNC]*** until *RESET* is driven high.

55.4.1 RSTOUT functionality

Pin PJ[5] can be configured to drive a reset signal (RSTOUT) without generating the internal reset. RSTOUT will follow the RESET pin until reconfigured in SW.

RSTOUT actively drives low during all Destructive and Functional resets and drives high at the end of reset PHASE3.

To assert RSTOUT in SW, it is necessary to reconfigure PJ[5] to be GPIO output. Writing this GPIO to '0' will assert RSTOUT without causing a reset. (If a real reset occurs while software has asserted this pin, the pin automatically switches back to RSTOUT and normal reset processing continues.

If the RSTOUT functionality is needed on PJ[5], any signal connected to PJ[5] must be able to withstand this pin driving low during reset. After reset, software can reconfigure this pin as any function available.

55.4.2 POWERUP Phase: power stabilization

The PMC module contains a set of modules for power on reset generation and supply monitoring. The device enters ***POWERUP*** phase on power on reset event. The device will remain in ***POWERUP*** phase until all supplies are in the operational range. Please refer to the PMC chapter for power on exit and details of the power supply stabilization.

55.4.3 *PHASE0* Phase: analog supply initial configuration

This phase is entered immediately from any phase on a power-on or enabled 'destructive' reset event, including PORST pin falling edge detection. The reset state machine exits ***PHASE0*** and enters ***PHASE1[DEST]*** on verification of the following:

- All enabled 'destructive' resets have been processed
- All processes that need to be done in ***PHASE0*** are completed. This includes mainly internal analog module start-up: PMC, RC-oscillator, Flash, I/Os.

During this phase, all digital modules are reset, including safety, security and test modules.

55.4.4 *PHASE1[DEST]* Phase: temporization and monitoring setup

This phase is entered either on exit from ***PHASE0*** or immediately from IDLE on an enabled, long external 'functional' reset event. The reset state machine exits ***PHASE1[DEST]*** and enters ***PHASE2[DEST]*** on verification of the following:

- All processes that need to be done in ***PHASE1[DEST]*** are completed. This includes mainly the initial configuration of the watch-dog to monitor reset process and temporization for reset signal propagation in case of reset.

55.4.5 *PHASE2[DEST]* Phase: flash initial configuration

This phase is entered on exit from ***PHASE1[DEST]***. The reset state machine exits ***PHASE2[DEST]*** and enters ***PHASE3[DEST]*** on verification of the following:

- All processes that need to be done in ***PHASE2[DEST]*** are completed

During this phase, the reset is released to the flash which can start its initial configuration. Watchdog is monitoring execution ensuring reset in case of flash configuration failure.

55.4.6 *PHASE3[DEST]* Phase: device configuration

This phase is entered on exit from ***PHASE2[DEST]***. The reset state machine exits ***PHASE3[DEST]*** and enters ***IDLE[FUNC]*** or ***IDLE[DEST]*** on verification of the following:

- All processes that need to be done in ***PHASE3[DEST]*** are completed

During this phase, the SSCM parse the DCF Records in UTEST flash and writes the data to the relevant DCF clients in the modules that require configuration. This includes:

- The trimming of the analog modules (voltage regulator, voltage detectors, I/Os, oscillator, …)
- Life cycle of the device: censorship configuration, debug security configuration, functional security configuration
- Application configuration bits: watchdog configuration, safety execution directives (self-test, LBIST/MBIST), Boot Assist Module options, Oscillator start-up during reset sequence configuration

This configuration are maintained until next internal power-on reset

At the end of ***PHASE3[DEST]***, the security status of the device has been read. If allowed by life cycle information full test mode can be entered.

In case configuration information are requesting self-test execution, the device will move to ***IDLE[DEST]***, otherwise device will directly move to ***IDLE[FUNC]***. When self-test execution is not requested, it is possible to maintain the device within ***PHASE3[DEST]*** by forcing either PORST or RESET pin low. This can be used for debug purpose to connect through JTAG to the internal debugger.

55.4.7 *IDLE[DEST]* Phase: self-test execution

This phase is entered on exit from ***PHASE3[DEST]***. The reset state machine exits ***IDLE[DEST]*** and enters ***PHASE1[FUNC]*** on verification of the following:

- Self test engine update self test completion flag and trigger associated functional reset

This phase is variable depending on self-test requirement.

During this phase, self-test control unit executes all the directives retrieved during previous phase. Dedicated watch-dog is configured to monitor self-test execution complete within expected time.

At completion of self-test, functional reset is generated and the reset state machine moves to ***PHASE1[FUNC]***.

55.4.8 *PHASE1[FUNC]* Phase: temporization and monitoring setup

This phase is entered either on exit from ***IDLE[DEST]*** or immediately from ***PHASE2[FUNC]***, ***PHASE3[FUNC]***, or ***IDLE[FUNC]*** on a non-masked external or 'functional' reset event if it has not been configured to trigger a 'short' sequence. The reset state machine exits ***PHASE1[FUNC]*** and enters ***PHASE2 [FUNC]*** on verification of the following:

- All enabled, non-shortened 'functional' resets have been processed
- All processes that need to be done in ***PHASE1[FUNC]*** are completed. This includes mainly the initial configuration of the watch-dog to monitor reset process and temporization for reset signal propagation in case of reset.

This phase is duplication of ***PHASE1[DEST]*** after self-test.

55.4.9 *PHASE2[FUNC]* Phase: flash initial configuration

This phase is entered on exit from ***PHASE1[FUNC]***. The reset state machine exits ***PHASE2[FUNC]*** and enters ***PHASE3[FUNC]*** on verification of the following:

- All processes that need to be done in ***PHASE2[DEST]*** are completed

During this phase, the reset is released to the flash which can start its initial configuration. Watchdog is monitoring execution ensuring reset in case of flash configuration failure.

This phase is duplication of ***PHASE2[DEST]*** after self-test.

55.4.10 *PHASE3[FUNC]* Phase: device configuration monitoring

This phase is entered either on exit from ***PHASE2[FUNC]*** or immediately from ***IDLE*** on an enabled, shortened external 'functional' reset event. The reset state machine exits ***PHASE3[FUNC]*** and enters ***IDLE*** on verification of the following:

- All processes that need to be done in ***PHASE3[FUNC]*** are completed. In particular PORST and RESET pin are not forced low externally.
- A minimum number of cycles have elapsed since the last enabled, shortened external 'functional' reset event

After ***PHASE3[FUNC]*** internal actions have been completed, it is possible to maintain the device within ***PHASE3[FUNC]*** by forcing either PORST or RESET pin low. This can be used for debug purpose to connect through JTAG to the internal debugger.

55.4.11 *IDLE[FUNC]* Phase

This is the final phase and is entered on exit from either ***PHASE3[DEST]*** or ***PHASE3[FUNC]***. When this phase is reached, the RGM releases control of the system to the platform and waits for new reset events that can trigger a reset sequence.

The SSCM will already have loaded the start address of the BAF code from the DCF records. As the RGM releases control of the system to the platform, CPU1 start executing at the start address of the BAF.

The state machine used to produce the reset sequence is shown in Figure 55-2.

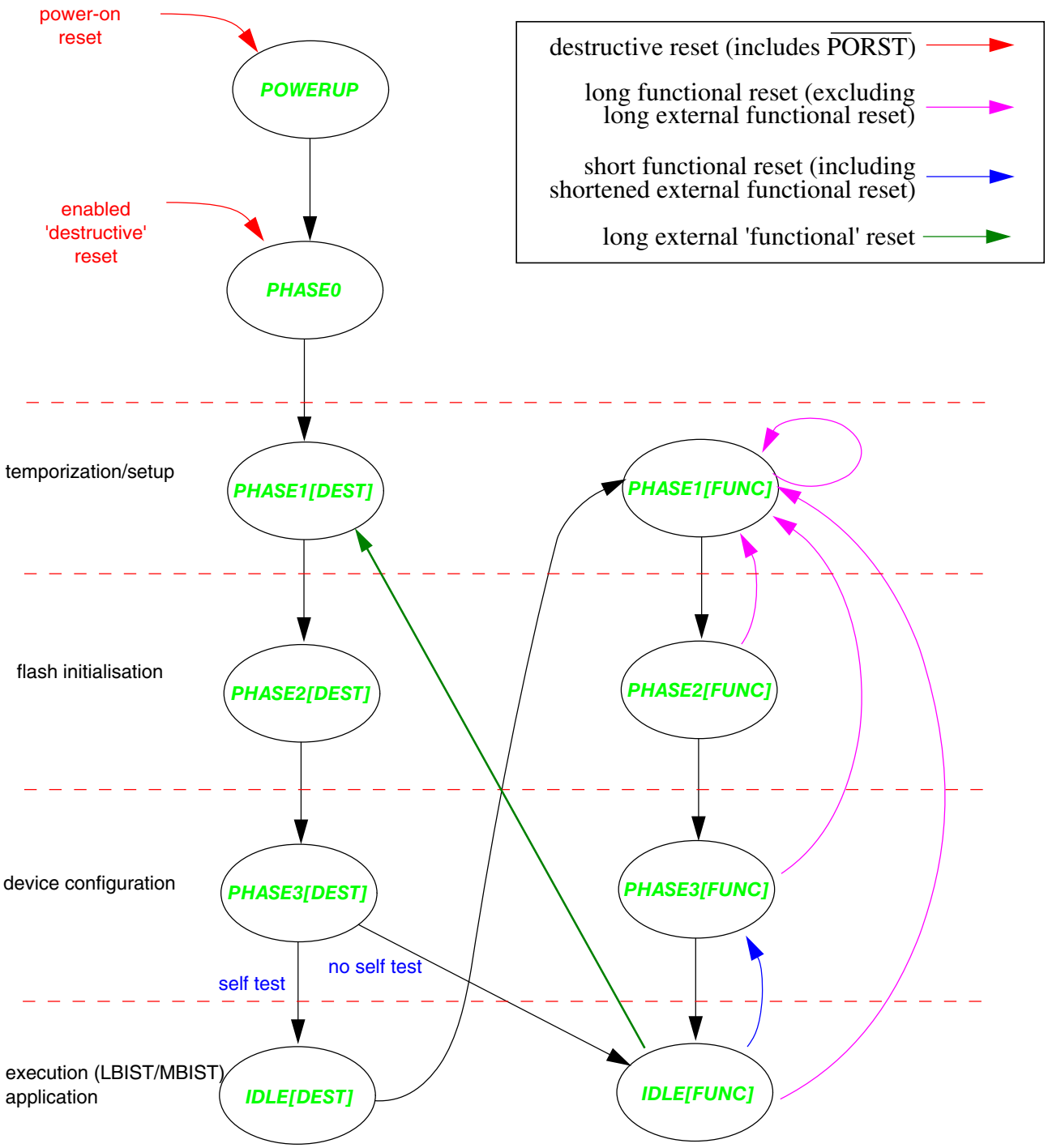
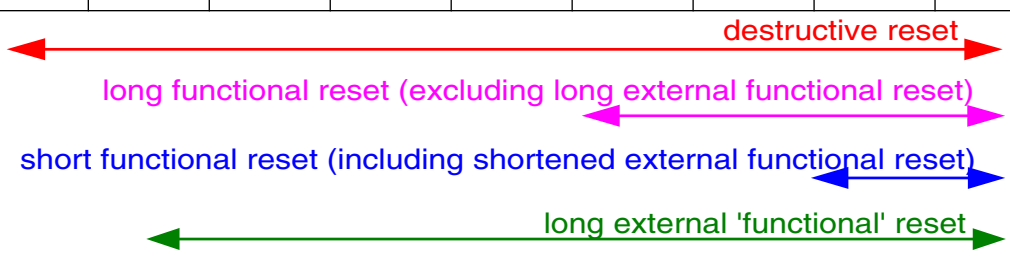


Figure 55-2. RGM state machine

55.5 Module status during reset sequence

The following table is providing status of main modules during the different reset phases:

Table 55-1. Module status during reset phases

Module	PHASE								
	P0 (RED)	P1 [DEST] (RED)	P2 [DEST] (RED)	P3 [DEST] (RED)	IDLE [DEST] (RED)	P1 [FUNC] (PURPLE)	P2 [FUNC] (PURPLE)	P3 [FUNC] (BLUE)	IDLE [FUNC] (WHITE)
Internal RCOSC	ON	ON	ON	ON	ON	ON	ON	ON	ON
Power management controller	ON	ON	ON	ON	ON	ON	ON	ON	ON
Reset management controller	ON[1]	ON	ON	ON	ON	ON	ON	ON	ON
system watchdog	RESET	RESET	ON	RESET	LBIST[2]	RESET	ON	RESET	ON[3]
flash module	RESET	RESET	ON	ON	LBIST	RESET	ON	ON	ON
OSC	RESET	RESET	RESET	ON[4]	ON[5]	ON[5]	ON[5]	ON[5]	ON[5]
DEVICE CONFIG	RESET	RESET	RESET	LOAD	HOLD	HOLD	HOLD	HOLD	HOLD
SSCM	RESET	RESET	RESET	ON	ON	RESET	RESET	ON	ON
FCCU	RESET	RESET	CONFIG	CONFIG	LBIST	RESET/ ON[5]	CONFIG/ ON	CONFIG/ ON	ON
STCU	RESET	RESET	CONFIG	CONFIG	ON	IDLE	IDLE	IDLE	IDLE[6]
pit_rti_1 timer	RESET	RESET	RESET	RESET	LBIST	IDLE/ON	IDLE/ON	IDLE/ON	IDLE/ON
other modules including cores	RESET	RESET	RESET	RESET	LBIST	RESET	RESET	RESET	ON[7]
Debug Modules	RESET	RESET	RESET	RESET	RESET	ON	ON	ON	ON
									

1. RESET only during POR.
2. STCU time-out monitoring LBIST/MBIST execution
3. Depending on flash configuration
4. Depending on flash configuration
5. FCCU remains ON during functional reset, it is instead reset in PHASE1[FUNC] if coming from IDLE[DEST]
6. Can be triggered by SW
7. After enabling from BAF

Chapter 56
Reset Generation Module (MC_RGM)

56.1 Introduction

56.1.1 Overview

The reset generation module (MC_RGM) centralizes the different reset sources and manages the reset sequence of the chip. It provides a register interface and a reset sequencer. Various registers are available to monitor and control the chip reset sequence. The reset sequencer is a state machine which controls the different phases (***PHASE0***, ***PHASE1***, ***PHASE2***, ***PHASE3***, and ***IDLE***) of the reset sequence and the reset signals generated in the system.

The following figure shows the MC_RGM block diagram.

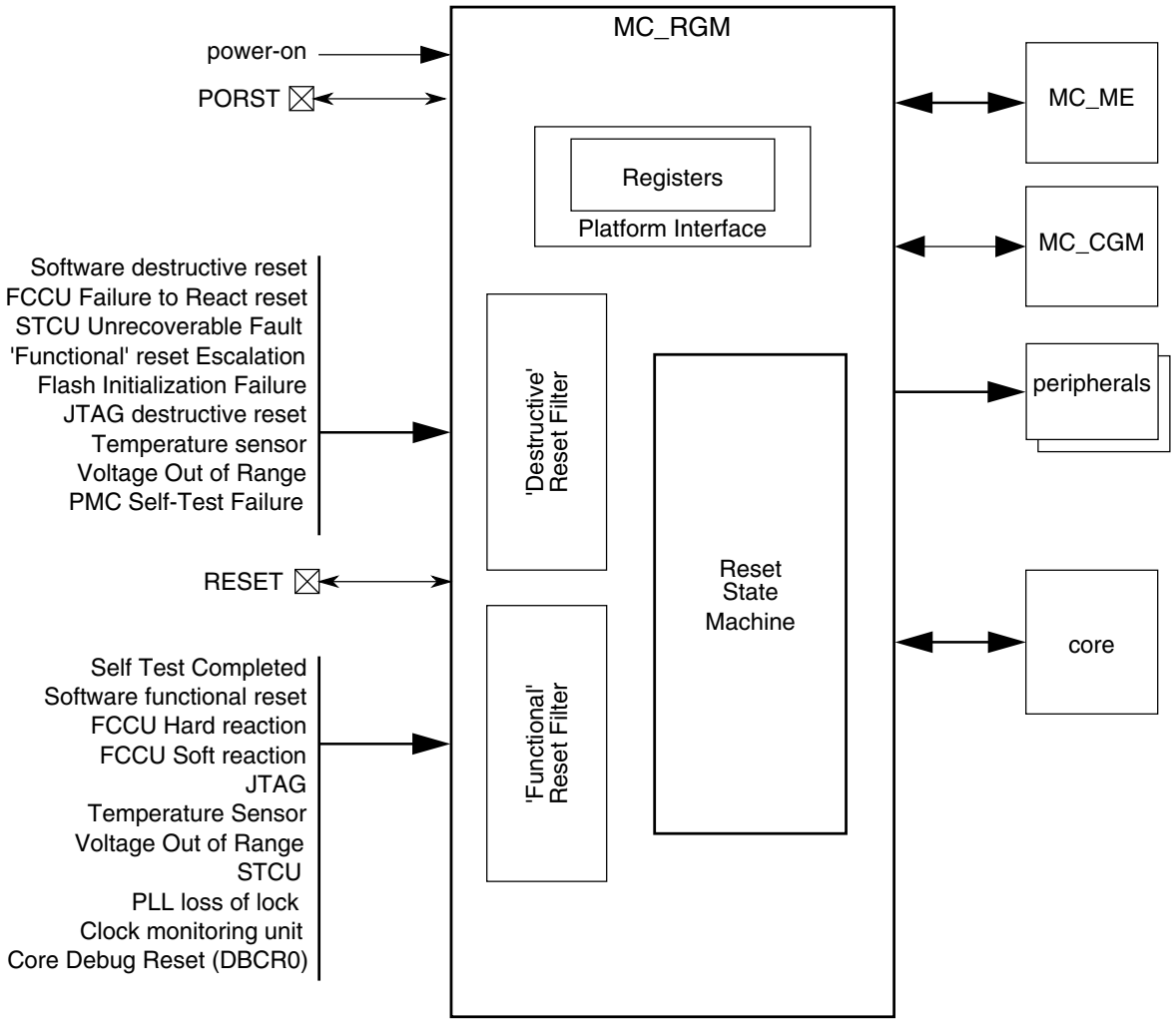


Figure 56-1. MC_RGM Block Diagram

Peripheral resets cannot be controlled individually.

56.1.2 Features

The MC_RGM contains the functionality for the following features:

- 'destructive' resets management
- 'functional' resets management
- signaling of reset events after each reset sequence (reset status flags)
- conversion of reset events to ***SAFE*** mode or interrupt request events
- short reset sequence configuration
- bidirectional reset behavior configuration

- configurable escalation of recurring 'functional' resets to 'destructive' reset
- configurable escalation of recurring 'destructive' resets to keep chip in reset state until next power-on reset

56.1.3 Reset Sources

The different reset sources are organized into two families: 'destructive' and 'functional'.

- A 'destructive' reset source is associated with an event related to a critical - usually hardware - error or dysfunction. When a 'destructive' reset event occurs, the full reset sequence is applied to the chip starting from ***PHASE0***. This resets the full chip ensuring a safe start-up state for both digital and analog modules, and the memory content must be considered to be unknown. 'Destructive' resets are
  - power-on reset
  - External PowerOn reset
  - Software destructive reset
  - FCCU Failure to React reset
  - STCU Unrecoverable Fault
  - Security reset request from SSCM
  - 'Functional' reset Escalation
  - Flash Initialization Failure
  - JTAG destructive reset
  - Temperature sensor
  - Voltage Out of Range
  - PMC self test failure reset
- A 'functional' reset source is associated with an event related to a less-critical - usually non-hardware - error or dysfunction. When a 'functional' reset event occurs, a partial reset sequence is applied to the chip starting from ***PHASE1***. In this case, most digital modules are reset normally, while the state of analog modules or specific digital modules (e.g., debug modules, flash modules) as well as the system memory content is preserved. 'Functional' resets are
  - external reset

- Self Test Completed
- Software functional reset
- FCCU Hard reaction
- FCCU Soft reaction
- JTAG functional reset
- Temperature Sensor
- Voltage Out of Range

When a reset is triggered, the MC_RGM state machine is activated and proceeds through the different phases (i.e., ***PHASEn*** states). Each phase is associated with a particular chip reset being provided to the system. A phase is completed when all corresponding phase completion gates from either the system or internal to the MC_RGM are acknowledged. The chip reset associated with the phase is then released, and the state machine proceeds to the next phase up to entering the ***IDLE*** phase. During this entire process, the MC_ME state machine is held in ***RESET*** mode. Only at the end of the reset sequence, when the ***IDLE*** phase is reached, does the MC_ME enter the ***DRUN*** mode.

Alternatively, it is possible for software to configure some reset source events to be converted from a reset to either a ***SAFE*** mode request issued to the MC_ME or to an interrupt issued to the core (see Section "'Destructive' Event Alternate Request Register (RGM_DEAR)" for 'destructive' resets and Section "'Functional' Event Reset Disable Register (RGM_FERD)" and Section "'Functional' Event Alternate Request Register (RGM_FEAR)" for 'functional' resets).

56.2 External Signal Description

The MC_RGM interfaces to the bidirectional reset pin RESET.

Table 56-1. MC_RGM Reset Phase Duration Dependency on safe_clock_busy During Functional Mode

Reset Phase Transition			Duration Counter Start
To	From	safe_clock_busy Already Deasserted During 'From' Phase	
PHASE0	power-on	-	immediate
	PHASE1	yes	immediate
		no	on safe_clock_busy deassertion
	PHASE2	-	immediate
	PHASE3	yes	immediate
		no	on safe_clock_busy deassertion
	IDLE	-	on safe_clock_busy deassertion
PHASE1	***PHASE0***	-	immediate
	PHASE2	-	immediate
	PHASE3	yes	immediate
		no	on safe_clock_busy deassertion
	IDLE	-	on safe_clock_busy deassertion
PHASE2	***PHASE1***	-	immediate
PHASE3	***PHASE2***	-	immediate
	IDLE	-	on safe_clock_busy deassertion
IDLE	***PHASE3***	-	-

56.3 MC_RGM Registers

The following subsections show the MC_RGM registers memory map and the detailed definition of each register.

The MC_ME contains registers for:

- mode selection and status reporting
- mode configuration
- mode transition interrupts status and mask control
- scalable number of peripheral sub-mode selection and status reporting
- scalable number of core enabling and disabling control and status reporting

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the ME_RUN_PC0 register may be accessed as a word at address offset 0x0080, as a half-word at address offset 0x0082, or as a byte at address offset 0x0083.

Some fields may be read-only, and their reset value of ‘1’ or ‘0’ and the corresponding behavior cannot be changed.

MC_RGM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	‘Destructive’ Event Status (MC_RGM_DES)	32	R/W	0000_0001h	56.3.1/2490
10	‘Destructive’ Event Reset Disable (MC_RGM_DERD)	32	R/W	0000_0000h	56.3.2/2492
20	‘Destructive’ Event Alternate Request (MC_RGM_DEAR)	32	R/W	0000_0000h	56.3.3/2494
30	‘Destructive’ Bidirectional Reset Enable (MC_RGM_DBRE)	32	R/W	0000_0400h	56.3.4/2496
300	‘Functional’ Event Status (MC_RGM_FES)	32	R/W	0000_0000h	56.3.5/2498
310	‘Functional’ Event Reset Disable (MC_RGM_FERD)	32	R/W	0000_0000h	56.3.6/2500
320	‘Functional’ Event Alternate Request (MC_RGM_FEAR)	32	R/W	0000_0000h	56.3.7/2501
330	‘Functional’ Bidirectional Reset Enable (MC_RGM_FBRE)	32	R/W	0180_0468h	56.3.8/2502
340	‘Functional’ Event Short Sequence (MC_RGM_FESS)	32	R/W	0000_0040h	56.3.9/2504
604	Functional Reset Threshold (MC_RGM_FRET)	8	R/W	0Fh	56.3.10/ 2506
608	‘Destructive’ Reset Threshold (MC_RGM_DRET)	8	R/W	00h	56.3.11/ 2507

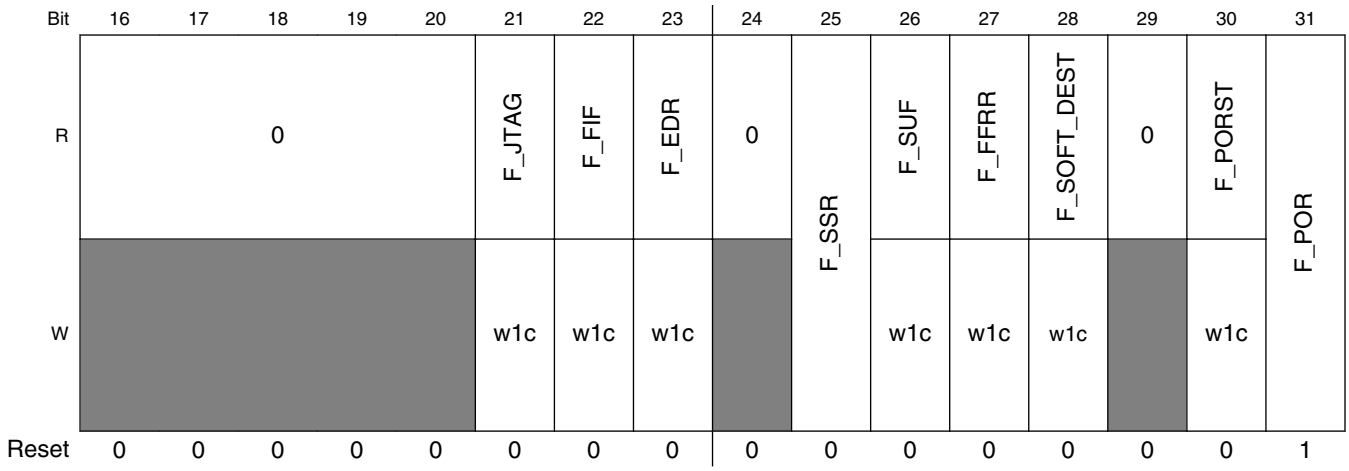
56.3.1 ‘Destructive’ Event Status (MC_RGM_DES)

This register contains the status of the ‘destructive’ reset sources, including those configured to not generate a reset sequence. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Register bits are cleared by writing ‘1’.

This register is reset only on power-on.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved						F_PMC	F_VOR	F_TSR	0						
W							w1c	w1c	w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



MC_RGM_DES field descriptions

Field	Description
0–5 Reserved	This field is reserved.
6 F_PMC	Flag for PMC self test failure after boot up (POR) This reset is generated only in case of PMC self test failure in hardware initiated self test run of PMC after boot up. This reset can be disabled by device configuration in UTEST area. 0 No Voltage Out of Range event has occurred since either the last clear or the last power-on reset assertion 1 A Voltage Out of Range event has occurred
7 F_VOR	Flag for Voltage Out of Range 0 No Voltage Out of Range event has occurred since either the last clear or the last power-on reset assertion 1 A Voltage Out of Range event has occurred
8 F_TSR	Flag for Temperature sensor 0 No Temperature sensor event has occurred since either the last clear or the last power-on reset assertion 1 A Temperature sensor event has occurred
9–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 F_JTAG	Flag for JTAG destructive reset 0 No JTAG destructive reset event has occurred since either the last clear or the last power-on reset assertion 1 A JTAG destructive reset event has occurred
22 F_FIF	Flag for Flash Initialization Failure 0 No Flash Initialization Failure event has occurred since either the last clear or the last power-on reset assertion 1 A Flash Initialization Failure event has occurred
23 F_EDR	Flag for 'Functional' reset Escalation

Table continues on the next page...

MC_RGM_DES field descriptions (continued)

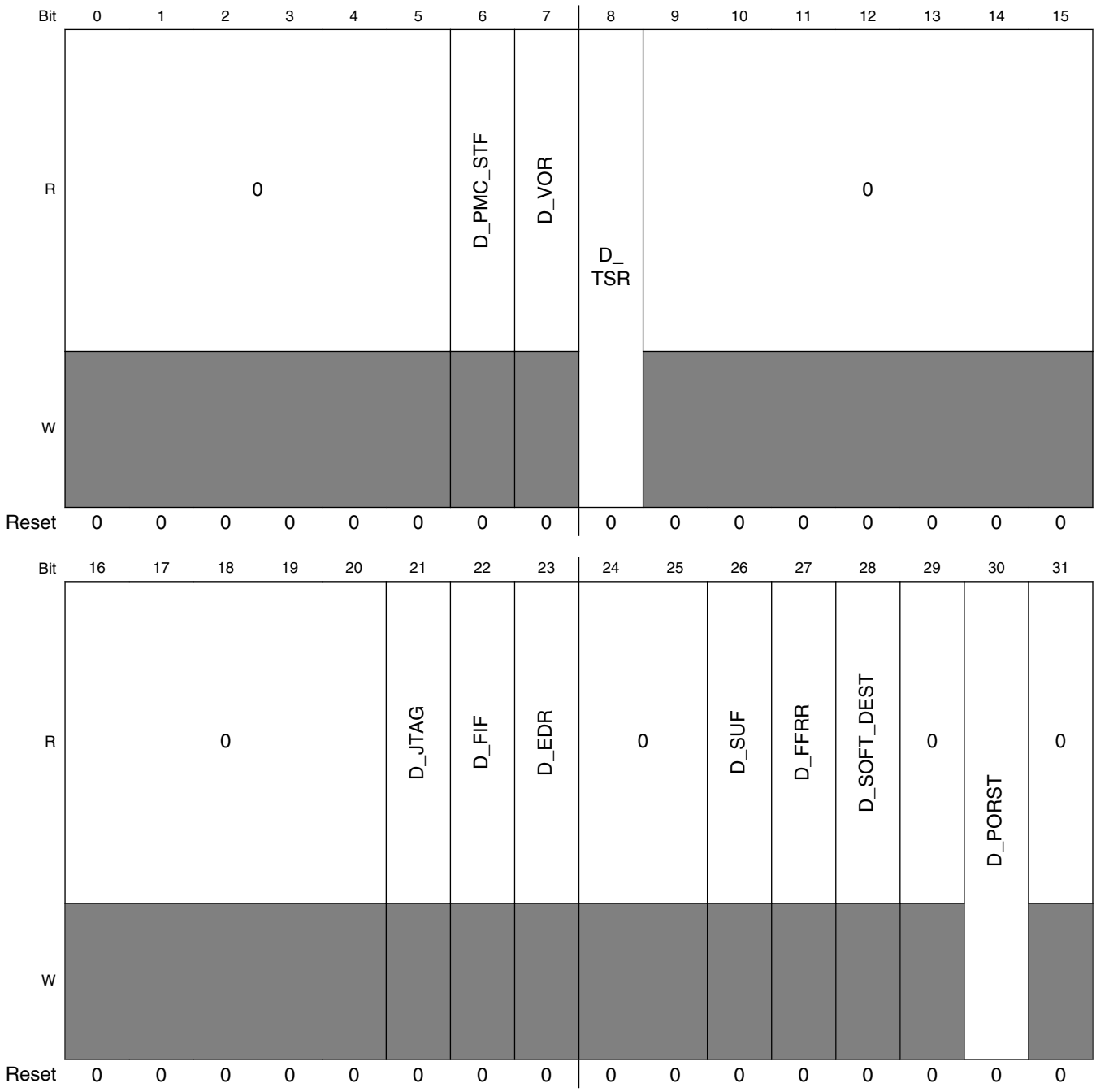
Field	Description
	0 No 'Functional' reset Escalation event has occurred since either the last clear or the last power-on reset assertion 1 A 'Functional' reset Escalation event has occurred
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 F_SSR	Flag for SSCM security reset Active high security reset request. Once asserted, it is deasserted only when the SSCM gets its module reset. It is asserted on ECC errors and life cycle manipulation. It is also generated on fgf_enc* signals mismatch with the flash block select if parameter EN_ENC_ERR_GEN is '1'.
26 F_SUF	Flag for STCU Unrecoverable Fault 0 No STCU Unrecoverable Fault event has occurred since either the last clear or the last power-on reset assertion 1 An STCU Unrecoverable Fault event has occurred. This reset is generated only during offline self test run, during the device initialization after POR. This can be disabled by device configuration in UTEST.
27 F_FFRR	Flag for FCCU Failure to React reset 0 No FCCU Failure to React reset event has occurred since either the last clear or the last power-on reset assertion 1 An FCCU Failure to React reset event has occurred
28 F_SOFT_DEST	Flag for Software destructive reset 0 No Software destructive reset event has occurred since either the last clear or the last power-on reset assertion 1 A Software destructive reset event has occurred
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 F_PORST	Flag for External PowerOn reset 0 No External PowerOn reset event has occurred since either the last clear or the last power-on reset assertion 1 An External PowerOn reset event has occurred
31 F_POR	Flag for Power-On reset 0 No power-on event has occurred since the last clear 1 A power-on event has occurred

56.3.2 'Destructive' Event Reset Disable (MC_RGM_DERD)

This register provides dedicated bits to disable particular 'destructive' reset sources. When a 'destructive' reset source is disabled, the associated 'destructive' event will trigger either a SAFE mode request or an interrupt request (see Section, 'Destructive' Event Alternate Request (RGM_DEAR)). It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Each byte can be written only once after a 'destructive' or power-on reset.

This register is reset only on power-on.

Address: 0h base + 10h offset = 10h



MC_RGM_DERD field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 D_PMC_STF	Disable PMC Self_Test Failure

Table continues on the next page...

MC_RGM_DERD field descriptions (continued)

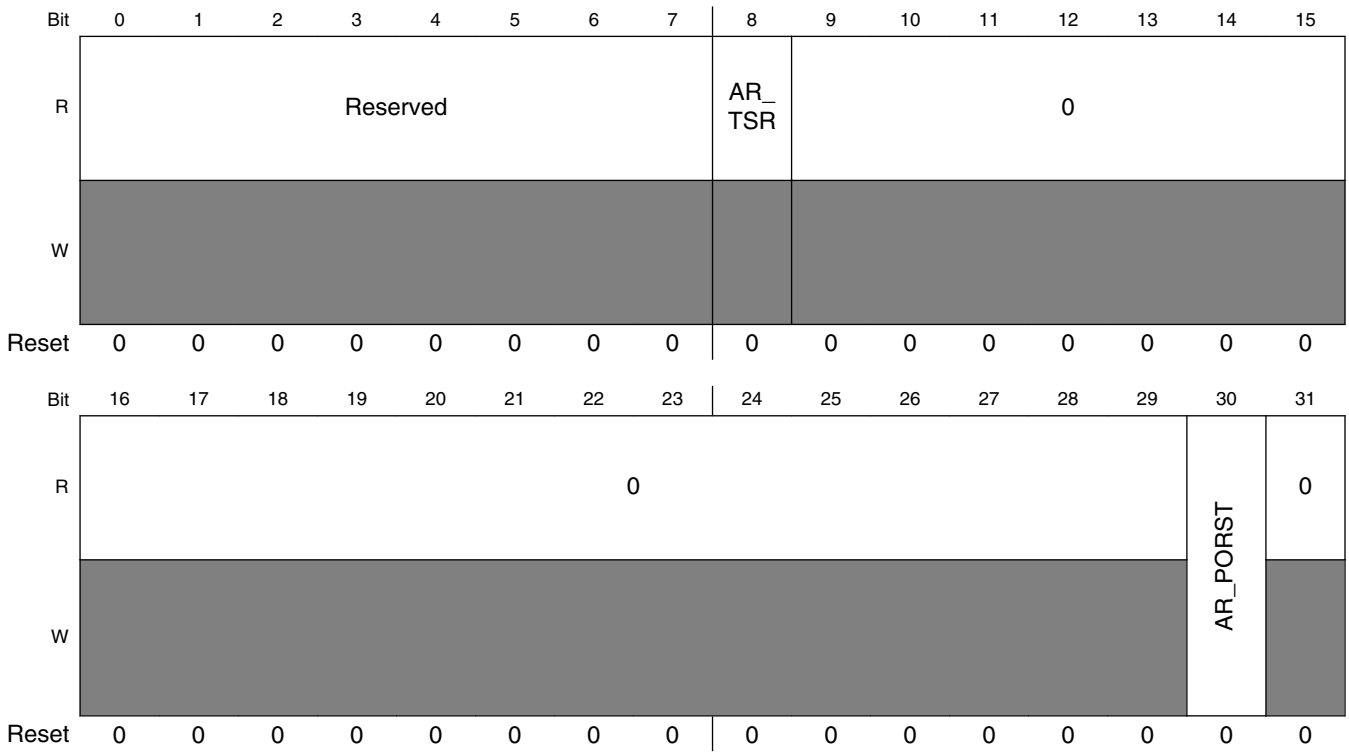
Field	Description
	0 A PMC selt test failure ‘destructive’ reset event triggers a reset sequence
7 D_VOR	Disable Voltage Out of Range 0 A Voltage Out of Range ‘destructive’ reset event triggers a reset sequence
8 D_TSR	Disable Temperature sensor 0 A Tempearature sensor ‘destructive’ reset event triggers a reset sequence 1 A Temperature sensor ‘destructive’ event generates a SAFE mode
9–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 D_JTAG	Disable JTAG destructive reset 0 A JTAG ‘destructive’ reset event triggers a reset sequence
22 D_FIF	Disable Flash Initialization Failure 0 A Flash Initialization Failure ‘destructive’ reset event triggers a reset sequence
23 D_EDR	Disable ’Functional’ reset Escalation 0 A ’Functional’ reset Escalation ‘destructive’ reset event triggers a reset sequence
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 D_SUF	Disable STCU Unrecoverable Fault 0 An STCU Unrecoverable Fault ‘destructive’ reset event triggers a reset sequence
27 D_FFRR	Disable FCCU Failure to React reset 0 An FCCU Failure to React reset ‘destructive’ reset event triggers a reset sequence
28 D_SOFT_DEST	Disable Software destructive reset 0 A Software ‘destructive’ reset event triggers a reset sequence
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 D_PORST	Disable External PowerOn reset 0 An External PowerOn reset ‘destructive’ reset event triggers a reset sequence 1 An External PowerOn reset ‘destructive’ event generates either a SAFE mode or an interrupt request depending on the values in RGM_DEAR register
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

56.3.3 ‘Destructive’ Event Alternate Request (MC_RGM_DEAR)

This register defines an alternate request to be generated when a reset on a 'destructive' event has been disabled. The alternate request can be either a SAFE mode request to MC_ME or an interrupt request to the system. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

This register is reset only on power-on.

Address: 0h base + 20h offset = 20h



MC_RGM_DEAR field descriptions

Field	Description
0–7 Reserved	This field is reserved.
8 AR_TSR	Alternate Requests for Temperature sensor 'Destructive' Reset 0 Generate a SAFE mode request on a Temperature sensor 'destructive' reset event if the reset is disabled.
9–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 AR_PORST	Alternate Requests for external POR ‘Destructive’ Reset 0 Generate a SAFE mode request on a Temperature sensor ‘destructive’ reset event if the reset is disabled 1 Generate an interrupt request on a Temperature sensor ‘destructive’ reset event if the reset is disabled
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

56.3.4 ‘Destructive’ Bidirectional Reset Enable (MC_RGM_DBRE)

This register enables the generation of an external reset on 'destructive' reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

This register is reset only on power-on.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0						BE_PMC_STF	BE_VOR	BE_TSR	0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					BE_JTAG	BE_FIF	BE_EDR	0		BE_SUF	BE_FFRR	BE_SOFT_DEST	0	BE_PORST	BE_POR
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

MC_RGM_DBRE field descriptions

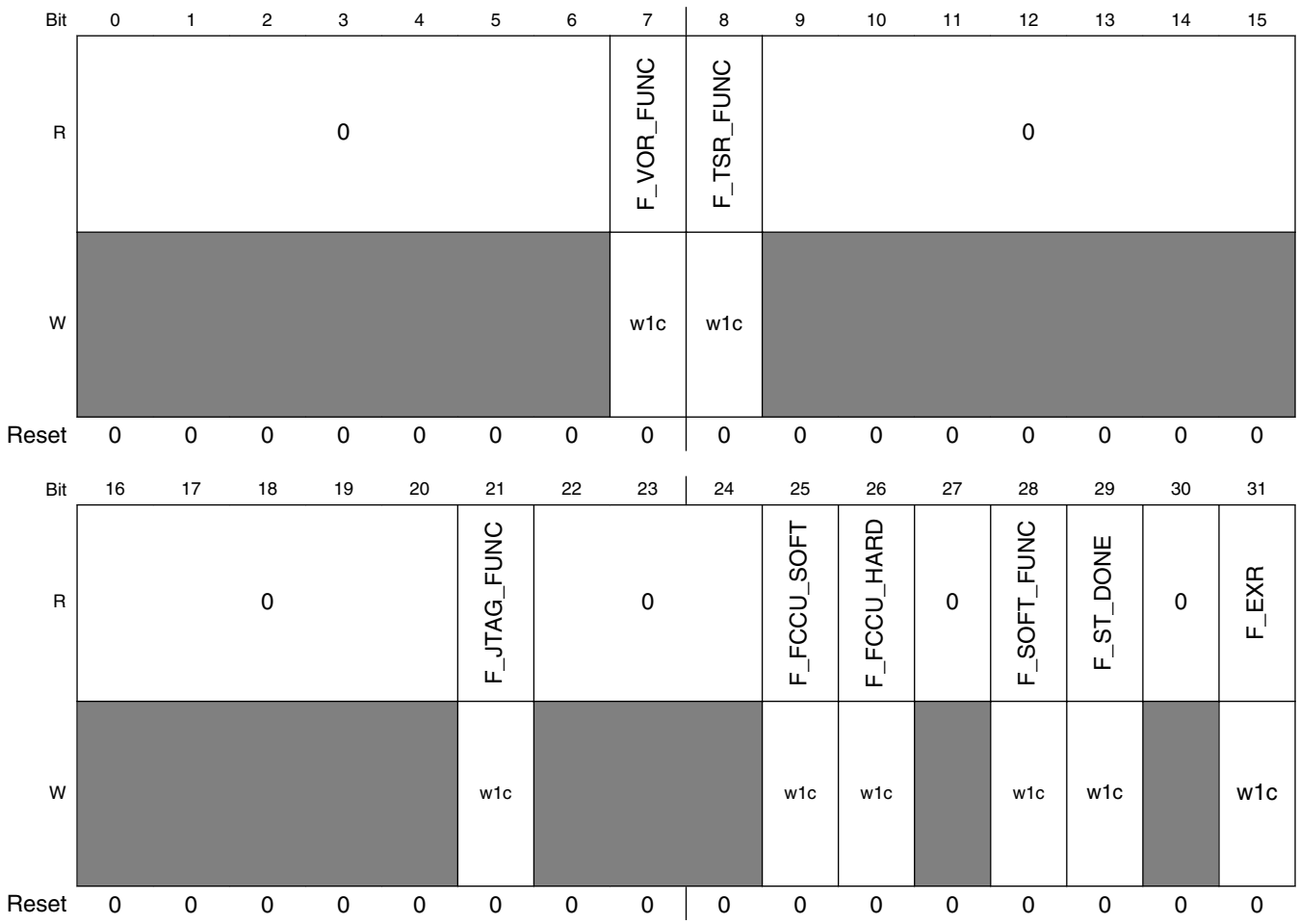
Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 BE_PMC_STF	Bidirectional Reset Enables for PMC Selt Test Failure 'Destructive' Reset 0 RESET_B is asserted on a PMC Selt Test Failure ‘destructive’ reset event if the reset is enabled
7 BE_VOR	Bidirectional Reset Enables for Voltage Out of Range ‘Destructive’ Reset 0 RESET_B is asserted on a Voltage Out of Range ‘destructive’ reset event if the reset is enabled
8 BE_TSR	Bidirectional Reset Enables for Temperature sensor ‘Destructive’ Reset 0 RESET_B is asserted on a Temperature sensor ‘destructive’ reset event if the reset is enabled
9–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 BE_JTAG	Bidirectional Reset Enables for JTAG destructive reset ‘Destructive’ Reset 0 RESET_B is asserted on a JTAG ‘destructive’ reset event if the reset is enabled 1 RESET_B is not asserted on a JTAG destructive reset event
22 BE_FIF	Bidirectional Reset Enables for Flash Initialization Failure ‘Destructive’ Reset 0 RESET_B is asserted on a Flash Initialization Failure ‘destructive’ reset event if the reset is enabled
23 BE_EDR	Bidirectional Reset Enables for ’Functional’ reset Escalation ‘Destructive’ Reset 0 RESET_B is asserted on a ’Functional’ reset Escalation ‘destructive’ reset event if the reset is enabled
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 BE_SUF	Bidirectional Reset Enables for STCU Unrecoverable Fault ‘Destructive’ Reset 0 RESET_B is asserted on an STCU Unrecoverable Fault ‘destructive’ reset event if the reset is enabled
27 BE_FFRR	Bidirectional Reset Enables for FCCU Failure to React reset ‘Destructive’ Reset 0 RESET_B is asserted on an FCCU Failure to React reset ‘destructive’ reset event if the reset is enabled
28 BE_SOFT_DEST	Bidirectional Reset Enables for Software ‘Destructive’ Reset 0 RESET_B is asserted on a Software destructive reset ‘destructive’ reset event if the reset is enabled
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 BE_PORST	Bidirectional Reset Enables for External PowerOn reset ‘Destructive’ Reset 0 RESET_B is asserted on an External PowerOn reset ‘destructive’ reset event if the reset is enabled
31 BE_POR	Bidirectional Reset Enables for Power On ‘Destructive’ Reset 0 RESET_B is asserted on a Power On ‘destructive’ reset event if the reset is enabled

56.3.5 ‘Functional’ Event Status (MC_RGM_FES)

This register contains the status of the 'functional' reset sources, including those configured to not generate a reset sequence. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Register bits are cleared by writing '1' if the triggering event has already been cleared at the source.

This register is reset only on power-on.

Address: 0h base + 300h offset = 300h



MC_RGM_FES field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 F_VOR_FUNC	Flag for Voltage Out of Range ‘Functional’ Reset

Table continues on the next page...

MC_RGM_FES field descriptions (continued)

Field	Description
	0 No Voltage Out of Range ‘Functional’ Reset event has occurred since either the last clear or the last ‘destructive’ reset assertion 1 A Voltage Out of Range ‘Functional’ Reset event has occurred
8 F_TSR_FUNC	Flag for Temperature Sensor ‘Functional’ Reset 0 No Temperature Sensor ‘Functional’ Reset event has occurred since either the last clear or the last ‘destructive’ reset assertion 1 A Temperature Sensor ‘Functional’ Reset event has occurred
9–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 F_JTAG_FUNC	Flag for JTAG ‘Functional’ Reset 0 No JTAG ‘Functional’ Reset event has occurred since either the last clear or the last ‘destructive’ reset assertion 1 A JTAG ‘Functional’ Reset event has occurred
22–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 F_FCCU_SOFT	Flag for FCCU Soft reaction ‘Functional’ Reset 0 No FCCU Soft reaction ‘Functional’ Reset event has occurred since either the last clear or the last ‘destructive’ reset assertion 1 An FCCU Soft reaction ‘Functional’ Reset event has occurred
26 F_FCCU_HARD	Flag for FCCU Hard reaction ‘Functional’ Reset 0 No FCCU Hard reaction ‘Functional’ Reset event has occurred since either the last clear or the last ‘destructive’ reset assertion 1 An FCCU Hard reaction ‘Functional’ Reset event has occurred
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 F_SOFT_FUNC	Flag for Software functional reset 0 No Software functional reset event has occurred since either the last clear or the last ‘destructive’ reset assertion 1 A Software functional reset event has occurred
29 F_ST_DONE	Flag for Self Test Completed ‘Functional’ Reset 0 No Self Test Completed event has occurred since either the last clear or the last ‘destructive’ reset assertion 1 A Self Test Completed event has occurred
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 F_EXR	Flag for External Reset 0 No external reset event has occurred since either the last clear or the last power-on reset assertion 1 An external reset event has occurred

56.3.6 ‘Functional’ Event Reset Disable (MC_RGM_FERD)

This register provides dedicated bits to disable 'functional' reset sources.When a 'functional' reset source is disabled, the associated 'functional' event will trigger either a SAFE mode request or an interrupt request (see Section 'Functional' Event Alternate Request (RGM_FEAR)). It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Each byte can be written only once after a 'destructive' or power-on reset.

This register is reset only on power-on and any ‘destructive’ reset.

Address: 0h base + 310h offset = 310h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							D_VOR_FUNC	D_TSR_FUNC	0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					D_JTAG_FUNC	0			D_FCCU_SOFT	D_FCCU_HARD	0	D_SOFT_FUNC	D_ST_DONE	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_RGM_FERD field descriptions

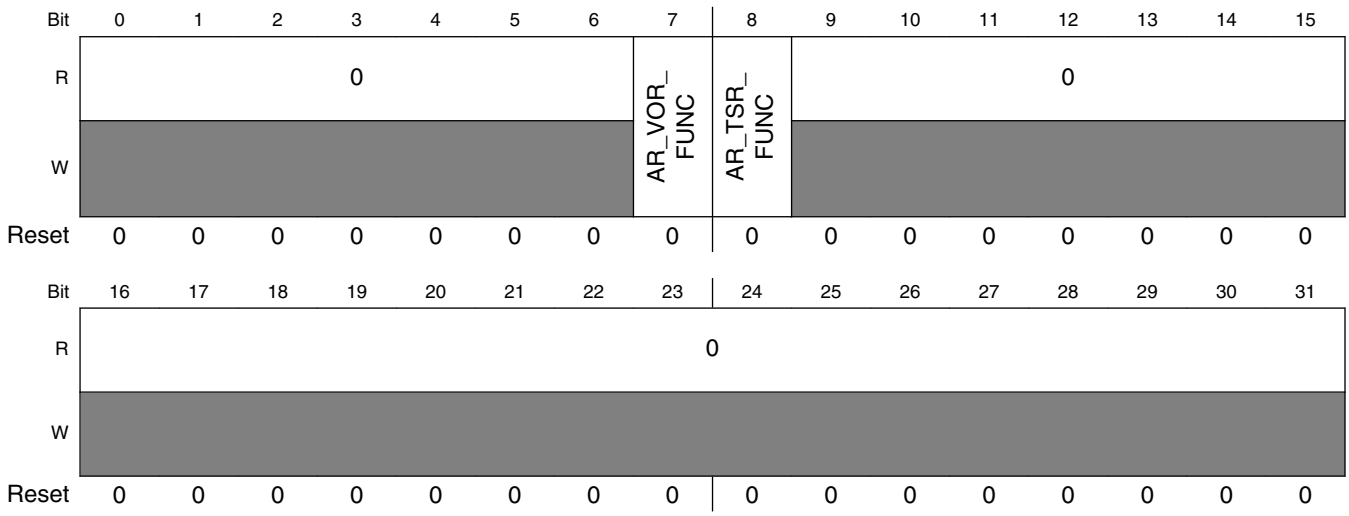
Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 D_VOR_FUNC	Disable Voltage Out of Range 'Functional' Reset 0 A Voltage Out of Range ‘functional’ reset event triggers a reset sequence 1 A Voltage Out of Range ‘functional’ reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_VOR_FUNC
8 D_TSR_FUNC	Disable Temperature Sensor ‘Functional’ Reset 0 A Temperature Sensor ‘functional’ reset event triggers a reset sequence 1 A Temperature Sensor ‘functional’ reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_TSR_FUNC
9–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 D_JTAG_FUNC	Disable JTAG 'Functional' Reset 0 A JTAG ‘functional’ reset event triggers a reset sequence
22–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 D_FCCU_SOFT	Disable FCCU Soft reaction 0 An FCCU Soft reaction event triggers a reset sequence
26 D_FCCU_HARD	Disable FCCU Hard reaction 0 An FCCU Hard reaction event triggers a reset sequence
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 D_SOFT_FUNC	Disable Software functional reset 0 A Software functional reset event triggers a reset sequence
29 D_ST_DONE	Disable Self Test Completed 0 A Self Test Completed event triggers a reset sequence
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

56.3.7 ‘Functional’ Event Alternate Request (MC_RGM_FEAR)

This register defines an alternate request to be generated when a reset on a 'functional' event has been disabled. The alternate request can be either a SAFE mode request to MC_ME or an interrupt request to the system. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

This register is reset only on power-on and any ‘destructive’ reset.

Address: 0h base + 320h offset = 320h



MC_RGM_FEAR field descriptions

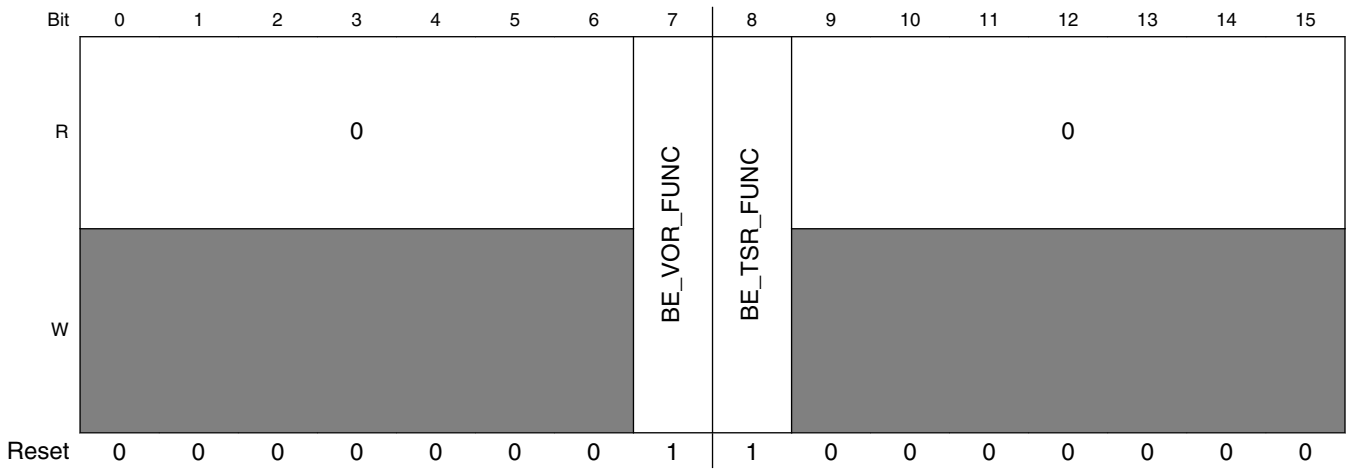
Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 AR_VOR_FUNC	Alternate Request for Voltage Out of Range ‘Functional’ Reset 0 Generate a SAFE mode request on a Voltage Out of Range ‘Functional’ Reset event if the reset is disabled 1 Generate an interrupt request on a Voltage Out of Range ‘Functional’ Reset event if the reset is disabled
8 AR_TSR_FUNC	Alternate Request for Temperature Sensor ‘Functional’ Reset 0 Generate a SAFE mode request on a Temperature Sensor ‘Functional’ Reset event if the reset is disabled 1 Generate an interrupt request on a Temperature Sensor ‘Functional’ Reset event if the reset is disabled
9–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

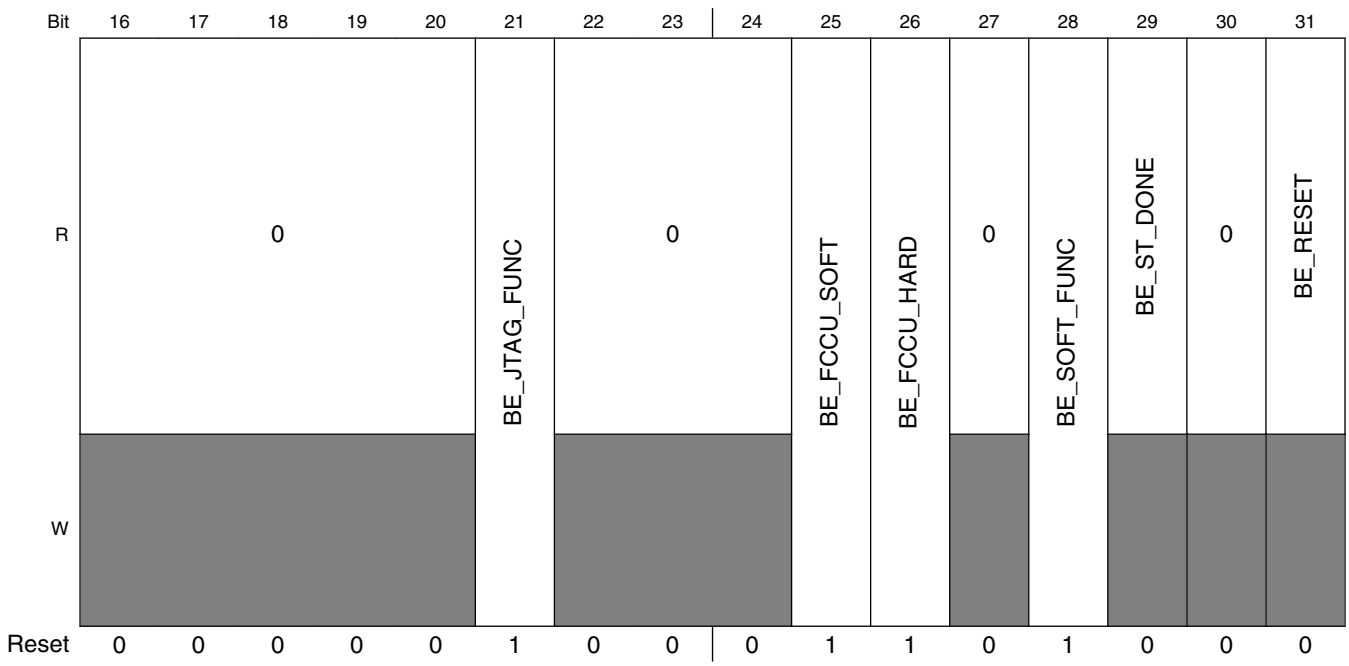
56.3.8 ‘Functional’ Bidirectional Reset Enable (MC_RGM_FBRE)

This register enables the generation of an external reset on 'functional' reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read in user mode.

This register is reset only on power-on and any ‘destructive’ reset.

Address: 0h base + 330h offset = 330h





MC_RGM_FBRE field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 BE_VOR_FUNC	Bidirectional Reset Enable for Voltage Out of Range ‘Functional’ Reset 0 RESET_B is asserted on a Voltage Out of Range ‘Functional’ Reset event if the reset is enabled 1 RESET_B is not asserted on a Voltage Out of Range ‘Functional’ Reset event
8 BE_TSR_FUNC	Bidirectional Reset Enable for Temperature Sensor ‘Functional’ Reset 0 RESET_B is asserted on a Temperature Sensor ‘Functional’ Reset event if the reset is enabled 1 RESET_B is not asserted on a Temperature Sensor ‘Functional’ Reset event
9–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_RGM_FBRE field descriptions (continued)

Field	Description
21 BE_JTAG_FUNC	Bidirectional Reset Enable for JTAG 'Functional' Reset 0 RESET_B is asserted on a JTAG 'Functional' Reset event if the reset is enabled 1 RESET_B is not asserted on a JTAG 'Functional' Reset event
22–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 BE_FCCU_SOFT	Bidirectional Reset Enable for FCCU Soft reaction 0 RESET_B is asserted on an FCCU Soft reaction event if the reset is enabled 1 RESET_B is not asserted on an FCCU Soft reaction event
26 BE_FCCU_ HARD	Bidirectional Reset Enable for FCCU Hard reaction 0 RESET_B is asserted on an FCCU Hard reaction event if the reset is enabled 1 RESET_B is not asserted on an FCCU Hard reaction event
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 BE_SOFT_FUNC	Bidirectional Reset Enable for Software functional reset 0 RESET_B is asserted on a Software functional reset event if the reset is enabled 1 RESET_B is not asserted on a Software functional reset event
29 BE_ST_DONE	Bidirectional Reset Enable for Self Test Completed 'Functional' Reset 0 RESET_B is asserted on a Self Test Completed 'Functional' Reset event if the reset is enabled 1 RESET_B is not asserted on a Self Test Completed 'Functional' Reset event
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 BE_RESET	Bidirectional Reset Enable for RESET External Reset 0 RESET_B is asserted on an external reset event if the reset is enabled 1 RESET_B is not asserted on an external reset event

56.3.9 'Functional' Event Short Sequence (MC_RGM_FESS)

This register defines which reset sequence will be done when a 'functional' reset sequence is triggered. The 'functional' reset sequence can either start from PHASE1 or from PHASE3, skipping PHASE1 and PHASE2.

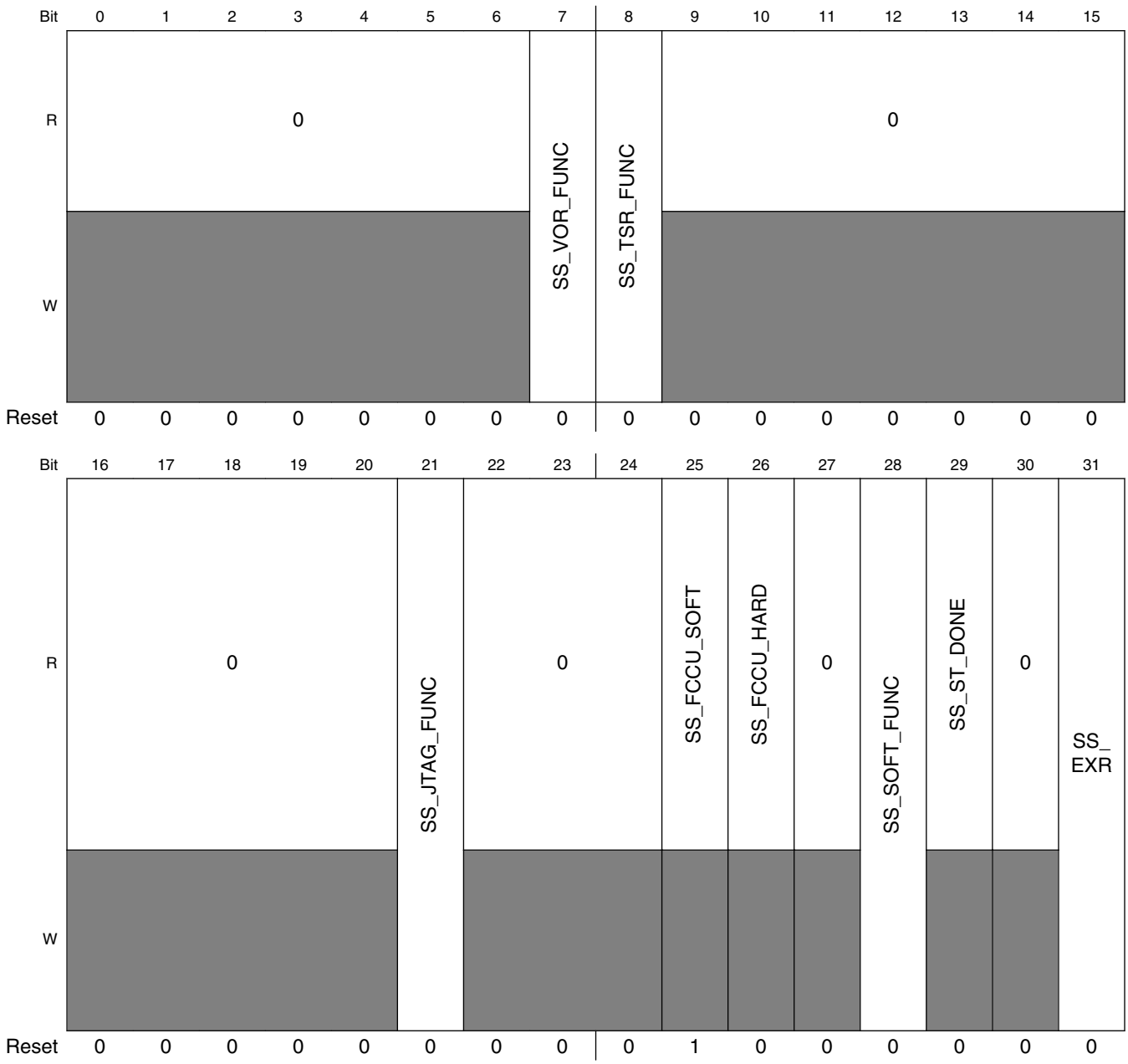
NOTE

This could be useful for a fast reset sequence, for example to skip flash reset.

It can be accessed as read/write in either supervisor mode or test mode. It can be accessed as read-only in user mode.

This register is reset only on power-on and any 'destructive' reset.

Address: 0h base + 340h offset = 340h



MC_RGM_FESS field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SS_VOR_FUNC	Short Sequence for Voltage Out of Range ‘Functional’ Reset 0 The reset sequence triggered by a Voltage Out of Range ‘Functional’ Reset event will start from PHASE1 1 The reset sequence triggered by a Voltage Out of Range ‘Functional’ Reset event will start from PHASE3, skipping PHASE1 and PHASE2

Table continues on the next page...

MC_RGM_FESS field descriptions (continued)

Field	Description
8 SS_TSR_FUNC	Short Sequence for Temperature Sensor 'Functional' Reset 0 The reset sequence triggered by a Temperature Sensor 'Functional' Reset event will start from PHASE1 1 The reset sequence triggered by a Temperature Sensor 'Functional' Reset event will start from PHASE3, skipping PHASE1 and PHASE2
9–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 SS_JTAG_FUNC	Short Sequence for JTAG 'Functional' Reset 0 The reset sequence triggered by a JTAG 'Functional' Reset event will start from PHASE1 1 The reset sequence triggered by a JTAG 'Functional' Reset event will start from PHASE3, skipping PHASE1 and PHASE2
22–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 SS_FCCU_SOFT	Short Sequence for FCCU Soft reaction 0 The reset sequence triggered by an FCCU Soft reaction event will start from PHASE1
26 SS_FCCU_ HARD	Short Sequence for FCCU Hard reaction 0 The reset sequence triggered by an FCCU Hard reaction event will start from PHASE1
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 SS_SOFT_FUNC	Short Sequence for Software functional reset 0 The reset sequence triggered by a Software functional reset event will start from PHASE1 1 The reset sequence triggered by a Software functional reset event will start from PHASE3, skipping PHASE1 and PHASE2
29 SS_ST_DONE	Short Sequence for Self Test Completed 0 The reset sequence triggered by a Self Test Completed event will start from PHASE1
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 SS_EXR	Short Sequence for External Reset 0 The reset sequence triggered by an external reset event will start from PHASE1 1 The reset sequence triggered by an external reset event will start from PHASE3, skipping PHASE1 and PHASE2

56.3.10 Functional Reset Threshold (MC_RGM_FRET)

This register sets the threshold for 'functional' reset escalation to a 'destructive' reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Writing a non-zero value to the FRET field will enable the ‘functional’ reset escalation function. Writing any value to this register will reset the ‘functional’ reset counter. See Section “‘Functional’ Reset Escalation” for details on the ‘functional’ reset escalation function.

This register is reset only on power-on and any ‘destructive’ reset.

Address: 0h base + 604h offset = 604h

Bit	0	1	2	3	4	5	6	7
Read	0				FRET			
Write								
Reset	0	0	0	0	1	1	1	1

MC_RGM_FRET field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 FRET	‘Functional’ Reset Escalation Threshold - If the value of this field is 0, the ‘functional’ reset escalation function is disabled. Any other value is the number of ‘functional’ resets which will cause a ‘destructive’ reset if the RGM_FRET register is not written to beforehand.

56.3.11 ‘Destructive’ Reset Threshold (MC_RGM_DRET)

This register sets the threshold for ‘destructive’ reset escalation to keep the chip in the reset state until the next power-on reset triggers a new reset sequence. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Writing a non-zero value to the DRET field will enable the ‘destructive’ reset escalation function. Writing any value to this register will reset the ‘destructive’ reset counter. See Section “‘Destructive’ Reset Escalation” for details on the ‘destructive’ reset escalation function.

This register is reset only on power-on.

Address: 0h base + 608h offset = 608h

Bit	0	1	2	3	4	5	6	7
Read	0				DRET			
Write								
Reset	0	0	0	0	0	0	0	0

MC_RGM_DRET field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 DRET	'Destructive' Reset Escalation Threshold — If the value of this field is 0, the 'destructive' reset escalation function is disabled. Any other value is the number of 'destructive' resets which will keep the chip in the reset state until the next power-on reset triggers a new reset sequence if the RGM_DRET register is not written to beforehand.

56.4 Functional Description

56.4.1 Reset State Machine

The main role of MC_RGM is the generation of the reset sequence which ensures that the correct parts of the chip are reset based on the reset source event. This is summarized in the following table.

Table 56-14. MC_RGM Reset Implications

Source	What Gets Reset	External Reset Assertion
power-on reset	all	yes
'destructive' resets	all except some clock/reset management	programmable[1]
external reset	all except some clock/reset management and debug	programmable[2, 2]
'functional' resets	all except some clock/reset management and debug	programmable[3]
shortened 'functional' resets[3, 3]	flip-flops except some clock/reset management	programmable[3]

1. The assertion of the external reset is controlled via the **RGM_DBRE** register.
2. The assertion of the external reset is controlled via the **RGM_FBRE** register.
3. The short sequence is enabled via the **RGM_FESS** register.

Note

JTAG logic has its own independent reset control and is not controlled by the MC_RGM in any way.

The reset sequence is comprised of five phases managed by a state machine, which ensures that all phases are correctly processed through waiting for a minimum duration and until all processes that need to occur during that phase have been completed before proceeding to the next phase.

The state machine used to produce the reset sequence is shown in the following figure.

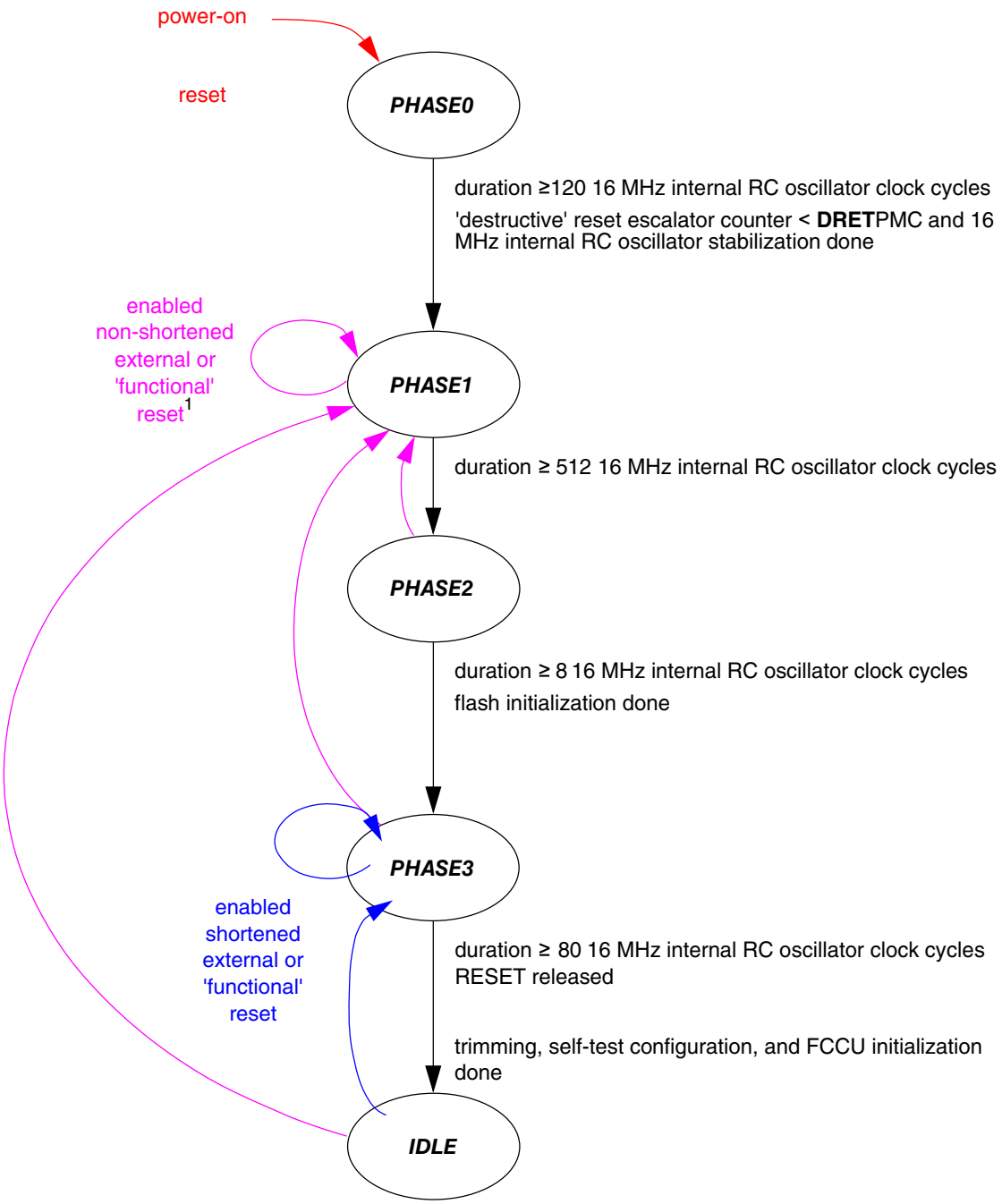


Figure 56-13. MC_RGM State Machine

56.4.1.1 *PHASE0* Phase

This phase is entered immediately from any phase on a power-on or enabled 'destructive' reset event. The reset state machine exits ***PHASE0*** and enters ***PHASE1*** on verification of the following:

- all enabled 'destructive' resets have been processed

- all processes that need to be done in ***PHASE0*** are completed
  - PMC and 16 MHz internal RC oscillator stabilization
- a minimum of 120 16 MHz internal RC oscillator clock cycles have elapsed since power-up completion and the last enabled 'destructive' reset event
- the 'destructive' reset escalator counter has not reached the value in the **DRET** field of the **RGM_DRET** resister

56.4.1.2 *PHASE1* Phase

This phase is entered either on exit from ***PHASE0*** or immediately from ***PHASE2***, ***PHASE3***, or ***IDLE*** on a non-masked external or 'functional' reset event if it has not been configured to trigger a 'short' sequence. The reset state machine exits ***PHASE1*** and enters ***PHASE2*** on verification of the following:

- all enabled, non-shortened 'functional' resets have been processed
- a minimum of 512 16 MHz internal RC oscillator clock cycles have elapsed since the last enabled external or non-shortened 'functional' reset event

56.4.1.3 *PHASE2* Phase

This phase is entered on exit from ***PHASE1***. The reset state machine exits ***PHASE2*** and enters ***PHASE3*** on verification of the following:

- all processes that need to be done in ***PHASE2*** are completed
  - flash initialization
- a minimum of 8 16 MHz internal RC oscillator clock cycles have elapsed since entering ***PHASE2***

56.4.1.4 *PHASE3* Phase

This phase is entered either on exit from ***PHASE2*** or immediately from ***IDLE*** on an enabled, shortened 'functional' reset event. The reset state machine exits ***PHASE3*** and enters ***IDLE*** on verification of the following:

- all processes that need to be done in ***PHASE3*** are completed

- trimming, self-test configuration, and FCCU initialization

- a minimum of 80 16 MHz internal RC oscillator clock cycles have elapsed since the last enabled, shortened 'functional' reset event

56.4.1.5 *IDLE* Phase

This is the final phase and is entered on exit from ***PHASE3***. When this phase is reached, the MC_RGM releases control of the system to the platform and waits for new reset events that can trigger a reset sequence.

56.4.2 'Destructive' Resets

A 'destructive' reset indicates that an event has occurred after which critical register or memory content can no longer be guaranteed.

The status flag associated with a given 'destructive' reset event (**RGM_DES.F_<destructive reset>** bit) is set when the 'destructive' reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software's responsibility to determine which reset source is the most critical for the application.

A given 'destructive' reset dest_rst[*n*] can be optionally disabled by writing bit **RGM_DERD.D_<destructive reset>**.

Note

The **RGM_DERD** register can be written only once between two 'destructive' reset events.

The chip's low-voltage detector threshold ensures that, when the reset corresponding to the core supply low-voltage detect is enabled, the supply is sufficient to have the 'destructive' event correctly propagated through the digital logic. Therefore, if a given 'destructive' reset is enabled, the MC_RGM ensures that the associated reset event will be correctly triggered to the full system. However, if the given 'destructive' reset is disabled and the voltage goes below the digital functional threshold, functionality can no longer be ensured, and the reset may or may not be asserted.

An enabled 'destructive' reset will trigger a reset sequence starting from the beginning of ***PHASE0***.

56.4.3 External Reset

The MC_RGM manages the external reset coming from RESET. The detection of a falling edge on RESET will start the reset sequence from the beginning of ***PHASE1***.

The status flag associated with the external reset falling edge event (**RGM_FES.F_EXR** bit) is set when the external reset is asserted and the power-on reset is not asserted.

An enabled external reset will normally trigger a reset sequence starting from the beginning of ***PHASE1***. Nevertheless, the **RGM_FESS** register enables the further configuring of the reset sequence triggered by the external reset. When **RGM_FESS.SS_EXR** is set, the external reset will trigger a reset sequence starting directly from the beginning of ***PHASE3***, skipping ***PHASE1*** and ***PHASE2***. This can be useful especially when an external reset should not reset the flash.

The MC_RGM may also assert the external reset if the reset sequence was triggered by one of the following:

- a power-on reset
- a 'destructive' reset event configured via the **RGM_DBRE** register to assert the external reset
- an external reset event
- a 'functional' reset event configured via the **RGM_FBRE** register to assert the external reset

In this case, RESET_B is asserted until all conditions for exiting ***PHASE3*** have been met, with the exception of the RESET assertion check.

56.4.4 'Functional' Resets

A 'functional' reset indicates that an event has occurred after which it can be guaranteed that critical register and memory content is still intact.

The status flag associated with a given 'functional' reset event (**RGM_FES.F_<functional reset>** bit) is set when the 'functional' reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software's responsibility to determine which reset source is the most critical for the application.

The 'functional' reset can be optionally disabled by software writing bit **RGM_FERD.D_<functional reset>**.

Note

The **RGM_FERD** register can be written only once between two power-on reset events.

An enabled 'functional' reset will normally trigger a reset sequence starting from the beginning of ***PHASE1***. Nevertheless, the **RGM_FESS** register enables the further configuring of the reset sequence triggered by a 'functional' reset. When **RGM_FESS.SS_<functional reset>** is set, the associated 'functional' reset will trigger a reset sequence starting directly from the beginning of ***PHASE3***, skipping ***PHASE1*** and ***PHASE2***. This can be useful especially in case a 'functional' reset should not reset the flash module.

See the MC_ME chapter for details on the ***DRUN*** modes.

56.4.5 Alternate Event Generation

The MC_RGM provides alternative events to be generated on reset source assertion. When a reset source is asserted, the MC_RGM normally enters the reset sequence. Alternatively, it is possible for some reset source events to be converted from a reset to either a ***SAFE*** mode request issued to the MC_ME or to an interrupt request issued to the core.

Alternate event selection for a given reset source is made via the **RGM_F/DERD** and **RGM_F/DEAR** registers as shown in the following table.

Table 56-15. MC_RGM Alternate Event Selection

RGM_F/DERD Bit Value	RGM_F/DEAR Bit Value	Generated Event
0	X	reset
1	0	***SAFE*** mode request
1	1	interrupt request

The alternate event is cleared by deasserting the source of the request (that is, at the reset source that caused the alternate request) and also clearing the appropriate **RGM_F/DES** status bit.

Note

SAFE mode requests are generated regardless of whether the system clock is running. Interrupt requests are generated with the system clock, and therefore, if the system clock is disabled at the time of the event, the interrupt request isn't asserted until the system clock is re-enabled.

Note

If a masked 'destructive' reset event which is configured to generate a ***SAFE*** mode/interrupt request occurs during ***PHASE0***, it is ignored, and the MC_RGM will not send any safe mode/interrupt request to the MC_ME.

Note

If a masked 'functional' reset event which is configured to generate a ***SAFE*** mode/interrupt request occurs during ***PHASE1***, it is ignored, and the MC_RGM will not send any safe mode/interrupt request to the MC_ME.

56.4.6 'Functional' Reset Escalation

'Functional' reset escalation can be used to generate a 'destructive' reset if a number of 'functional' or external resets has occurred between software writes to the **RGM_FRET** register. This function is enabled by writing a non-zero value to the **FRET** field of this register.

Once 'functional' reset escalation has been enabled, the MC_RGM increases a counter on each 'functional' or external reset which causes a reset sequence to be initiated (that is, entrance into ***PHASE1*** or ***PHASE3*** from the ***IDLE*** phase). This counter is cleared on a write of any value to the **RGM_FRET** register and on any power-on or 'destructive' reset. If the counter reaches the value in the **RGM_FRET** register's **FRET** field, the MC_RGM asserts.

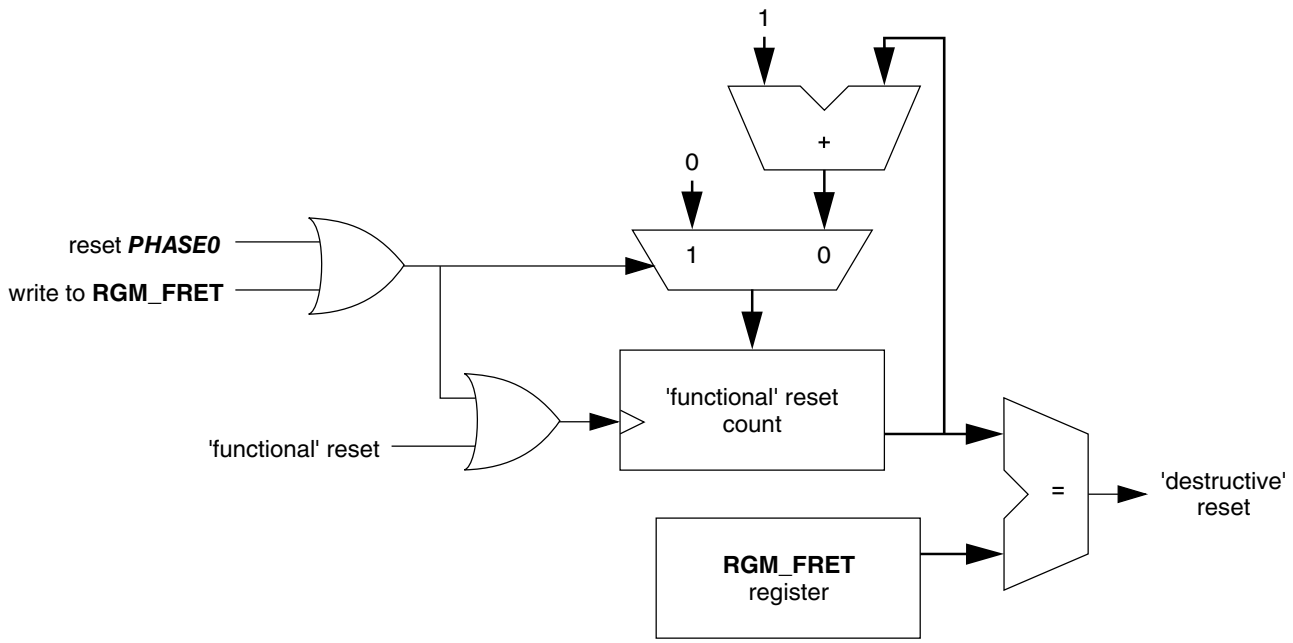


Figure 56-14. MC_RGM 'Functional' Reset Escalation to 'Destructive' Reset

56.4.7 'Destructive' Reset Escalation

'Destructive' reset escalation can be used to keep the chip in the reset state until the power-on triggers a reset sequence if a number of 'destructive' resets has occurred between software writes to the **RGM_DRET** register. This function is enabled by writing a non-zero value to the **DRET** field of this register.

Once 'destructive' reset escalation has been enabled, the MC_RGM increases a counter on each 'destructive' reset and which causes a reset sequence to be initiated (i.e., entrance into ***PHASE0*** from the ***IDLE*** phase) or an ongoing reset sequence to restart (i.e., entrance into ***PHASE0*** from ***PHASE1***, ***PHASE2***, or ***PHASE3***). This counter is cleared on a write of any value to the **RGM_DRET** register and on any power-on reset. If the counter reaches the value in the **RGM_DRET** register's **DRET** field, the MC_RGM enters reset ***PHASE0*** and stays there until the next power-on reset occurs.

On a power-on reset, the dest_rst_escalation output is deasserted. It is asserted by the MC_RGM when the counter reaches the value in the **RGM_DRET** register's **DRET** field. It is not deasserted until the next power-on reset.

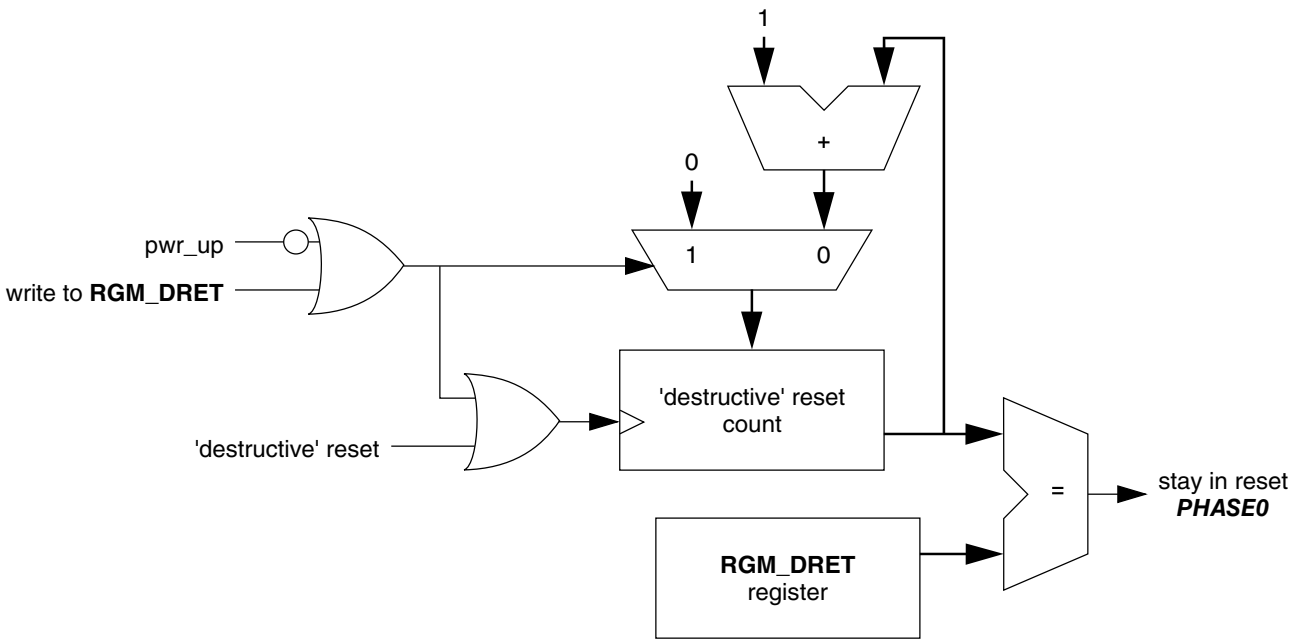


Figure 56-15. MC_RGM 'Destructive' Reset Escalation to Stay in *PHASE0*

Chapter 57
Boot Assist Flash (BAF)

57.1 BAF chip-specific module information

57.1.1 Search boot header and boot options

The BAF searches the flash memory for the boot header, checking the first word location of the 5 × 16 KB blocks, then the first of the 4 × 256 KB blocks. The blocks are searched in the order shown in Table 57-1. Once a boot header is found, no further blocks are searched.

Table 57-1. Locations of boot headers

Search order	Block	Address
1	16KB Code Flash block 0	00F9_C000h
2	16KB Code Flash block 1	00FA_0000h
3	16KB Code Flash block 2	00FA_4000h
4	16KB Code Flash block 3	00FA_8000h
5	16KB Code Flash block 4	00FA_C000h
6	256KB Code Flash block 0	0100_0000h
7	256KB Code Flash block 1	0104_0000h
8	256KB Code Flash block 2	0108_0000h
9	256KB Code Flash block 3	010C_0000h

The first header found that contains the value 005Ah in the first halfword is valid for booting. The whole header structure is shown in Table 57-2.

Table 57-2. Boot header structure

Address Offset	Contents
00h	Boot Header Configuration (see Table 57-3)
04h	Reserved for future use
08h	Reserved for future use

Table continues on the next page...

Table 57-2. Boot header structure (continued)

Address Offset	Contents
0Ch	Reserved for future use
10h	CPU0 Reset Vector
14h	CPU1 Reset Vector
18h	CPUC Reset Vector

Table 57-3. Boot header configuration

Offset 00h															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0				0				5				A			
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0				0				0				CPU1	CPUC	CPU0	0

Table 57-4. Boot header configuration field description

Field	Description
CPU1	CPU1 is enabled (will start execution at CPU1 Reset Vector)
CPUC	CPU_SC is enabled (will start execution at CPU_SC Reset Vector, must be the same as CPU0 Reset Vector)
CPU0	CPU0 is enabled (will start execution at CPU0 Reset Vector)

The cores that are enabled in the boot header configuration field are enabled by programming MC_ME_CCTL*n*[DRUN] register to run in all DRUN modes and their reset vector is set in register MC_ME_CADDR*n*. Then, the memory area of the BAF is protected from execution for functional safety considerations. This is achieved by disabling BAF execution in PFLASH control register3 (PFLASHC_PFCR3[BAF_DIS]). Finally, a mode change (from DRUN to DRUN again) is requested from the MC_ME module to activate the new core configuration and make the cores jump to their respective reset vectors. At this point the BAF has finished its operation.

57.1.2 LinFLEXD_1 and FlexCAN_1 pin configuration

Table 57-5 shows the device pin configurations for LINFlexD_1 and FlexCAN_1 modules.

Table 57-5. FlexCAN_1 and LINFlexD_1 pin configuration

Pads	Reset Function	Initial Serial Boot mode		Serial Boot mode after valid CAN message received	Serial Boot mode after valid LINFlexD message received
		Function	Pad configuration	Function	Function
PG[6]	GPIO Input buffer off Output buffer off	CAN1TX	Output Push/Pull medium no pullup/pulldown	CAN1TX	LIN1TX
PG[5]	GPIO Input buffer off Output buffer off	LIN1RX and CAN1RX	Input pullup hystersis TTL voltage level	CAN1RX	LIN1RX

57.1.3 BAF memory map

The BAF is located in a 16 KB block of flash that is mapped adjacent to the UTEST flash memory block. It is one time programmable (OTP) and is programmed during factory test. The address space and memory used by the BAF code are shown in Table 57-6.

Table 57-6. BAF memory map

BAF component	Address
BAF address range	0040_4000h – 0040_7FFFh
BAF version number	0040_4000h
BAF code entry point	0040_4100h
BAF code exit point	5180_0080h
BAF data area and stack	5180_0000h – 5180_01FFh

Table 57-7 shows UTEST memory addresses used by the BAF.

Table 57-7. UTEST addresses

UTEST component	Address
Soft DCF Record Start Address	0040_00C0h
DCF Start Record	0040_0300h

57.1.4 Reset values on BAF exit

Table 57-8. Reset Values on BAF exit

Sl. No.	Module Name	Register	BAF Exit Reset Value
1	MC_ME	ME_IS (Interrupt status)	0x00000001
2	MC_ME	ME_DMTS (Debug mode transition status)	0x30000000
3	FCCU	FCCU_NCF_CFG_0	0xFFF7FFFF
4	LIN_Flex	LINSR	0x00000020
5	LIN_Flex	LINESR	0x00008000

57.1.5 Serial boot password behavior

Due to the SWT implementation on this chip, the serial boot password behavior described in Download 64-bit password and password check needs to be read as follows for this chip:

Table 57-9. Chip-specific serial boot password behavior

Generic module behavior described in Download 64-bit password and password check	Behavior specific to this chip
"If an incorrect password is supplied, BAF puts the device into static mode (low power safe mode)."	If an incorrect password is supplied, BAF puts the device into static mode (low power safe mode), *and causes a functional reset after 30s*.
"If 64 bits of password data is not received after approximately 30s, BAF causes a destructive reset of the MCU."	If 64 bits of password data is not received after approximately 30s, BAF causes a *functional* reset of the MCU.

57.2 Introduction

The MCU is booted through a collaboration of several blocks, hardware, and firmware. The first boot phases are performed by a state machine inside the System Status and Configuration Module (SSCM). Once completed, SSCM send reset vector to boot core, which sets PC to BAF start address.

The BAF code then checks the life cycle of the device. If the life cycle is FAIL_ANALYSIS, the BAF enters a loop in which it services the watchdog and CSE initialization is enabled. The BAF does not execute further since the read access to flash memory boot sectors for the cores may be disabled if the device life cycle is FAIL_ANALYSIS. Otherwise, it searches for a boot header and boots the application code in internal flash memory.

If no boot header is found in internal flash memory, it downloads application code serially using the LINFlexD or FlexCAN modules (see Serial boot configuration for configuration details).

57.3 BAF image header

BAF image contains a 256-byte header starting from the BAF version number (see table "BAF memory map" table in the "BAF Configuration" section of this chapter). The header contains important information such as BAF version. The BAF header is explained in Table 57-10.

Table 57-10. BAF image header

Address offset	Size (bits)	Field description
0000h	32	BAF image version
0004h – 00FFh	Reserved	

57.3.1 BAF image version

The BAF version is a 32-bit field in the image header at the BAF starting address (see the "BAF memory map" table in the "BAF Configuration" section of this chapter). The 32-bit field is explained in Table 57-11.

Table 57-11. BAF image version

Field	Description
[0:7]	Major Number: This field contains the major version number for the BAF image
[8:15]	Minor Number: This field contains the minor version number for the BAF image
[16:23]	BAF Type: This field contains customer specific information.
[24:31]	Reserved

57.4 Functional description

A functional description of the BAF modes is contained in the following sections.

57.4.1 Boot modes

Depending on the state and the content of its flash memory the device can enter one of three different boot modes:

- Boot from internal flash – If the internal flash contains application code with a valid boot header, the application is booted. The boot header contains flags that enable/ disable individual cores and their reset vectors.
- Serial boot – If no valid boot header is found and the current life cycle phase of the device is CUST_DEL (Customer Delivery) or JDP/FSL Production, an attempt is made to download the application by a serial protocol over LINFlexD or FlexCAN physical interface. The downloaded code is then executed by the initial boot core.
- Static mode – The device enters power safe mode and waits for an external reset.

57.4.2 Device initialization

The BAF performs a minimal initialization of core registers and clocks required to enter the above described boot modes, then attempts to interact with the buddy device[1] in the following way:

1. Buddy device: If the device is an emulation device (buddy device present) and the calibration RAM content is valid, then this RAM is automatically remapped over the calibration flash memory. This is intended to assist with cold-start calibration.
2. BAF sets up the exception vector table to catch machine check exceptions, which may occur if there are any uncorrectable ECC errors. In the event of an uncorrectable ECC error, the BAF issues a destructive reset from the machine check exception handler if the exception is due to the reading of instructions from the BAF region. If an exception occurs during reading of the Boot header, the BAF ignores the ECC error and jumps to next boot header. If an ECC error occurs during reading of D-MEM, the BAF will still issue a destructive reset. BAF programs the Mode Entry module’s MC_ME_MCTL[TARGET_MODE] to issue a destructive reset.
3. BAF uses memory of the Core_1 D-MEM area as in the 'BAF memory map' table (at the beginning of this chapter) for its stack, data and any code that executes from volatile memory. During the BAF start-up procedure, it initializes this section of memory so that there is no ECC error when BAF accesses this memory area.

1. "Buddy device" refers to a companion device that is used only for system development and debug activities. It is not used in production systems.

57.4.3 Flow of control

Figure 57-1 shows the overall flow of control in the BAF. The individual steps are explained in the following sections.

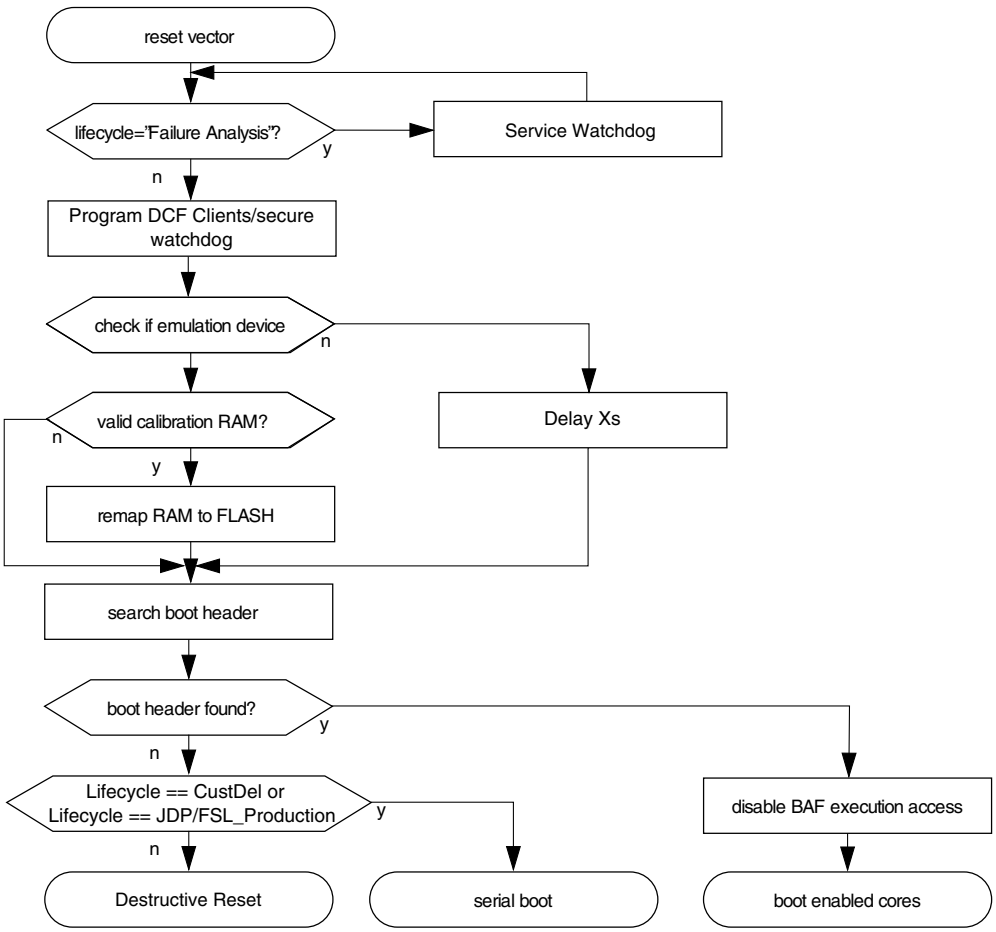


Figure 57-1. Flow of control

57.4.4 Optionally remap calibration RAM

If the device is an emulation device (buddy device present, SIUL2_MIDR1[ED] = 1), and the external overlay RAM has a valid content, then this RAM is automatically remapped over the selected flash memory regions. The details of the mapping are defined by a 400-byte configuration table at the beginning of the external overlay RAM. The layout of the table is depicted in Table 57-12.

The content of the overlay RAM is assumed valid if all of the following are true:

- The BD_INIT flag in the initialization status register of the buddy device (BD_SIUL2_ISR) is set.
- A CRC calculation over the configuration table yields the correct result.
- The version of the calibration data matches the version of the application software in flash memory (*((uint64_t*)ADDR_SWID) = DATID).

In this case, all 32 calibration remap descriptors (PFCRDs) are setup with the values contained in the configuration table. Finally, the PFCRDE field is copied to the enable register in the PFLASH controller.

Table 57-12. Remap configuration table

Address Offset	Name	Contents
00h	CRC_REF	CRC reference value
04h	ADDR_SWID	Address of 64 bit ID value in SW FLASH
08h	DATID	64 bit ID value of the calibration data
0Ch		
10h	PFCRDE	CRD enable flags
14h	LSTARTADDR0	PFCRD0 (logical start address)
18h	PSTARTADDR0	PFCRD0 (physical start address)
1Ch	CRDSize0	PFCRD0 (Master Mask, size)
20h	LSTARTADDR1	PFCRD1 (logical start address)
24h	PSTARTADDR1	PFCRD1 (physical start address)
28h	CRDSize1	PFCRD1 (Master Mask, size)
…	…	…
188h	LSTARTADDR31	PFCRD31 (logical start address)
18Ch	PSTARTADDR31	PFCRD31 (physical start address)
190h	CRDSize31	PFCRD31 (Master Mask, size)

The CRC calculation is performed using the CRC unit applying the CRC-32 (Ethernet protocol) polynomial. The calculation is initialized with a seed of 0000h. Then, the 100 32-bit words starting at table address offset 04h are moved to the CRC_INP register by the CPU. Finally, the content of the CRC_OUTP register is compared to the CRC_REF value.

57.4.5 BAF SAFE mode event handling

BAF executes under different conditions viz. Power-On-Reset (POR), Destructive Reset or any Functional Reset (Long or Short).

Due to device configuration, the device mode can be SAFE or DRUN during BAF execution. BAF issues a DRUN Mode transition to enable any peripheral or transfer control to application. If the device mode is SAFE or changes to SAFE during ongoing DRUN mode transition, then BAF issues an additional SAFE mode transition. After BAF execution when the control is transferred to the application, the device mode can be either SAFE or DRUN. It is recommended that the application (in its bootstrap) should take necessary action after idenitifying the current mode.

NOTE

Safe mode handling is only available in Flash Boot use case. Serial boot mode does not support SAFE mode handling as XOSC is disabled in SAFE mode.

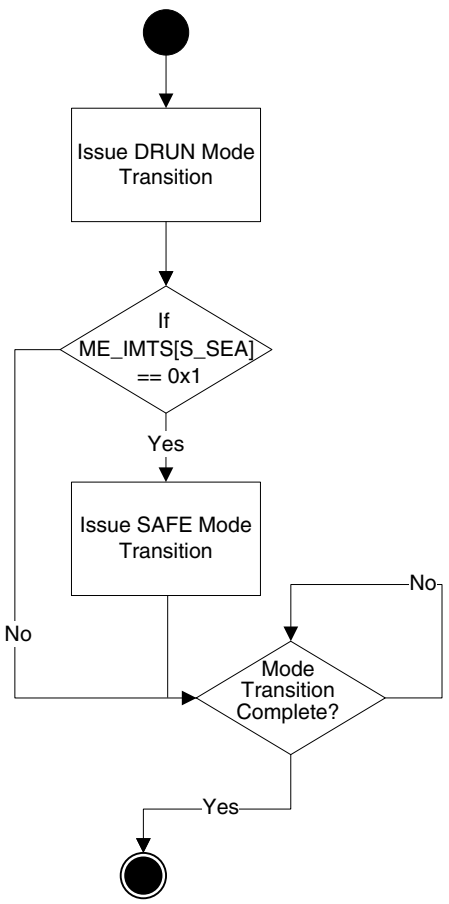


Figure 57-2. SAFE Mode Handling

57.4.6 BAF NMI Event Handling

BAF executes under different conditions viz. Power-On-Reset (POR), Destructive Reset or any Functional Reset (Long or Short). Due to device configuration, there can be a case where pending FCCU NMI events are latched in the device. These NMI events are

masked as described in NMI/WKPU interface. When BAF issues device mode transition to transfer control to application (MC_ME_CADDR*x*[RMC] = 1), the FCCU NMI event gets un-masked causing exception till RMC take effect and resets the core. The exception to CPU0 causes an additional fault (CPU0 Machine check) to be latched in the FCCU. Reset to the core also masks FCCU NMI Reaction.

As a workaround, the application can install a CALLBACK – BAF callback DCF client to initialize FCCU fault reaction to default state (No reaction for NMI Faults). This ensures that NMI faults occurring (during the BAF execution) are logged in the system. Application can then in its boot code can take corrective action and re-initialize FCCU.

As a best practice, application must check for any pending FCCU events and clear them.

57.4.7 Initialization of BAF DCF clients

57.4.7.1 Soft DCF clients

DCF clients are 32-bit hardware registers that are written by the SSCM during reset with data that is stored in DCF record format in UTEST region of flash memory. These DCF records are used to configure module registers, and select device operating modes.

The BAF extends the DCF record concept by implementing soft DCF clients. The 64-bit DCF records include:

- 32-bits of data
- 15-bit chip select field (selects the module which implements the DCF clients)
- 15-bit address field (selects a specific DCF client within the selected module)
- Parity bit (not used with UTEST DCF records)
- Stop bit (to indicate the end of DCF record has been reached)

As part of its execution flow, BAF searches through the list of DCF records in UTEST flash to determine if any of the records are to be processed by BAF.

The search starts at one of two addresses: either the start address of DCF records in UTEST (DCF Start Record), or the address specified in UTEST (Soft DCF Record Start Address). If either of the two least significant bits of the 32-bit address stored at the Soft DCF Record Start Address is set, the default start address at the DCF Start Record is used. If the two least significant bits are clear, this 32-bit value is assumed to be the start address of the search.

The default setting of the two least significant bits is 11b (erased value of flash memory is FFFF_FFFFh). A user may intentionally program a dummy address with the least two significant bits set in order to disable the search for an alternate address (as the UTEST block is OTP). Valid search start address must start on a 4-byte boundary.

The DCF record search finishes when a DCF record is read with a STOP bit set. The STOP bit may be set because that memory location is not programmed yet (so the list can be added to) or it can be an intentionally programmed STOP record to indicate that no more records are added to the list. BAF does not execute the DCF record with STOP bit set. BAF parses the STOP bit before parsing the rest of the records.

BAF uses Chip Select 14 (CS14) for BAF DCF clients. For records with CS14 asserted, the BAF uses the address field (address[14:0]) in DCF records to determine how to process the 32-bits of data. The soft DCF clients programmed by BAF are listed in Table 57-13.

The DCF clients which are used for initialization of security watchdog are explained in Security watchdog configuration.

A group of four Soft DCF clients is used to provide a mechanism to write configuration information to address locations. The first DCF client receive 32-bit address information. The next three DCF Clients receive the data that is to be written to the address contained in the first client. The three data clients allow 32-bit, 16-bit and 8-bit data to be written to the address in the first client. These four clients can be accessed repeatedly to write whole string of data to memory.

The serial boot DCF client provides a mechanism which can configure the BAF to perform serial boot or boot from internal flash. Details can be viewed in SER_BOOT_CBACK – BAF serial boot DCF client.

Table 57-13. BAF DCF client list

Address	Name	Label	Description
0000h	Security Watchdog Control register	SEC_SWT_CR	Data to write to SWT Control Register.
0001h	Security Watchdog Timeout register	SEC_SWT_TO	Security watchdog counter value.
0002h	Security Watchdog service address	SEC_IAC8	Instruction Address compare register value - Code present at this address services the watchdog.
0003h	Security watchdog CPU select	SEC_SWT_CPU	Determines which CPU is to use the security watchdog.
0004h – 01FFh	Reserved	—	Reserved for future use
0200h	Address	ADDR	The address to which DATA*n* is written.
0201h	32-bit data	DATA32	The 32-bit data that is written to ADDR.
0202h	16-bit data	DATA16	The 16-bit data that is written to ADDR.

Table continues on the next page...

Table 57-13. BAF DCF client list (continued)

Address	Name	Label	Description
0203h	8-bit data	DATA8	The 8-bit data that is written to ADDR.
0204h – 020Fh	Reserved	—	Reserved for future use.
0210h	Callback Address	CALLBACK	The address of a callback function.
0211h	Serial boot Callback address	SER_BOOT_CBACK	The address of serial boot callback function.
0212h – 0213h	Reserved	—	Reserved for future use.
0214h	Enable Calibration Delay Loop	CALIBRATION_PD_DELAY_ENABLE	Enables calibration remap equivalent delay loop on PD.
0215h	LIN_CLK Domain Select	BAF_DCF_CLIENT_LIN_CLK_DOMAIN	Selects LIN_CLK domain in A-synchronous mode.
0216h	IVPR1 for CPU1	IVPR1_CPU1	Programs the IVPR1 for CPU1.
0217h – 7FFF	Reserved	—	Reserved for future use.

NOTE

BAF boots with a watchdog timeout window set to 16ms. Parsing more and more DCF Records may increase boot time. But it should be taken care that number of DCF records should not be very high as it can cause watchdog window timeout.

57.4.7.2 Security watchdog configuration

BAF initializes the security watchdog before it jumps to the HOST CPU application. BAF implements four DCF clients for security watchdog initialization. The BAF parses the DCF records and looks for configuration data to be used to program the security watchdog registers. BAF first collects all information required to program the security watchdog from DCF records and then programs the security watchdog registers. BAF configures only SWT1 as security watchdog for the CPU1 (Core_1) core. Configuration of the Security watchdog for other cores has to be performed by the application running on that core as BAF cannot program other core's IAC8 register. Security watchdog detailed configuration steps are explained in the section below.

57.4.7.2.1 Security Watchdog Control Register (SEC_SWT_CR)

BAF parses the DCF records. If BAF finds a record with CS14 set and address field as 0000h while parsing, BAF programs the DCF client "Security watchdog control register" with the 32-bit value present in the DATA section of the DCF records. Once written, any subsequent writes to this DCF Client are ignored by BAF.

57.4.7.2.2 Security Watchdog Timeout (SEC_SWT_TO)

The BAF parses the DCF records. While parsing, if the BAF finds a DCF record with CS14 set and address field as 0001h, the BAF programs the security watchdog timeout DCF Clients. This is the security watchdog timeout register. The BAF checks the value obtained from DCF records. If the value yields a timeout greater than 30 seconds, then the BAF changes this value to 30 seconds. Once written, any subsequent writes to this DCF client are ignored.

57.4.7.2.3 Security Watchdog Service Address (SEC_IAC8)

BAF configures the security watchdog in 'fixed address execution service mode'. In this mode, the watchdog is serviced by executing code at the address loaded into the designated IAC8 (Instruction Address Compare) register. If BAF finds a DCF record with CS14 set and address field as 0002h while parsing the DCF records, BAF programs the DCF clients for SEC_IAC8. This DCF client can only be written once. Subsequent writes are ignored.

NOTE

To program the core IAC8 special purpose register, the ownership of this register should be with software. If Hardware has its ownership software will not be able to write to it using mtspr instruction.

57.4.7.2.4 Security Watchdog CPU select (SEC_SWT_CPU)

This DCF client indicates to the BAF which Core will use the security watchdog. The BAF parses the DCF record and if it finds a record with CS14 set and an address field of 0003h, the BAF programs this DCF client. This means that the BAF uses the two least significant bits out of the 32-bit data in the DCF record to find the CPU number which will use the security watchdog. Table 57-15 and Table 57-14 shows the data layout for this client in the DCF record. This client may only be written once during reset. Subsequent writes are ignored.

NOTE

Due to the limitation that only the IAC8 on the CPU running the BAF code can be accessed, the BAF can only configure the Security Watchdog to operate on CPU1. If another CPU is selected, the Security watchdog is NOT configured.

Writing this client causes the BAF to configure the security watchdog based on data written to the three previous DCF clients. If any of the three previous DCF clients were not written, data is configured erroneously.

If security watchdog is assigned to CPU0, relevant CPU IAC8 register is programmed along with other debug configuration registers to enable IAC8 as watch point.

During normal BAF operation, SWT1 is configured to perform a conventional task watchdog function (cause reset if not regularly serviced). To assign the security watchdog to the CPU1, first SWT3 is configured to perform the watchdog function task. The timeout value, configuration, and current time is read from SWT1 and copied to SWT3. Then SWT1 is unlocked with current sequence and enabled as security watchdog.

Table 57-14. Security Watchdog Core select bit configuration

DCF Client Address 0003h															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0				0				0				0			
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0				0				0				0	0	SC1	SC0

Table 57-15. Security Watchdog CPU select bit description

Field	Description
[31:2]	Reserved
SC[1:0]	These bits select the CPU which will use the security watchdog. 00 Reserved 01 CPU1 10 Reserved 11 Reserved

NOTE

BAF programs FCCU for enabling SWT2 and SWT3 as fault source and makes them hardware recoverable. This is done to ensure that fault latched in FCCU get cleared on reset and on next SWT timeout FCCU can again trigger reset. BAF does not clear the faults latched in FCCU as they can be of interest for application.

NOTE

Currently, only the boot Core (CPU1) can be configured by the BAF to use the security watchdog. Future versions of the BAF may support other Cores, or customer code can configure other Cores to use the Security Watchdog.

57.4.7.3 BAF DCF client - ADDRESS

BAF parses the DCF records. If BAF finds a DCF record with CS14 set and Address field as 0200h while parsing, BAF programs this DCF client. This client is written with a 32-bit address from the Data section of DCF record. This 32-bit address is used in conjunction with any one of the DATA32, DATA16 or DATA8 DCF Data clients. BAF does not immediately use the ADDRESS information, but it is stored and used later in conjunction with a DATAX DCF client (DATA32, DATA16 or DATA8). Then BAF writes the data to the address in memory specified by this ADDRESS DCF client.

The user can create a series of data writes to memory by programming a series of DCF records consisting of pairs of records. The first DCF record of the pair writes to ADDR to provide the address. The second DCF record of the pair writes to the DATA client (DATAX) to provide the data. Subsequent pairs of DCF records can be programmed to write to more memory locations. The number of pairs of DCF records that can be programmed is only limited by the amount UTEST space set aside for DCF records.

Table 57-16. BAF DCF ADDR client configuration

DCF Client Address 0200h															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ADDR[0:15]															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADDR[16:31]															

Table 57-17. BAF DCF ADDR client description

Field	Description
ADDR[0:31]	32-Bit Address DCF client. This Address value is used in conjunction with a DATAX value. The BAF writes DATAX value to ADDRESS value in the memory space of the MCU

57.4.7.4 BAF DCF client - DATAX

CAUTION

While using DATAX DCF records, it should be taken in account that BAF does not take care of ECC scheme and can generate ECC, if write is performed in memory locations which are protected by ECC and are not pre-initialized as required.

57.4.7.4.1 DATA32 – 32-bit data DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0201h while parsing, BAF programs this 32-bit DCF client. BAF takes the memory address provided by the ADDR client, masks the least significant 2-bits to '0' (Only 32-bit aligned writes are supported) and writes this 32-bit data to the address.

Table 57-18. DATA32 DCF client configuration

DCF Client Address 0201h															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DATA[0:15]															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DATA[16:31]															

Table 57-19. DATA32 DCF client description

Field	Description
DATA[0:31]	32-Bit DCF Data

57.4.7.4.2 DATA16 – 16-bit data DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0202h while parsing, BAF programs this 16-bit DCF client. BAF takes the memory address provided by the ADDR client, masks the least significant bits to '0' (Only 16-bit aligned writes are supported) and writes least significant 16-bit data to the address. Data in the 16 most significant bits are ignored.

Table 57-20. DATA16 DCF client configuration

DCF Client Address 0202h															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Reserved															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DATA[0:15]															

Table 57-21. DATA16 DCF client description

Field	Description
DATA[0:15]	32-Bit DCF Data

57.4.7.4.3 DATA8 – 8-bit data DCF client

The BAF parses the DCF records. During parsing, if the BAF finds a record with CS14 set and address field equal to 0203h, the BAF programs this 8-bit DCF client. The BAF takes the memory address provided by the ADDR client, and writes least significant 8-bit data to the address. Data in the 24 most significant bits are ignored.

Table 57-22. DATA8 DCF client configuration

DCF Client Address 0203h															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Reserved															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Reserved								DATA[0:7]							

Table 57-23. DATA8 DCF client description

Field	Description
DATA[0:7]	8-Bit DCF Data

57.4.7.5 CALLBACK – BAF callback DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field as 0210h while parsing, BAF programs this callback DCF client. This client is written with a 32-bit start address of a function which is invoked by BAF. The BAF code checks the start address is within the UTEST or within the BAF flash memory blocks, and then passes program flow to this address. If the address is found to be outside the UTEST and BAF flash memory blocks, no function call is executed. The callback is aborted and BAF moves on to the next DCF records. The function that is called is expected to comply with Power.org EABI and to terminate in an orderly manner. An incorrectly written function causes unpredictable operations in BAF. No parameters are passed to this function and no return value is expected.

Warning

Excessively long callback functions could cause the watchdog timer to time-out and cause a reset.

BAF executes the callback function as the DCF records are parsed. Therefore several callback functions maybe executed by writing several DCF records that write to the callback DCF clients. The callback functions are executed in the same order as the DCF records.

57.4.7.6 SER_BOOT_CBACK – BAF serial boot DCF client

The BAF parses DCF records. If it finds a record with CS14 set and an address field of 0211h during parsing, the BAF programs this serial boot DCF client. This DCF client is written with a 32-bit start address of a function which is invoked by the BAF. The BAF code checks to see if the start address is within the UTEST or BAF flash memory blocks, then program flow is passed to this address if it is within either of these blocks. If the address is outside the the UTEST and BAF flash memory blocks, no function call is executed. The DCF client execution is aborted, and the BAF moves to the next DCF record. The function that is called is expected to comply with Power.org EABI and to terminate in an orderly manner. An incorrectly written function causes unpredictable operations in the BAF.

Warning

No parameters are passed to the function invoked by the BAF.

Excessively long callback functions could cause the watchdog timer to time-out and cause a reset.

The serial boot callback function returns a character value. The return value informs BAF if serial or normal internal flash boot should be executed (see Table 57-24 for details).

Table 57-24. Serial boot callback function return value definition

Return values	Description
00h	Boot from internal flash using standard search for boot record (see "Search boot header and boot options" at the beginning of this chapter).
01h	Serial boot using both FlexCAN and LINFlexD (in UART mode). See Optionally perform serial boot for details.
02h – FFh	Reserved

57.4.7.7 CALIBRATION_PD_DELAY_ENABLE – BAF calibration PD delay enable DCF client

The BAF parses DCF records. If it finds a record with CS14 set and an address field of 0214h during parsing, the BAF programs this Calibration PD delay DCF client. This DCF client is written with a 32-bit key in DATA section of this DCF record. BAF supports Calibration Remap over external overlay ram on Emulation Device. On Production Device, this feature is not present but an equivalent delay loop can be activated. This is used if same BAF boot time is required on PD and ED. This delay can be activated using this DCF Client, if key present in DATA section of DCF record is

0x000055AA. For all other values, if DCF client itself is not present then delay is not activated. This DCF record can be programmed multiple times, but only the last instance of the DCF Record will be effective.

57.4.7.8 LINCLK_DOMAIN_SELECTION - BAF LINCLK Synch/Asynch Clock domain selection DCF Client

The BAF parses DCF records. If it finds a record with CS14 set and an address field of 0215h during parsing, BAF selects LINCLK (Linflex Baud Clock) domain according to user setup.

If DCF record is not present or if the DATA section of this DCF record is set to anything other than 0xA5A5, LINCLK must be Synchronous to Linflex System Clock (PBRIDGEx) and will be switched to System Clock Selector, Clock Divider 1 (MC_CGM_SC_DC1). Refer to the Clocking Chapter for more details.

If the DATA section of this DCF record is set to 0xA5A5, LINCLK must be Asynchronous to Linflex System Clock (PBRIDGEx) and will be switched to Auxiliary Clock Selector 0, Clock Divider 4 (MC_CGM_AC0_DC4). Refer to the Clocking Chapter for more details.

57.4.7.9 IVPR1_CPU1 - BAF set IVPR1 for CPU1 DCF Client

The BAF parses DCF records. If it finds a record with CS14 set and an address field of 0216h during parsing, the BAF programs the IVPR1 for CPU1. This DCF client is written with a 32-bit value in DATA section of this DCF record. BAF parses this data value and programs it to IVPR1 for CPU1 just before jumping to application. If this DCF client is not present, then IVPR1 will contain address of BAF vector table. This DCF record can be programmed multiple times, but only the last instance of the DCF Record will be effective.

57.4.8 Optionally perform serial boot

If no boot header is found in any of the locations mentioned above, the BAF determines the Life Cycle status of the device. If the Life Cycle is in CUST_DELIV (Customer Delivery) or JDP/FSL Production, it attempts a serial boot. Otherwise, the boot has failed and BAF issues a destructive reset.

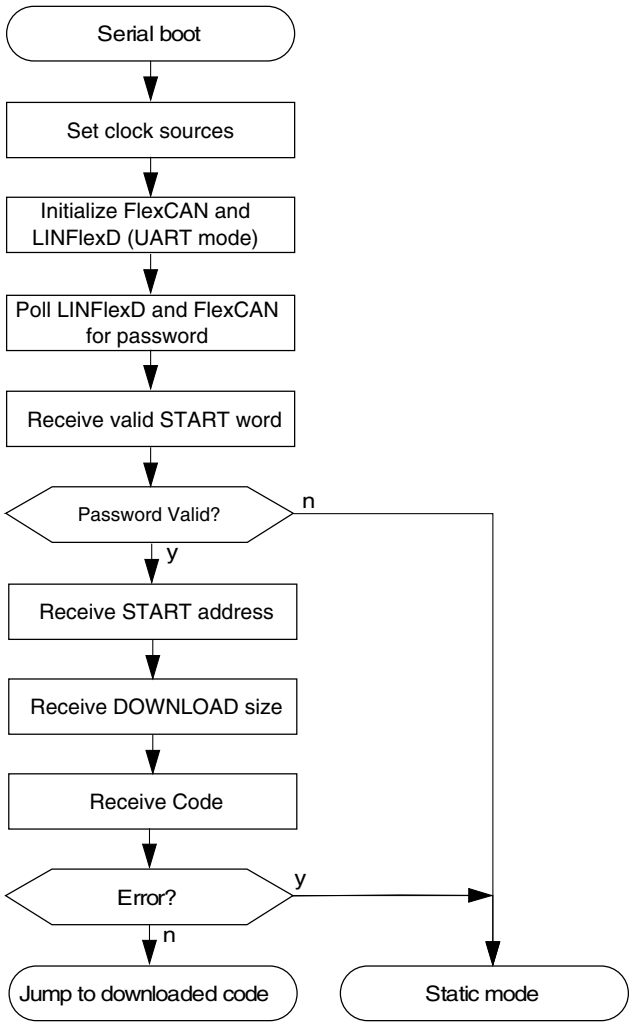


Figure 57-3. Flow of control during serial boot

The clock source in the MC_CGM module is configured and the PADs are set up for LINFlexD and FlexCAN signals (see Serial boot configuration for details), then the code is downloaded using LINFlexD or FlexCAN in Serial Boot mode. In the absence of errors causing the device to be placed in Static mode, the device state is restored to resemble the out-of-reset state and the downloaded code is executed.

Serial boot allows the download of application to internal SRAM via the LINFlexD or FlexCAN interface. After downloading the application image to SRAM, the BAF jumps to the start address provided as part of download protocol mentioned below and executes the downloaded application.

At the start of the process, both LINFlexD and FlexCAN are configured in passive listening mode, connected to the same pair of pads. As soon as the interface receives a valid message, the receiving interface is configured in full Tx and Rx mode. The interface that is not receiving is switched back to its reset state and does not participate further in the serial boot process.

If a CAN message of 8 bytes with ID 11h is received on the or FlexCAN controller first, the BAF program:

- Disables LINFlexD.
- Configures LINFlexD RXD input mux back to reset state.
- Transitions to CAN serial download protocol routine.

If a CAN message is received that does not contain 8 bytes or does not have an ID 11h by the FlexCAN controller, or the FlexCAN controller generates a CAN error, it is probable that the tool is sending a LIN frame. Under such circumstances the BAF program:

- Ignores the CAN message, if a message arrived.
- Clear any pending CAN error.
- Continue monitoring for valid CAN messages.

If any byte from LINFlexD is received first, the BAF program:

- Checks if the first byte is FEh of the password. If not, continue polling. The data may be a valid CAN message.
- If it is first byte of the password (FEh) then disable FlexCAN controller and restore to its reset state.
- Configures FlexCAN pads to their reset state.
- Transition to LINFlexD serial boot (UART mode).

57.4.8.1 Serial boot mode protocol

From a high-level perspective, the download protocol over CAN and LIN interfaces follows steps below:

1. Host transmits the 64-bit password to MCU.
2. Host transmits start address, size of downloaded code in bytes, NO_ECHO bit and VLE bit to MCU.
3. Host transmits a sequence of data bytes that are to be executed.
4. CPU1 executes code from start address provided in 2

NO_ECHO bit signifies if actual download data to be echoed or not. If NO_ECHO bit is set, then data will not be echoed otherwise if NO_ECHO bit is clear, then data will be echoed.

Each step must be complete before the next step starts.

The exact protocol between LinFlex and FlexCAN differs slightly.

57.4.8.2 LINFlexD download protocol

The communication is performed in half-duplex mode, Any transmission from the host is followed by the MCU transmission:

1. The host sends data to the MCU and starts waiting.
2. The MCU echoes the received 8-byte of password, 4-byte of load address and 4-byte of VLE-bit, NO_ECHO bit and 30-bit of download size to the host. The MCU echoes actual download data depending upon NO_ECHO bit.
3. The host verifies whether the echo is correct.
   - If the data is correct, the host can send the next byte of data.
   - If data is not correct, it is assumed the MCU has not correctly received the data and the only practical option is for the Host to reset the MCU and start again.

A more detailed description of these steps follows.

All multi-byte data structures are sent with MSB first.

The tool sends data in the following order:

1. 8 bytes that represent the password.
2. 4 bytes that represent the start address where the code is to be downloaded.
3. 4 bytes that represent the number of bytes to be downloaded, NO_ECHO bit and the VLE bit.
4. x bytes of code that is downloaded to SRAM and executed, (x must match the value specified in 3).

57.4.8.3 FlexCAN download protocol

The FlexCAN download protocol uses very similar structure to the LINFlexD protocol, but it makes use of the message ID's provided by CAN.

The tool sends data packed into standard CAN message in the following manner:

1. A message with 11h as the ID and 8-byte length to send the password. The MCU transmit the same message but with an ID of 1h.
2. A message with 12h as the ID and 8-byte length to send the start address, size of the code, NO_ECHO bit and VLE mode bit. The MCU transmit back the same data but with an ID of 2h.
3. A message with 13h as the ID is used to send data that is to be downloaded to SRAM. The MCU transmit the same data with an ID of 3h.

57.4.8.4 Download 64-bit password and password check

The first 64 bits received represent the password. The MCU only supports public passwords. The received password is always compared with the public password FEED_FACE_CAFE_BEEFh.

- If the correct password is received, BAF continues its task.
- If an incorrect password is supplied, BAF puts the device into static mode(low power safe mode).
- If 64 bits of password data is not received after approximately 30s, BAF causes a destructive reset of the MCU.

57.4.8.5 Download start address, VLE bit, NO_ECHO bit and code size

After the password, the next 8 bytes received by the MCU contains a 32-bit start address, the VLE mode bit, NO_ECHO bit and 30-bit code length as shown in Table 57-25.

The VLE bit (Variable Length Encoding) is used to indicate for which instruction set the downloaded code is compiled. The UART serial boot protocol supports download of VLE Code (VLE = 1) and BookE Code (VLE = 0).

The MCU only supports VLE code, i.e VLE=1.

The NO_ECHO bit is used to indicate if download data should be echoed by the MCU. If the NO_ECHO bit is set (for example, NO_ECHO = 1), then only 8 bytes of password, 4 bytes of download address, and 4 bytes of download size are echoed, but download data is not echoed. If the NO_ECHO bit is clear, then every byte sent to the MCU must be echoed.

The start address defines where the received data is stored and where the MCU branches after the download is finished. BAF ignores the three LSB bits of download address to avoid ECC on download address. But BAF does not align the address while jumping to application for executing. So, download address should be passed as 64-bit aligned for proper execution.

The Length defines how many data bytes have to be loaded.

Table 57-25. Start address, VLE bit, NO_ECHO bit and download size in bytes

Bytes 0-3															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Table continues on the next page...

Table 57-25. Start address, VLE bit, NO_ECHO bit and download size in bytes (continued)

START_ADDRESS[31:16]															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
START_ADDRESS[15:0]															
Bytes 4-7															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VLE	NO_ECHO	CODE_LENGTH[29:16]													
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CODE_LENGTH[15:0]															

57.4.8.6 Download data

Each byte of data received is downloaded into device SRAM starting from the address specified in the previous protocol step. It is not verified whether the provided address is a valid address in SRAM or is writable.

The address increments until the number of bytes of data received matches the code length specified in the previous protocol step.

Since the SRAM is protected by 64-bit wide Error Correction Code (ECC), BAF always writes to SRAM grouped in 64-bit double-words. If the last byte does not fall into a 64-bit boundary, the BAF fills it with zero's and renders the data aligned on a 64-bit boundary. After the last data is written to SRAM, the BAF writes a dummy double-word (0000_0000_0000_0000h) to the next address location. This is written to avoid possible ECC error during core prefetch.

57.4.8.7 Execute code

The BAF program waits for the last echo message transmission being completed. Then it restores the initial MCU configuration and jumps to the loaded code at Start Address which was received in step 2 of the protocol. At this point the BAF has finished its tasks and MCU is controlled by new code executing from SRAM.

57.4.9 Serial boot configuration

57.4.9.1 LINFlexD configuration

Boot using UART protocol is implemented by the LINFlexD module.

NOTE

See the pin configuration table at the beginning of this chapter for pins used by the LINFlexD module.

When Serial Boot mode is started the LINFlexD_Rx pin is configured as an input and muxed to the LINFlexD module. The LINFlexD_Tx pin remains in a passive input state until the first byte of protocol is received. It is then configured as an output and muxed to LINFlexD module. The application may not be using the Serial Boot Mode function, and may be using the pin used by LINFlexD_Tx as an input, which may have some external hardware driving the pin. Inadvertently driving this pin as an output would cause contention and possible damage.

The LINFlexD module operates in half-duplex mode. The interface is either transmitting a byte or receiving a byte, but never both at the same time. It is expected that LINFlexD pins are connected to a physical layer bus driver/receiver (PHY). Many PHYs enforce half-duplex mode by using a single data channel for both transmitting and receiving data (such as LIN or CAN). The side effect of this is that all data transmitted on the Tx pin is also received on the Rx pin. To prevent the LINFlexD module from receiving its own transmission, the Rx pin is blocked during transmission.

The LINFlexD controller is configured to operate at a baud rate of f_{XOSC} / 833, using an 8-bit data frame with no parity bit and 1 stop bit.

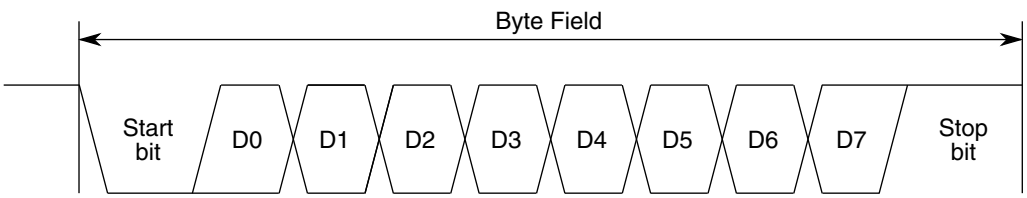


Figure 57-4. LINFlexD bit timing in UART mode

57.4.9.2 FlexCAN configuration

The FlexCAN controller is configured to operate at a baud rate equal to crystal frequency divided by 40, using the standard 11-bit identifier format detailed in CAN 2.0B specification (see Table 57-26 for baud rates).

NOTE

See the pin configuration table at the beginning of this chapter for pins used by the FlexCAN module.

The Bit timing is configured as shown in Figure 57-5.

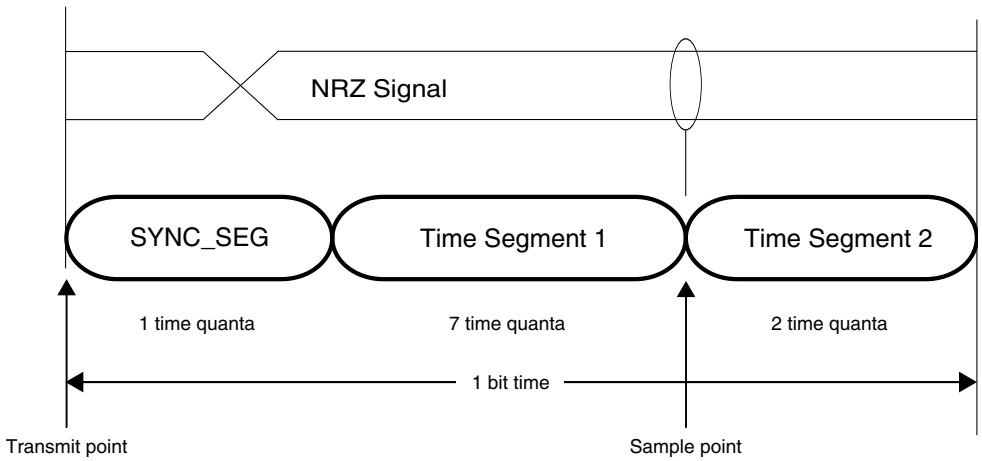


Figure 57-5. FlexCAN bit timing

57.4.9.3 LINFlexD and FlexCAN pin configuration

Please see the BAF configuration at the beginning of this chapter for pin details.

57.4.9.4 Baud rates

The LINFlexD baud rate is set by the frequency of an external crystal oscillator / 833.333. Table 57-26 shows some possible baud rates.

The FlexCAN baud rate is set by the frequency of the external crystal / 40.

Table 57-26. LINFLexD and FlexCAN baud rates

Crystal frequency	LINFlexD baud rate	FlexCAN baud rate
8 MHz	9600 baud	200 kbaud
12 MHz	14,400 baud	300 kbaud
16 MHz	19,200 baud	400 kbaud
20 MHz	24,000 baud	500 kbaud
40 MHz	48,000 baud	1000 kbaud

57.4.9.5 Protocol summary

Table 57-27 summarizes the LINFlexD protocol and BAF action during this boot mode.

Table 57-27. LINFlexD serial boot mode download protocol

Protocol steps	Host message sent	BAF response message	Action
1	64-bit password (MSB first)	64-bit password (MSB first)	Password checked for validity and compared with stored password.
2	32-bit store address	32-bit store address	Load address is stored for future use.
3	VLE bit + NO_ECHO bit + 30 bits of download data size in bytes (MSB first)	VLE bit + NO_ECHO bit + 30 bits of download data size in bytes (MSB first)	Size of download is stored for future use. Verify VLE bit.
4	8 bits of raw binary data	If NO_ECHO bit is set, data is echoed, otherwise data is not echoed.	8 x 8 bits of data are packed into 64-bit double-words. These double-words are stored in SRAM starting from the Load address. The Load address increments until the number of data received and stored matches the size as specified in the previous steps.
5	none	none	Branch to downloaded code

Table 57-28 shows the FlexCAN serial boot protocol.

Table 57-28. FlexCAN serial boot mode download protocol

Protocol step	Host message sent	BAF response message	Action
1	CAN ID 011h + 64-bit password	CAN ID 001h + 64-bit password	Password checked against fixed password (FEED_FACE_CAFE_BEEFh). Watchdog timer is refreshed if the password check is successful.
2	CAN ID 012h + 32-bit store address + 32-bit (1 VLE bit, 1 NO_ECHO bit and 30 bits of download data size in bytes)	CAN ID 002h + 32-bit store address + 32-bit (1 VLE bit, 1 NO_ECHO bit and 30 bits of download data size in bytes)	Load address and size of download are stored for future use.
3	CAN ID 013h + 8 to 64 bits of raw binary data	CAN ID 003h + 8 to 64 bits of raw binary data	Each byte of data received is store in MCU memory, starting at the address specified in the previous step and incrementing until the amount of data received and stored, matched the size as specified in the previous step.
4	None	None	The BAF returns IO pins and CAN module to their reset state, then branches to the first address the data was stored to (As specified in step 2)
5	none	none	Branch to downloaded code

57.5 Resources

- Clocks

    - Several clock configurations are made depending on the boot mode of the BAF. All settings are restored before execution of the application code (in flash memory or received via serial download).

- LINFlexD and FlexCAN
    - The LINFlexD and FlexCAN modules are used if serial boot mode is entered. This can be through failing to find a valid boot header, or from the SER_BOOT_CBACK callback function. All settings are restored before the application code in system RAM is executed.
- CRC
    - The CRC module is used to validate external overlay contents, if and only if the device is an emulation device and the buddy device has been initialized.
- PFLASH
    - PFLash module is used to disable BAF execution.
- Interrupts
    - No interrupts/exceptions are used or enabled, except the Machine Check exception, to handle ECC errors.
- Security watchdog
    - When SWT1 is chosen as security watchdog, SWT3 is used for task monitoring purpose.
- FCCU
    - FCCU is used to configure Fault Reaction of SWT1 and SWT3.

Chapter 58
System Status and Configuration Module (SSCM)

58.1 System Status and Configuration Module (SSCM) chip-specific module information

This section summarizes the System Status and Configuration Module (SSCM) configuration. There is one instance of the SSCM module on the MCU.

The SSCM reads the DCF records during boot, and parses this data to see if valid boot code exists. If the flash memory data from the DCF record is not valid, the boot sequence will automatically switch to the Boot Assist Flash (BAF) and search for serial boot modes.

The device specific reset value for the SSCM_MEMCONFIG register are shown in the table Table 58-1.

Table 58-1. SSCM_MEMCONFIG reset value

Field	Field Definition	Value
JPIN	JTAG Part ID	0346h
MREV	Minor Revision	0001b

Table 58-2 shows the address locations that the SSCM uses to search for startup information.

Table 58-2. SSCM startup information addresses

Description	Address
The location at which the SSCM will look for the UTEST DCF records (UTEST_OFFSET).	0040_0300h
Location of the Boot Assist code in flash memory (BAF)	0040_4000h

Table continues on the next page...

Table 58-2. SSCM startup information addresses (continued)

Description	Address
Locations that are checked for valid Reset Configuration Halfword (RCHW)	0100_0000h 0104_0000h 0108_0000h 010C_0000h 00F9_C000h 00FA_0000h 00FA_4000h 00FA_8000h

The SSCM module has a built-in register protection mechanism that affects the behavior of write operations to one or more registers. After software has activated the protection for a register, writes can only be performed in supervisor mode until the protection is either deactivated by software or the device undergoes reset.

See the Register Protection (REG_PROT) chapter for a detailed description of the register protection mechanism, and SSCM protected registers for the affected SSCM registers.

For the STCU transfer errors that can be indicated by SSCM_STATUS[CER], see the STCU2 chapter section IPS bus interface.

NOTE

The SSCM_STATUS[NXEN1] and [NXEN] fields can only be cleared by asserting JCOMP low (disabling Core Nexus) through the debug tool once Nexus3 access is enabled.

58.2 Introduction

58.2.1 Overview

The System Status and Configuration Module (SSCM) is pictured in the figure below.

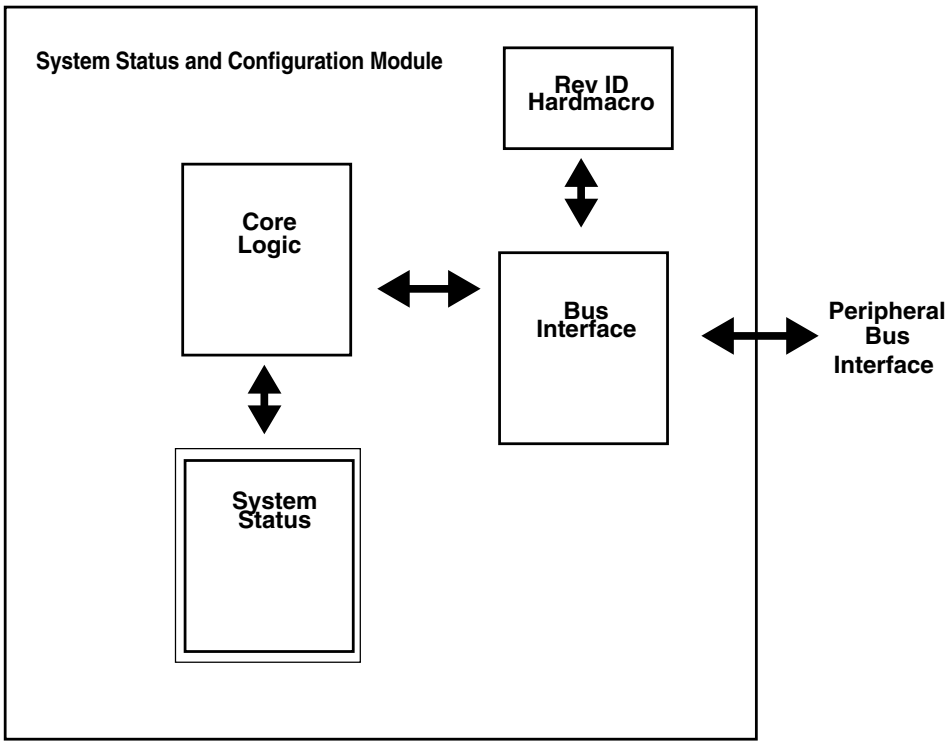


Figure 58-1. SSCM block diagram

58.2.2 Glossary

The table below shows a glossary of terms used through out the text.

Table 58-3. Glossary

Term	Definition
0011b	Binary numbers
0Fh	Hexadecimal numbers
Pin	External physical connection.
Signal	Electronic construct whose state or change in state conveys information.
X	In certain contexts, such as a signal encoding, this indicates a don't care. For example, if a field is binary coded X001b, the state of the first bit is a don't care.
y	y is used for a place holder of 1 hex digit for unknown values.

58.2.3 Features

The SSCM includes these distinct features:

- System Configuration and Status

- Device Mode and System Status
- Determine primary boot vector
- Decodes the Device Life Cycle

- Bus abort enable/disable
- Peripheral abort enable/disable

58.2.4 Modes of operation

The SSCM operates identically in all system modes.

58.3 External signal description

The SSCM has no external pins.

58.4 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the SSCM.

The table below shows the memory map for the SSCM. Note that all addresses are offsets; the absolute address may be calculated by adding the specified offset to the base address of the SSCM.

NOTE

All registers are accessible via 8-bit, 16-bit or 32-bit reads and/or writes. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. As an example, the STATUS register is accessible by a 16-bit read or write, SSCM Base + 02h, but performing a 16-bit access to SSCM Base + 0003h is illegal (that is, you should not make any transfer that is not aligned to burst size boundary).

NOTE

SSCM does not generate transfer errors in all memory holes.

SSCM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	SSCM System Status (SSCM_STATUS)	16	R	See section	58.4.1/2549
2	SSCM System Memory and ID Register (SSCM_MEMCONFIG)	16	R	See section	58.4.2/2551
28	Processor Start Address Register (SSCM_PSA)	32	R	See section	58.4.3/2551
34	Life Cycle Status Register (SSCM_LCSTAT)	32	R	See section	58.4.4/2552

58.4.1 SSCM System Status (SSCM_STATUS)

The System Status register reflects the current state of the system.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7
Read	0	CER	CERS	NXEN1	NXEN	1	Reserved	0
Write		w1c	w1c					
Reset	0	0	0	*	*	1	*	0

Bit	8	9	10	11	12	13	14	15
Read	BMODE			VLE	0	0	0	
Write								
Reset	*	*	*	*	0	0	0	0

* Notes:
- VLE field: Reset value depends on the associated option bits in flash memory.
- BMODE field: Reset value depends on the device status after leaving reset.
- Reserved field: This field can reset to 0 or 1.
- NXEN field: Reset value depends on the device status after leaving reset.
- NXEN1 field: Reset value depends on the device status after leaving reset..

SSCM_STATUS field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 CER	Configuration Error This field indicates that the SSCM has detected a configuration error during reset while loading initial device configuration. See the chip-specific SSCM information for details about sources of the error. If a non-permanent error is detected, this bit can be cleared by writing a 1. Refer to CERS-DCF Mechanism section. 0 No configuration problem detected by the SSCM 1 Device configuration is not correct

Table continues on the next page...

SSCM_STATUS field descriptions (continued)

Field	Description
2 CERS	Configuration Error for Safe DCF Clients This field indicates that the SSCM has detected a configuration error during reset while loading the Safe DCF Clients (DCF client with SPRD ADDR enabled, Triple Voting Enabled). If a non-permanent error is detected, this bit can be cleared by writing a 1. Refer to CERS-DCF Mechanism section. 0 No configuration problem detected by the SSCM 1 Device configuration is not correct
3 NXEN1	Processor 1 Nexus enable status 0 Processor 1 Nexus disabled. 1 Processor 1 Nexus enabled
4 NXEN	Processor 0 Nexus enable status 0 Processor 0 Nexus disabled. 1 Processor 0 Nexus enabled
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
6 Reserved	This field is reserved. This field is for internal use. Users should ignore read values and should not attempt to write the field.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–10 BMODE	Device Boot Mode This field is only updated during reset. 101 Boot from BootFlash
11 VLE	Variable Length Instruction Mode When booting in SC mode, this field indicates that the code stored in the flash memory is using the VLE instruction set. The value of this field is determined by the RCHW field of the flash memory boot sector. **NOTE:** Reset value depends on the associated option bits in flash memory. 0 SC Boot Location contains standard PPC code 1 SC Boot Location contains VLE code
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

58.4.2 SSCM System Memory and ID Register (SSCM_MEMCONFIG)

The System Memory and ID register is a read-only register which reflects the memory configuration of the system. It also contains the JTAG ID.

Address: 0h base + 2h offset = 2h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	JPIN										1	MREV				0
Write																
Reset	*	*	*	*	*	*	*	*	*	*	1	*	*	*	*	0

* Notes:
- MREV field: Reset value can be found in the SSCM Configuration section.
- JPIN field: Reset value reflects the JTAG ID of the device.

SSCM_MEMCONFIG field descriptions

Field	Description
0–9 JPIN	JTAG Part ID Number **NOTE:** Reset value reflects the JTAG ID of the device.
10 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
11–14 MREV	Minor Mask Revision
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

58.4.3 Processor Start Address Register (SSCM_PSA)

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SADR																															
W																																
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:
- SADR field: Reset value is the start address

SSCM_PSA field descriptions

Field	Description
0–31 SADR	Processor Start Address The boot processor will start executing application code from this address.In SC mode this indicates the location determined by the RCHW search. In BAF mode this will show the start address of the BAF code.

58.4.4 Life Cycle Status Register (SSCM_LCSTAT)

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						Reserved		0					LC		
W																
Reset	0	0	0	0	0	0	*	*	0	0	0	0	0	*	*	*

* Notes:
- LC field: Reset value is the Life Cycle of the device, and is unaffected by reset.
- Reserved field: Reset Value is '11' for a valid secured part.

SSCM_LCSTAT field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 Reserved	This field is reserved.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 LC	Life Cycle Determined by the SSCM by reading the LC slots. The encoding is: 000 Failure Analysis 001 Reserved 010 OEM Production 011 Customer Delivery 100 Reserved 101 Reserved 110 Freescale Production 111 In Field

58.5 Functional description

The primary purpose of the SSCM is to provide information about the current state and configuration of the system that may be useful for configuring application software and for debug of the system.

58.5.1 CERS-DCF Mechanism

The figure below shows that a faulty DCF record has been loaded between the original DCF record set and the final DCF record set. When a faulty DCF record is encountered, the SSCM_STATUS[CERS] field indicates that the device configuration is not correct. In a situation like the one shown below, CERS can be cleared by writing 1 to it, as long as the final DCF record set is correct.

If there is an error in the final set of loaded DCF records, CERS cannot be cleared. This situation indicates that the final DCF record set has at least one error. To fix this, the user should update flash memory after the faulty DCF record with a correct DCF record of the same type. For example, if the faulty DCF record is DCF2, then the correct DCF record should also be DCF2.

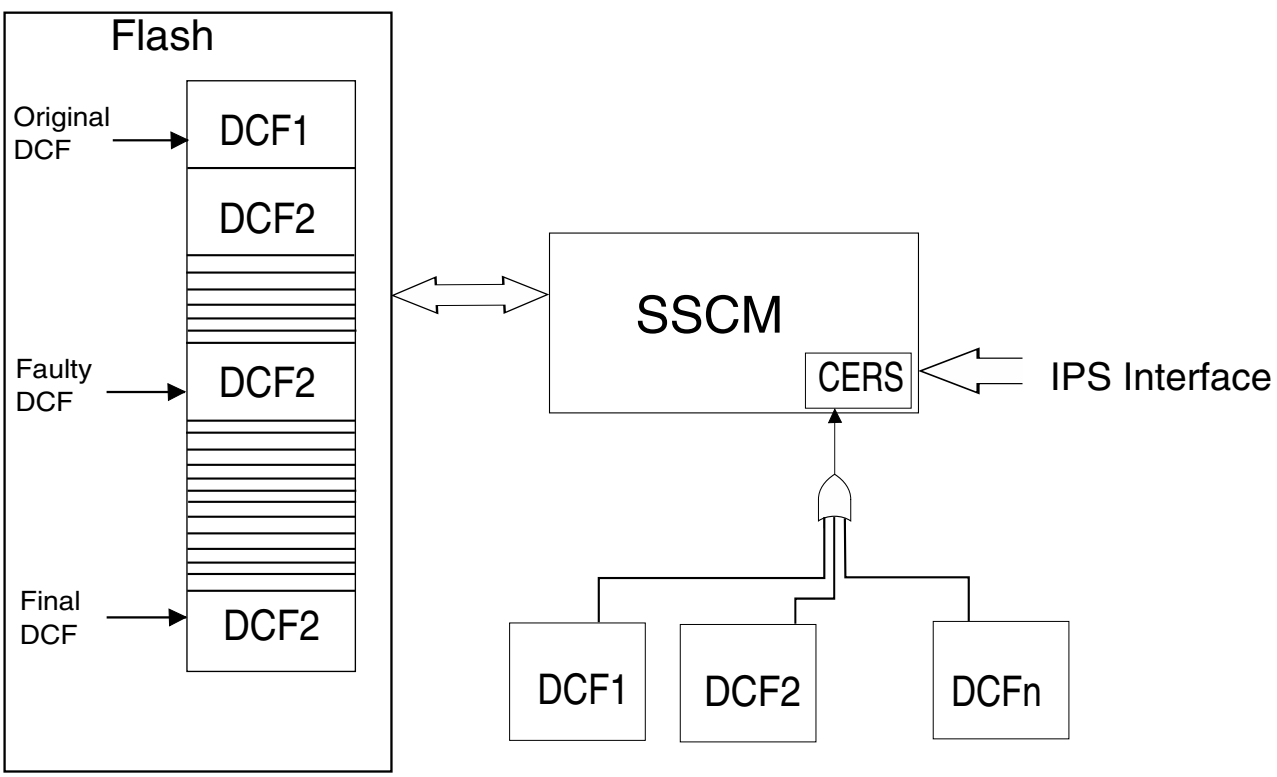


Figure 58-6. DCF Error and CERS Handling

58.5.2 ECC error monitoring

During flash memory scanning (such as Life Cycle or DCF records), the SSCM will monitor the flash memory status signals for ECC and other unexpected access errors. If such an error occurs, the SSCM will:

- Request a destructive reset
- Signal the FCCU or MEMU.

NOTE

For differential region, if any valid DCF records are programmed then there will be checked for ECC and stop record for DCF would not be checked for ECC.

58.5.3 Boot mode functionality

The device supports the following boot modes for the main boot core:

- Boot Assisted by Flash (BAF) - The device will boot from UTest NVM Flash Block, and perform device setup.

If booting is not possible with the selected configuration (for example, no Boot ID is found in the selected boot location) then the device will enter static mode.

58.5.4 BAF configuration

By default the device will boot from BAF.

In BAF mode (SSCM_STATUS[BMODE]=101b), the boot core will start executing BAF code at the location shown in the configuration section of the “Boot Assist Flash (BAF)” chapter.

58.5.5 Life Cycle

The SSCM will determine the Life Cycle (LC) of the device by reading the LC slots from the UTEST flash memory. The read operation is done during the reset phase with normal timings and it is protected by operating monitors and ECC check. In addition, a set of sanity checks executed over the LC read data guarantee the integrity of the final LC value.

At the end of the reset phase, the LC can have one of the following values:

- Freescale Production
- Customer Delivery
- OEM Production
- In Field
- Failure Analysis

Life cycle can only be moved in the direction shown below:



Figure 58-7. Life cycle sequence

The LC is written into 5 slots, 128 bits each, at fixed locations in the UTEST flash memory block. Each LC slot is read in one single atomic operation and it is organized into two fields:

- the valid field
- the invalid field

Depending on the possible combination of the data programmed into these fields, each LC slot can have one of the possible 4 statuses as seen in Table 58-9.

Table 58-9. LC slot status

LC slots		LC slot value
Valid field (64 bits)	Invalid field (64 bits)	
Erased	Erased	Erased
Marked	Erased	Active
Marked	Marked	Inactive
Any other values		Illegal

"Marked" in this case means that the value has been programmed with the bit pattern 55AA_50AF_55AA_50AFh. "Erased" is detected by the bit pattern FFFF_FFFF_FFFF_FFFFh. Table 58-10 shows how the slots are arranged in memory.

Table 58-10. LC slots in memory

Offset	LC slot word
00h	Valid[31:0]
04h	Valid[63:32]
08h	Invalid[31:0]
0Ch	Invalid[63:32]

The Life Cycle is determined as shown in Table 58-11. The priority of the entries is from top to bottom, so the first row that applies determines the resulting Life Cycle.

NOTE

"Erased" means the slot returns all 1s.

Table 58-11. LC slots

LC slot 0 Freescale Production	LC slot 1 Customer Delivery	LC slot 2 OEM Production	LC slot 3 In Field	LC slot 4 Failure Analysis	Resulting life cycle
Active	Erased	Erased	Erased	Erased	Freescale Production
Inactive	**Active**	Erased	Erased	Erased	Customer Delivery
Inactive	Inactive	**Active**	Erased	Erased	OEM Production
Inactive	Inactive	Inactive	**Active**	Erased	In Field
Inactive	Inactive	Inactive	Inactive	**Active**	Failure Analysis
Erased	Erased	Erased	Erased	Erased	System Reset
Invalid[1]	Invalid	Invalid	Invalid	Erased	In Field
Invalid	Invalid	Invalid	Invalid	Not Erased	System Reset

1. Any value programmed other than shown in table above for Erased, Active, and Inactive would be treated as illegal.

NOTE

When programming Life Cycle, ensure that programming is not interrupted and can complete without error.

58.6 Initialization and application information

58.6.1 Reset

The reset state of each individual bit appears in the detailed register descriptions.

58.7 Additional safety measures

58.7.1 Spurious reset protection

The SSCM implements protection against spurious module resets (for example, resets which only effect the SSCM, but not other modules) by gating the state machines status with the expected status of the MC_RGM. If a spurious reset occurs, the SSCM will not interfere with the flash memory bus or overwrite configuration registers. However, internal SSCM status will be lost.

Chapter 59
Power Management

59.1 Overview

The MPC5746R microcontroller includes a robust power management infrastructure that enables applications to select among various operational modes and to monitor internal voltages for high- and low-voltage conditions. The monitoring capability is also used to ensure supply voltages and internal voltages are within the required ranges before the microcontroller can exit RESET. This chapter gives an overview of the built-in power management features.

The power management infrastructure comprises the following modules:

- Power Control Unit (MC_PCU)
- Power Management Controller (PMC)

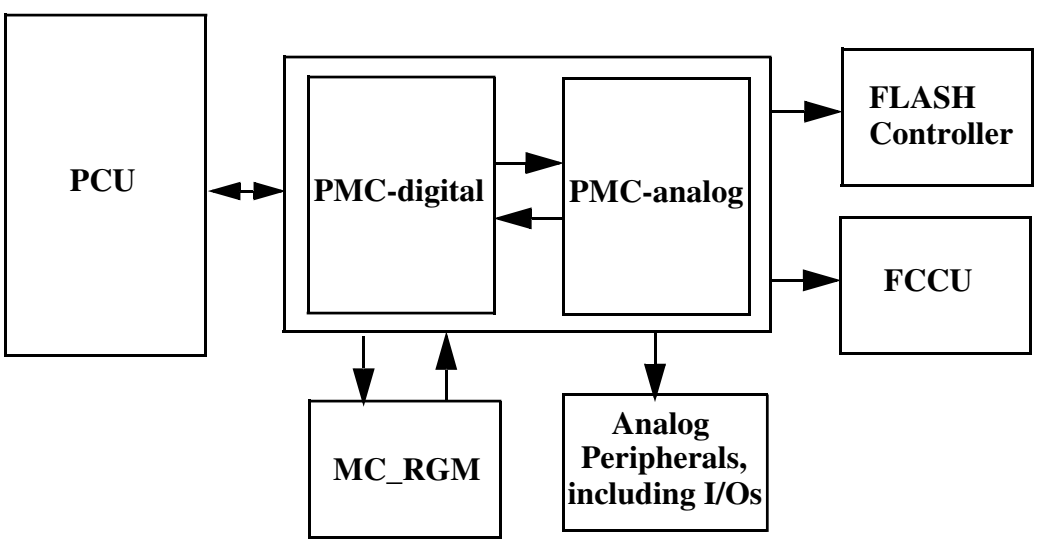


Figure 59-1. Power Management Overview

This chapter describes the module interaction in brief. For complete details, see the respective module chapters.

59.1.1 Power Management Controller (PMC)

The Power Management Controller implements a linear voltage regulator to generate the internal 1.25V digital supply from an external 3.5V to 5.5V supply and monitors various supply voltages of the MPC5746R chip for functional safety purposes. An additional regulator for the flash block is also within the PMU to provide the necessary 3.3V source to the FLASH module. Another internal regulator is present and generates a clean 1.26V supply for the LFAST PLL, as shown in the diagram in the next section. Depending on the supply voltages, power-on-reset (POR) and destructive/functional resets may be generated.

The PMC connects to the Reset Generation Module (MC_RGM), the Fault Collection Control Unit (FCCU) and the flash memory controller. The on-chip ADC receives signals from the PMC for diagnostic purposes.

59.1.2 Supply Concept

MPC5746R provides an on-chip regulator to derive the core supply from the external supply. A power transistor must be added on the PCB. See the MPC5746R Data Sheet for specifics on this device. Alternatively, an external linear or switching regulator can be used. The analog supply needs to be protected from noise on the digital supply at least by means of a decoupling capacitor. Better filters can be used if they do not affect regulator stability. A separate regulator for the analog supply is preferred.

Each of the PMC monitors supply voltage and release the resets when they are within the limits of operating conditions. The power-on-reset can be generated on-chip or can be provided from an external source, such as a brownout device.

Some supplies may already be connected internally inside the package depending on the package type, mainly in order to save pins in low pin count packages.

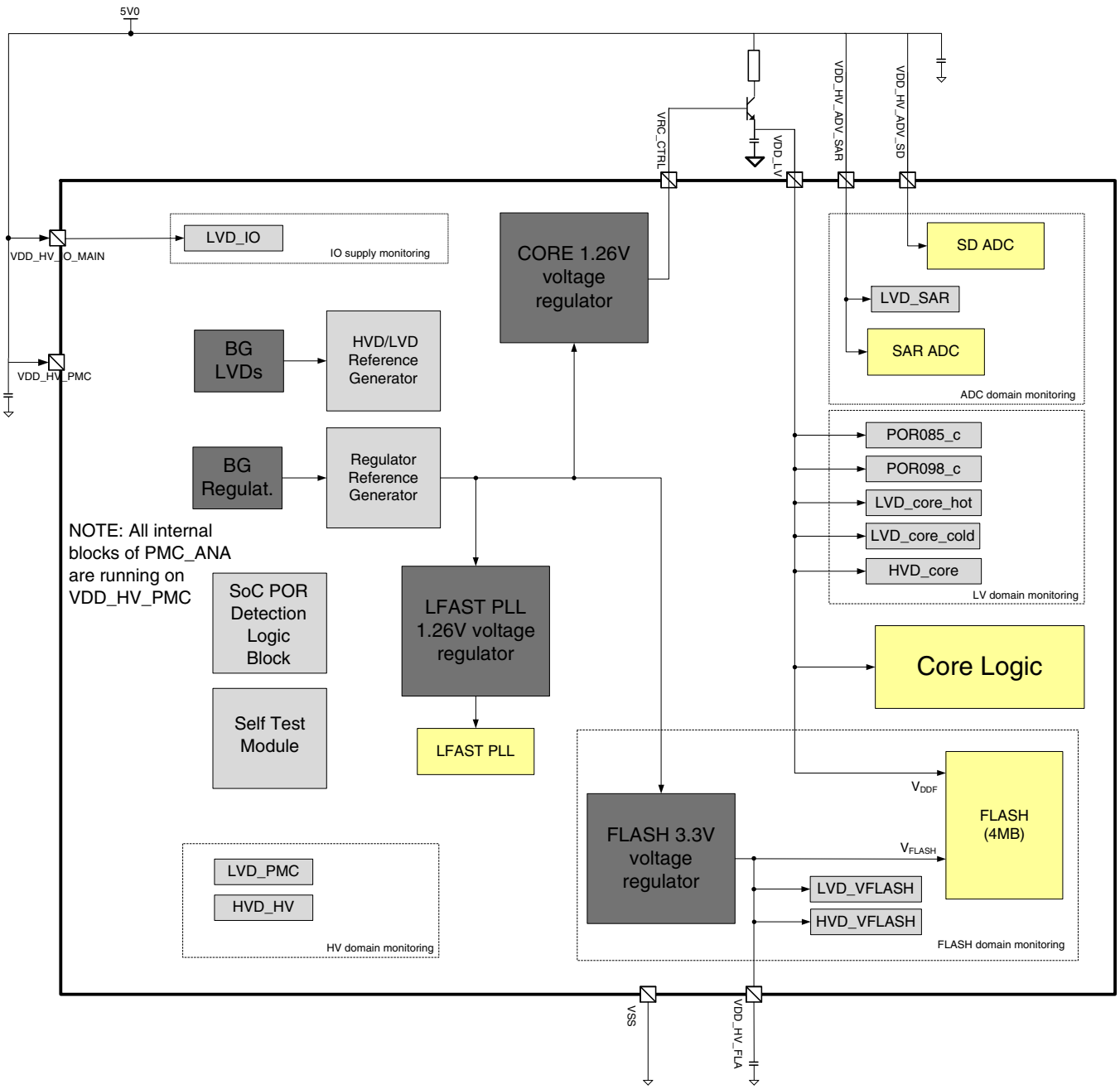


Figure 59-2. Supply concept

59.1.3 Power Modes

MPC5746R does not support ultra low-power modes that require power gating or separate power domains. The core logic implements STOP and low frequency power saving modes. A sudden transition to these low power modes from RUN mode (or vice versa) may result in de-stabilization of the core supply voltage regulator, which may

trigger LVD/HVD. It is therefore recommended to use PCFS (refer to MC_CGM chapter) to gradually change the system clock frequency and avoid sharp changes in the current load on the regulators.

59.1.4 External Supply

If an external regulator is used for VDD_LV supply, the Bipolar Junction Transistor (BJT) should be removed from the board. In that case, writing to the UTEST bit is required to disable the internal voltage regulator driver, which will remain active until RGM phase 3 (where the UTEST configuration is read). The VRC_CTRL pin may be left floating when using an external voltage regulator on VDD_LV supply.

59.1.5 Voltage Monitoring

The PMC senses the supply voltages continuously to keep the chip in safe operating range. Destructive or functional reset, depending on the PMC_RES register, or alternatively interrupts, are triggered when the voltage exceeds or falls below the limits defined by recommended electrical operating conditions. Please refer to the PMC_REE and PMC_RES register descriptions in the PMC_dig chapter for more details.

59.1.5.1 Core Supply Monitor

The digital core supply range is compared with a low and high voltage limit.

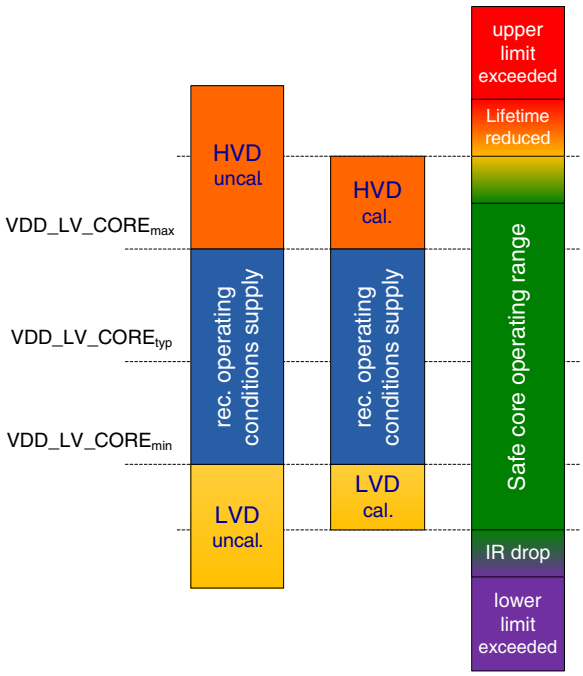


Figure 59-3. Core Supply Monitoring

The digital core is able to operate between the limits given by the MPC5746R Data Sheet. The lower limit is reduced by the amount of on-chip IR drop, that must be granted. The upper limit is defined by the beginning of lifetime degradation.

The available range for external supply voltage is further restricted by the tolerance of high (HVD) and low voltage monitors (LVD). See the MPC5746R Data Sheet for exact values. To ensure safe operating conditions for the core, the LVDs/HVDs are calibrated. Calibration of LVD/HVD takes places after the core is in the functionally safe operating range. It is performed in a way that LVD/HVD cover the allowed core operating range, i.e. by increasing the minimum value for LVD and decreasing the maximum value for HVD.

If LVD and HVD are disabled, the external core supply can have an extended operating range. External voltage monitors shall be used in that case to ensure functional safety. LVDs can be disabled using REE configuration in the UTEST area, and this is applied in RGM phase 3, before which the LVDs are still enabled (by default).

The LVD on the core supply is duplicated to monitor two points in the design: the maximum IR drop point (LVD_CORE_HOT) and the input voltage on the device point (LVD_CORE_COLD).

The core LVD/HVDs shall provide sufficient hysteresis or low-pass characteristic to prevent bouncing in case of short duration core supply over/undershoot.

59.1.5.2 IO, REG and other High Voltage Supplies Monitor

All 5 V supplied modules must remain fully functional in an extended operating range that includes the LVD calibrated tolerance. Calibration takes place when the core supply is within a safe operating range. After calibration, the typical LVD value is centered within the recommended operating conditions low limit and the safe operating conditions low limit to provide margins for supply undershoots.

If LVDs are disabled, external voltage monitors shall be used to ensure functional safety. The LVDs shall provide sufficient hysteresis or low-pass characteristic to prevent bouncing in case of short duration 5 V supply undershoots.

NOTE

All voltage monitors cannot be disabled; some of them are always active. Refer to "Voltage monitor electrical characteristics" in the Data Sheet for more details.

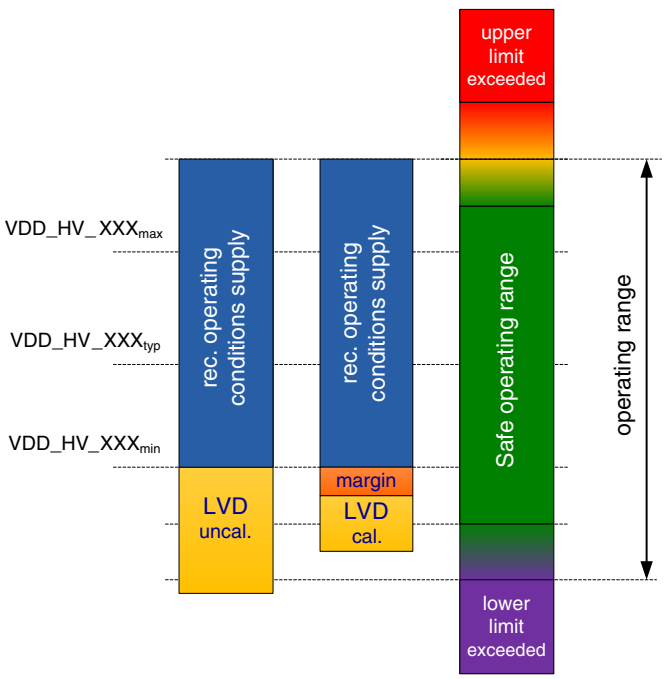


Figure 59-4. High Voltage Supply Monitoring

59.1.6 PMC Power-on Reset

The Power-on Reset (POR) Generator is required to initialize the chip during supply rise. It is a voltage monitor which detects if a supply exceeds a defined threshold voltage level. The POR checks the core voltage supply VDD_LV and the regulator supply VDD_HV_PMC. When both supplies have reliable levels for operation of the PMC and RESET generator, the internal PMC blocks, including LVD/HVD, are activated.

The POR watches the rising edge of the supply before the recommended operating conditions are reached. The POR shall function correctly even if the supply voltage is non-monotonic. It includes sufficient hysteresis to avoid bouncing in case of slow supply ramps. Within the safe operating conditions, the LVDs/HVDs take control over reset generation.

59.1.7 SoC Power-on Reset

The POR needs to hold the device in reset regardless of how slow the supply rise time is, until the point at which the soc_por is designed to release.

The figure below explains the logic for the generation of soc_por_b to the RGM, where soc_por_b is the combination of all of the PORs and auxiliary PORs from the analog PMC block and combinations of essential LVDs.

LVDs:

1. LVD_IO_MAIN for IRC and PORST / RESET pads
2. LVD_HV for PMC
3. LVD_FLASH for Flash

SOC_POR_MASK is the PMC self-test mode that overrides all POR/LVDs, and hence voltage monitoring will be disabled during LVD/HVD self-test.

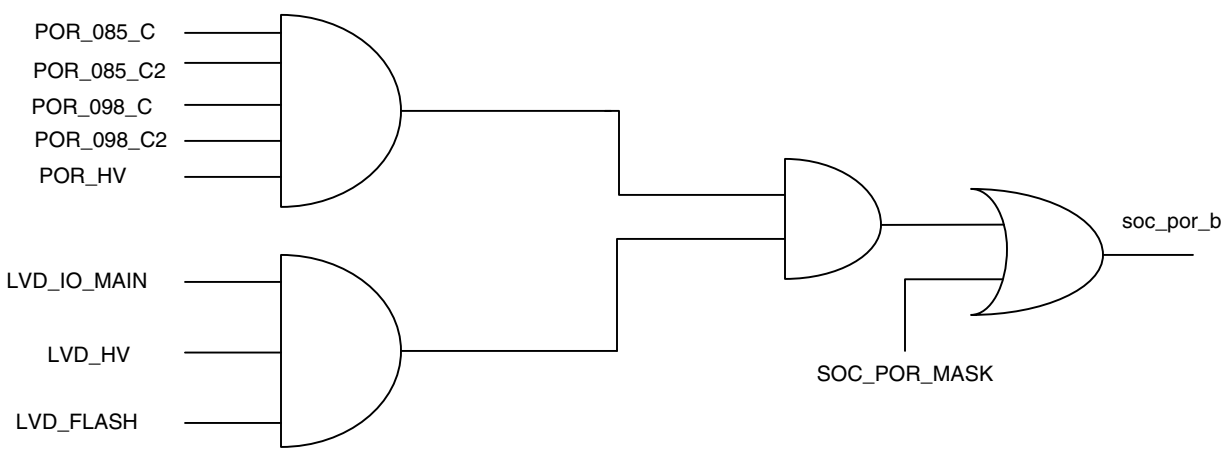


Figure 59-5. SoC POR Logic

59.1.8 Power-up Sequence

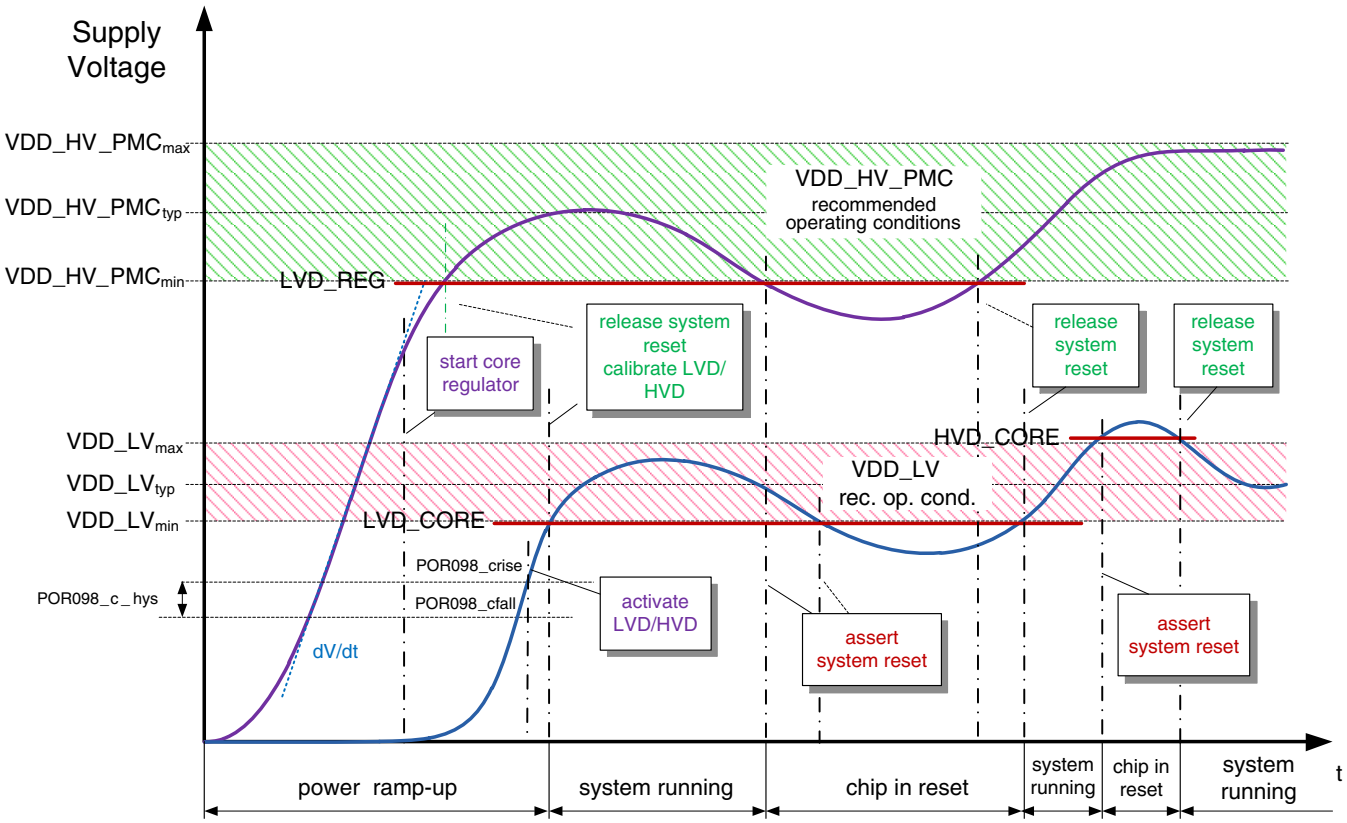


Figure 59-6. Chip Power-on RESET sequence timing diagram

The power-up sequence runs whenever the system will get through a power cycle or external reset.

When the VDD_HV_PMC voltage exceeds the POR_HV rising edge threshold, all HV LVDs are reset to their uncalibrated state and armed. The core voltage regulator CORE_REG starts to work and the core supply voltage rises.

When the core voltage crosses the POR098_c rising edge threshold, the core LVDs and HVDs are reset to their uncalibrated state and armed.

When the core supply and the regulator voltage reach their minimum value according to the recommended operating conditions, the system reset is released and the system boot process starts. The LVDs/HVDs are now calibrated during RGM phase 3 to provide tighter limits for voltage overshoot/undershoot detection.

As soon as the core or any supply (on which voltage monitoring is enabled) falls below the limit of the corresponding LVD thresholds, a reset is triggered; the type of reset is determined by the configuration. The system is held in reset state until all LVD threshold

levels are exceeded again. If the voltage exceeds the threshold of the HVD (on VDD_LV, VDD_HV_PMC or ADC supplies, which have HVD monitoring), a destructive reset is asserted. It is released when the voltage falls below the HVD threshold.

When core or regulator supply fall below the POR098_c or LVD_PMC falling edge limit, respectively, the system is reset and core supply is disabled. The chip will restart with a complete power-up cycle. The chip goes into a power-up cycle if any of the PORs or essential LVDs are triggered. These are:

1. All PORs and auxiliary PORs

2. LVDs:

-a. IO_MAIN for IRC and PORST / RESET pads

-b. PMC

-c. CORE_HOT

-d. Flash

59.1.9 ADC Interface

The ADC interface provides the PMC internal voltages for diagnostic purposes. The analog multiplexer and the buffer is inside the PMC. The SAR ADC0 converts the selected voltage, which can then be evaluated by the CPU. Refer to the PMC_dig chapter, PMC_ADC_CS register description for details.

Table 59-1. ADC interface

S.No.	pmc_adc_ch_sel[5:0]	ADC Channel Output	Expected Value
0	6'b000000	ADC Channel OFF	
1	6'b011110	por085_c sense value	1.25V (CORE Supply)
2	6'b011101	por085_c reference value	0.915V
3	6'b011100	por098_c sense value	1.25V (CORE Supply)
4	6'b011011	por098_c reference value	1.01V
5	6'b011010	por085_c2 sense	1.25V (CORE SUPPLY)
6	6'b011001	por085_c2 reference value	0.905V
7	6'b011000	por098_c2 sense value	1.25V (CORE SUPPLY)
8	6'b010111	por098_c2 reference value	0.993V
9	6'b010110	lvd_core_cold sense	1.25V (CORE SUPPLY)
10	6'b010101	lvd_core_cold reference	1.24V
11	6'b010100	lvd_core_hot sense	1.25V (CORE SUPPLY)
12	6'b010011	lvd_core_hot reference	1.21V

Table continues on the next page...

Table 59-1. ADC interface (continued)

13	6'b010010	hvd_core sense	1.25V (CORE SUPPLY)
14	6'b010001	hvd_core reference	1.43V
15	6'b010000	por260_c divider tap point	2.55V
16	6'b001111	por260_c reference	1.2V
17	6'b001110	por260_c2 divider tap point	2.55V
18	6'b001101	por260_c2 reference	1.2V
19	6'b001100	lvd_pmc divider tap point	1.98V
20	6'b001011	lvd_pmc reference	1.2V
21	6'b001010	hvd600 divider tap point	1.13V
22	6'b001001	hvd600 reference	1.2V
23	6b'001000	lvd_flash divider tap point	1.31V
24	6'b000111	lvd_flash reference	1.2V
25	6'b000110	hvd_flash divider tap point	1.13V
26	6'b000101	hvd_flash reference	1.2V
27	6'b000100	lvd_io divider tap point	1.97V
28	6'b000011	lvd_io reference	1.2V
29	6'b000010	lvd_fec divider tap point	1.48V
30	6'b000001	lvd_fec reference	1.2V
31	6'b100000	Reserved	
32	6'b111111	Reserved	
33	6'b111110	lvd_msc_5 divider tap point	1.48V
34	6'b111101	lvd_msc_5 reference	1.2V
35	6'b111100	lvd_jtag divider tap point	1.94V
36	6'b111011	lvd_jtag reference	1.2V
37	6'b111010	lvd_buddy reference	1.05V
38	6'b111001	lvd_buddy sense	1.2V(BUDDY Device Supply)
39	6'b111000	lvd_sar_adc divider tap point	2.25V
40	6'b110111	lvd_sar_adc reference	1.2V
41	6'b110110	hvd_sar_adc divider tap point	1.13V
42	6'b110101	hvd_sar_adc reference	1.2V
43	6'b110100	bandgap voltage with only curve trimming	1.205V
44	6'b110011	nwellbias regulator output	1.25V
45	6'b110010	nwellbias regulator reference	1.25V
59	6'b100100	ioseg3_sense	VDD_HV_IO Segment3
60	6'b100011	ioseg2_sense	VDD_HV_IO Segment2
61	6'b100010	vrefh_sd_adc	5V
62	6'b100001	vrefh_sd_adc	5V
63	6'b100000	vss_sd_adc	0
	Others	Reserved	

59.1.10 Safety Measures

The PMC implements a self-test architecture. In this test, all LVD/HVD voltage comparators are checked for stuck-at level faults by artificially triggering them to trip and checking that the comparator trips in both directions. Self-test is triggered automatically on each POR/destructive reset, and can be triggered by the user's software. The self-test will generate non-critical faults, which are collected in the FCCU module. There is also a UTEST bit which, if enabled, will cause a destructive reset if PMC self-test fails. The default state of the UTEST bit is "disabled".

Parts of the PMC design that can act as a single point of failure are replicated, such as the analog voltage reference generation circuits.

Various internal voltages can be monitored during PMC operation, with assistance of the ADC.

The PMC provides a signal that keeps the I/O pads in a reset condition if any of the voltage monitors is triggered, and the system goes into the destructive reset state.

Chapter 60
Power Management Controller digital interface (PMC_dig)

60.1 PMC_dig chip-specific module information

The Power Management Unit (PMU) implements a linear voltage regulator to generate the internal 1.25V digital supply from an external 3.5V to 5.5V supply. It also contains an option to bypass this regulator and instead use an externally supplied 1.25V input.

An additonal regulator for the flash block is also within the PMU to provide the 3.3V source necessary to the FLASH module.

Various voltage monitors also reside within the PMU, allowing the visibility of internal regualting points and external supplies. Other functional blocks such as the Power On reset (POR) are also available.

60.1.1 PMC registers whose reset values can be defined by DCF clients

Reset values for the following PMC registers are defined by the corresponding DCF clients:

- **PMC_REE_TD**: DCF_PMC_REE_TD DCF client
- **PMC_REE**: DCF_PMC_REE_CTRL DCF client
- **PMC_RES**: DCF_PMC_RES_CTRL DCF client

60.1.2 PMC registers with chip-specific reset values

The following PMC registers have chip-specific register values:

- **PMC_REE_TD**: 0x0000_0000 with DCF_PMC_REE_TD at default value
- **PMC_REE**: 0x0000_4052 with DCF_PMC_REE_CTRL at default value

- **PMC_RES**: 0x0000_0000 with DCF_PMC_RES_CTRL at default value
- **PMC_ADC_CS**: 0x0000_0100

60.1.3 Self test bit definition

The PMC_SELF_TEST_UM_VD_REG[VD_ST_CTRL] field is defined as follows for this chip.

Table 60-1. PMC_SELF_TEST_UM_VD_REG[VD_ST_CTRL] field bit definition

Field Value	Tested LVD/HVD
000000	Functional mode
000001	por085_core comparator test
000010	por098_core comparator test
000011	por085_core2 comparator test
000100	por098_core2 comparator test
000101	lvd_core_cold comparator test
000110	lvd_core_hot comparator test
000111	hvd_core comparator test
001000	por260_core comparator test
001001	por260_core2 comparator test
001010	lvd_pmc comparator test11
001011	hvd600_core comparator test
001100	lvd_flash comparator test
001101	hvd_flash comparator test
001110	lvd_io comparator test
001111 to 010001	Reserved
010010	lvd_sar_adc comparator test
010011 to 111111	Reserved

60.2 PMC introduction

The power management controller (PMC) consists of two blocks: an analog power management block (and a digital controller for the analog block).

The PMC_dig block contains all of the registers and digital logic to generate all of the enable signals, trim control bits, reset generation, and test mode logic for the analog block. The PMC digital logic also contains the controls for PMC analog outputs to be sensed with the ADC module.

In addition, the PMC digital logic contains the enable signals, trim bits and other registers for the temperature sensor. There are also several custom interfaces to various other blocks: The Reset Generation Module (RGM) block receives the LVD values during POR to use for phase transition conditions.

60.2.1 Features

The PMC module provides:

- Configuration and control information to the analog PMC block.
- An interface with Flash memory for retrieval of trim/control information, during POR, for HVDs, LVDs, regulators, and voltage bandgap.
- Configuration and monitoring of the two temperature sensors.
- Interrupt or reset generation for HVDs and LVDs (based on the configuration).
- Host access, via the IPS bus, to all register controls, configurations, and status.
- Self test logic for the LVDs and HVDs.
- Software triggered fault injection to the FCCU for LVDs, HVDs, temperature sensors, and LVD self test.
- 50MHz ipg clock frequency.
- Status and Control Register: LVD (Low-voltage detection) and HVD (high-voltage detection) for:-
  - PMC voltage supply (VDD_HV_PMC) - LVD
  - VDD core voltage supply - POR/LVD/HVD
  - VDD ADC voltage supply - LVD/HVD
  - Flash memory voltage supply - LVD
  - Oscillators/PLL voltage supply - LVD
  - VDDIO I/O voltage supply - LVD
- Status and control Register for Bandgap voltage reference:
  - LVDs and HVDs
  - ADC voltage reference channel
- DCF Client interface: An interface with Flash memory for retrieval of PMC configuration information from UTEST

60.3 Memory Map and Registers

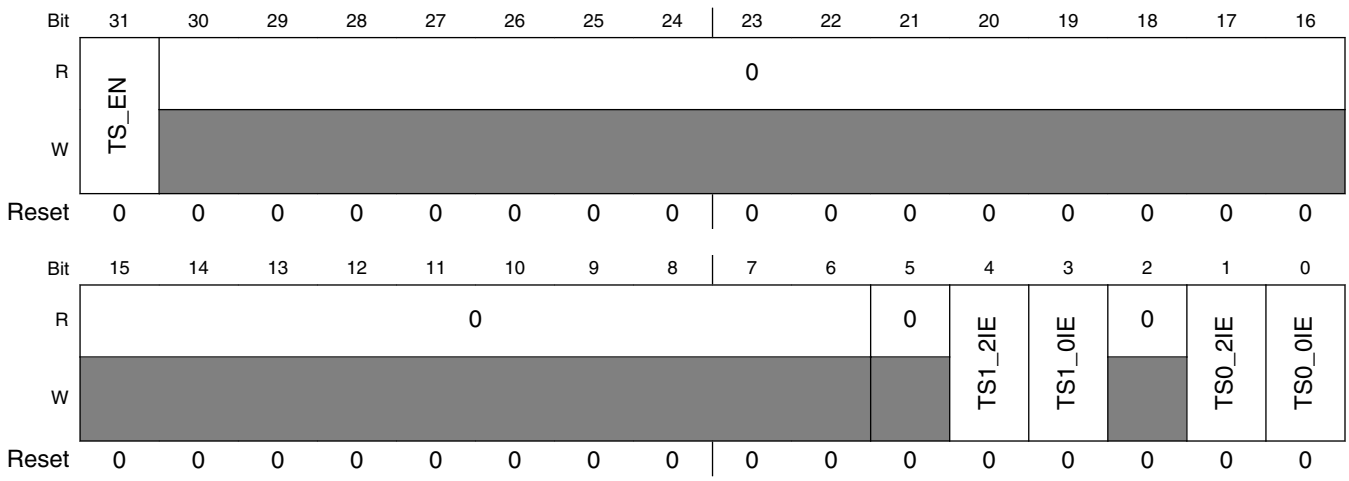
PMC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A4	Temperature Sensor Interrupt Enable Register (PMC_TS_IER)	32	R/W	0000_0000h	60.3.1/2575
A8	Temperature Event Status register (PMC_ESR_TD)	32	R/W	0000_0000h	60.3.2/2577
AC	Temperature Reset Event Enable register (PMC_REE_TD)	32	R/W	See section	60.3.3/2579
B0	Temperature Reset Event Selection register (PMC_RES_TD)	32	R/W	0000_0000h	60.3.4/2580
B4	Temperature detector configuration register (PMC_CTL_TD)	32	R/W	0000_0303h	60.3.5/2582
C0	Temperature Fault Injection Register (PMC_TS_FIR)	32	W	0000_0000h	60.3.6/2583
D0	Voltage Detect User Mode Test Register (PMC_SELF_TEST_UM_VD_REG)	32	R/W	0000_0000h	60.3.7/2584
D4	FCCU Fault Injection Register (PMC_FIR)	32	W	0000_0000h	60.3.8/2586
D8	LVD HVD Interrupt Enable Control Register (PMC_LVD_HVD_INTR_ENB)	32	R/W	0000_0000h	60.3.9/2587
DC	LVD HVD Event Status Register (PMC_LVD_HVD_EVENT_STATUS)	32	w1c	0000_0000h	60.3.10/ 2590
E0	Reset Enable Control Register (PMC_REE)	32	R/W	See section	60.3.11/ 2592
E4	Reset Type (Destructive / Functional) Selection Register (PMC_RES)	32	R/W	See section	60.3.12/ 2595
E8	ADC Channel Select Register (PMC_ADC_CS)	32	R/W	See section	60.3.13/ 2599

60.3.1 Temperature Sensor Interrupt Enable Register (PMC_TS_IER)

This configuration register contains a set of Interrupt Enable bits that correspond to each of the Temperature Sensor Event Status Registers (TS_ESR). These bits indicate whether an interrupt occurs when the voltage event is seen. A '0' indicates that no interrupt is to be sent, and a '1' indicates that an interrupt will occur when the voltage detect event occurs. These bits are enabled via a write of one to the MSB of this register (IE_EN). After this, these bits can be read or written at any time.

Address: 0h base + A4h offset = A4h



PMC_TS_IER field descriptions

Field	Description
31 TS_EN	Temperature Sensor interrupts Enable 0 Temperature sensor interrupts are not enabled. 1 Enable temperature sensor interrupts.
30–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 TS1_2IE	Temperature Sensor 1 input 2 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the temperature rises above its corresponding threshold (flag1). 0 No interrupt. 1 Interrupt occurs.
3 TS1_0IE	Temperature Sensor 1 input 0 Interrupt Enable.

Table continues on the next page...

PMC_TS_IER field descriptions (continued)

Field	Description
	This bit determines whether an interrupt is seen by the system when the temperature falls below its corresponding threshold. 0 No interrupt. 1 Interrupt occurs.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 TS0_2IE	Temperature Sensor 0 input 2 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the temperature rises above its corresponding threshold (flag1). 0 No interrupt. 1 Interrupt occurs.
0 TS0_0IE	Temperature Sensor 0 input 0 Interrupt Enable. This bit determines whether an interrupt is seen by the system when the temperature falls below its corresponding threshold. 0 No interrupt. 1 Interrupt occurs.

60.3.2 Temperature Event Status register (PMC_ESR_TD)

This Event Status Register indicates the past state of the two temperature sensor signals. If the temperature event has ever passed the trigger event since the last clearing event, the flag bit is set. In order to clear the flag, a one must be written to the flag bit that needs to be cleared.

Address: 0h base + A8h offset = A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0										0	TEMP1_2	TEMP1_0	0	TEMP0_2	TEMP0_0
W												w1c	w1c		w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMC_ESR_TD field descriptions

Field	Description
31–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 TEMP1_2	Temperature sensor 1 status flag 2. This bit is the temperature status flag associated with the temperature for the temperature sensor 1 point (150C). It is asserted when the temperature exceeds its corresponding threshold. It gets cleared by writing one to this same bit location, even if the temperature is still above the threshold.

Table continues on the next page...

PMC_ESR_TD field descriptions (continued)

Field	Description
	If the IER[TS1_2IE] bit is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP1_2} is also asserted, a system reset is generated, which clears IER[TS1_2IE] and negate the interrupt request. If REE_TD[TEMP1_2] is asserted and a destructive reset is selected (via RES_TD[TEMP_2] being 0), then a destructive reset is generated. If RES_TD[TEMP1_2] is set, then a functional reset is generated. 0 Currently no occurrence. 1 Temperature occurrence detected.
3 TEMP1_0	Temperature sensor 1 status flag 0. This bit is the temperature status flag associated with the temperature for the cold temperature sensor 1 point. It is asserted when the temperature gets below its corresponding threshold. It gets cleared by writing one to this same bit location, even if the temperature is still below the threshold. If the IER[TS1_0IE] bit is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP1_0] is also asserted, a system reset is generated, which clears IER[TS1_0IE] and negate the interrupt request. If REE_TD[TEMP1_0] is asserted and a destructive reset is selected (via RES_TD[TEMP_0] being 0), then a destructive reset is generated. If RES_TD[TEMP1_0] is set, then a functional reset is generated. 0 Currently no occurrence. 1 Temperature occurrence detected.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 TEMP0_2	Temperature sensor 0 status flag 2. This bit is the temperature status flag associated with the temperature for the temperature sensor point (150C). It is asserted when the temperature exceeds its corresponding threshold. It gets cleared by writing one to this same bit location, even if the temperature is still above the threshold. If the IER[TS0_2IE] bit is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP0_2 is also asserted, a system reset is generated, which clears IER[TS0_2IE] and negate the interrupt request. If REE_TD[TEMP0_2] is asserted and a destructive reset is selected (via RES_TD[TEMP0_2] being 0), then a destructive reset is generated. If RES_TD[TEMP_2] is set, then a functional reset is generated. 0 Currently no occurrence. 1 Temperature occurrence detected.
0 TEMP0_0	Temperature sensor 0 status flag 0. This bit is the temperature status flag associated with the temperature for the cold temperature sensor point. It is asserted when the temperature gets below its corresponding threshold. It gets cleared by writing one to this same bit location, even if the temperature is still below the threshold. If the IER[TS0_0IE] bit is also asserted, an interrupt is sent to the CPU. If REE_TD[TEMP0_0] is also asserted, a system reset is generated, which clears IER[TS0_0IE] and negate the interrupt request. If REE_TD[TEMP0_0] is asserted and a destructive reset is selected (via RES_TD[TEMP0_0] being 0), then a destructive reset is generated. If RES_TD[TEMP0_0] is set, then a functional reset is generated. 0 Currently no occurrence. 1 Temperature occurrence detected.

60.3.3 Temperature Reset Event Enable register (PMC_REE_TD)

This Reset Event Enable Register controls whether the temperature detect signal event causes a reset. If the desired flag bit is enabled, the temperature detect event causes a reset to be generated when the selected temperature passes the trigger event.

These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared). If the flash loaded bits are programmed to be disabled (clear), these bits cannot be enabled (set).

Address: 0h base + ACh offset = ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0										0	TEMP1_2	TEMP1_0	0	TEMP0_2	TEMP0_0
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	*	*	0*	*	0

* Notes:
- The reset value is chip-specific. Refer to the PMC_dig chip-specific module information for the reset value.
- TEMP1_2 field:
- TEMP1_0 field:
- TEMP0_2 field:

PMC_REE_TD field descriptions

Field	Description
31–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 TEMP1_2	Temperature sensor 1 detect assertion reset enable 2. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP1_2] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.
3 TEMP1_0	Temperature sensor 1 detect assertion reset enable 0.

Table continues on the next page...

PMC_REE_TD field descriptions (continued)

Field	Description
	This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP1_0] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 TEMP0_2	Temperature sensor 0 detect assertion reset enable 2. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP0_2] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.
0 TEMP0_0	Temperature sensor 0 detect assertion reset enable 0. This bit defines whether a temperature detect assertion generates a system reset. The RES_TD[TEMP0_0] bit determines whether the reset is functional or destructive. 0 Disabled. Temperature Sensor detect does not cause system reset. 1 Enabled. Temperature Sensor detect causes system reset.

60.3.4 Temperature Reset Event Selection register (PMC_RES_TD)

This Reset Event Select Register controls whether the temperature sensor detect signal event causes a destructive or functional reset. If the desired flag bit is enabled, the temperature sensor event causes either a destructive or functional reset to be generated when the selected temperature passes the trigger event.

Address: 0h base + B0h offset = B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0										0	TEMP1_2	TEMP1_0	0	TEMP0_2	TEMP0_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

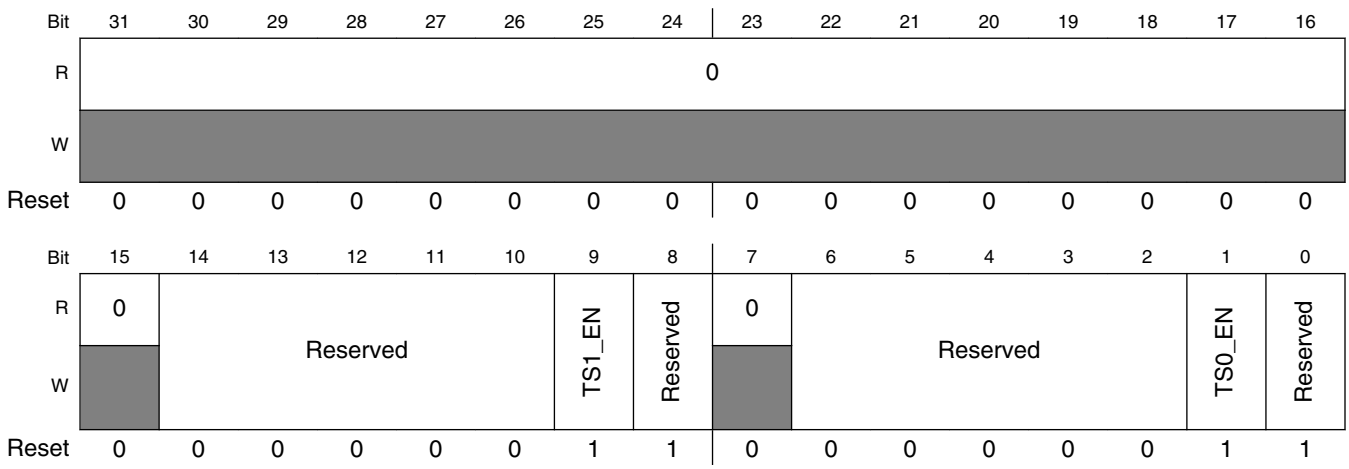
PMC_RES_TD field descriptions

Field	Description
31–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 TEMP1_2	TEMP1_2 Reset Event Select. This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset. The bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.
3 TEMP1_0	TEMP1_0 Reset Event Select. This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset. The bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 TEMP0_2	TEMP0_2 Reset Event Select. This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset. The bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.
0 TEMP0_0	TEMP0_0 Reset Event Select. This bit defines whether a temperature detect assertion generates a destructive reset or a functional reset. The bit determines whether the reset event is enabled. 0 Destructive Reset Generated. Temperature Sensor assertion causes a destructive system reset. 1 Functional Reset Generated. Temperature Sensor assertion causes a functional system reset.

60.3.5 Temperature detector configuration register (PMC_CTL_TD)

Temperature Sesnor Configuration register.

Address: 0h base + B4h offset = B4h



PMC_CTL_TD field descriptions

Field	Description
31–15 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
14–10 Reserved	This field is reserved. **NOTE:** Always write the reset value to this field.
9 TS1_EN	Temperature Sensor 1 Enable. 0 Disable output. 1 Enable output.
8 Reserved	This field is reserved. **NOTE:** Always write the reset value to this field.
7 Reserved	Reserved This field is reserved. This read-only field is reserved and always has the value 0.
6–2 Reserved	This field is reserved. **NOTE:** Always write the reset value to this field.
1 TS0_EN	Temperature Sensor 0 Enable. 0 Disable output. 1 Enable output.

Table continues on the next page...

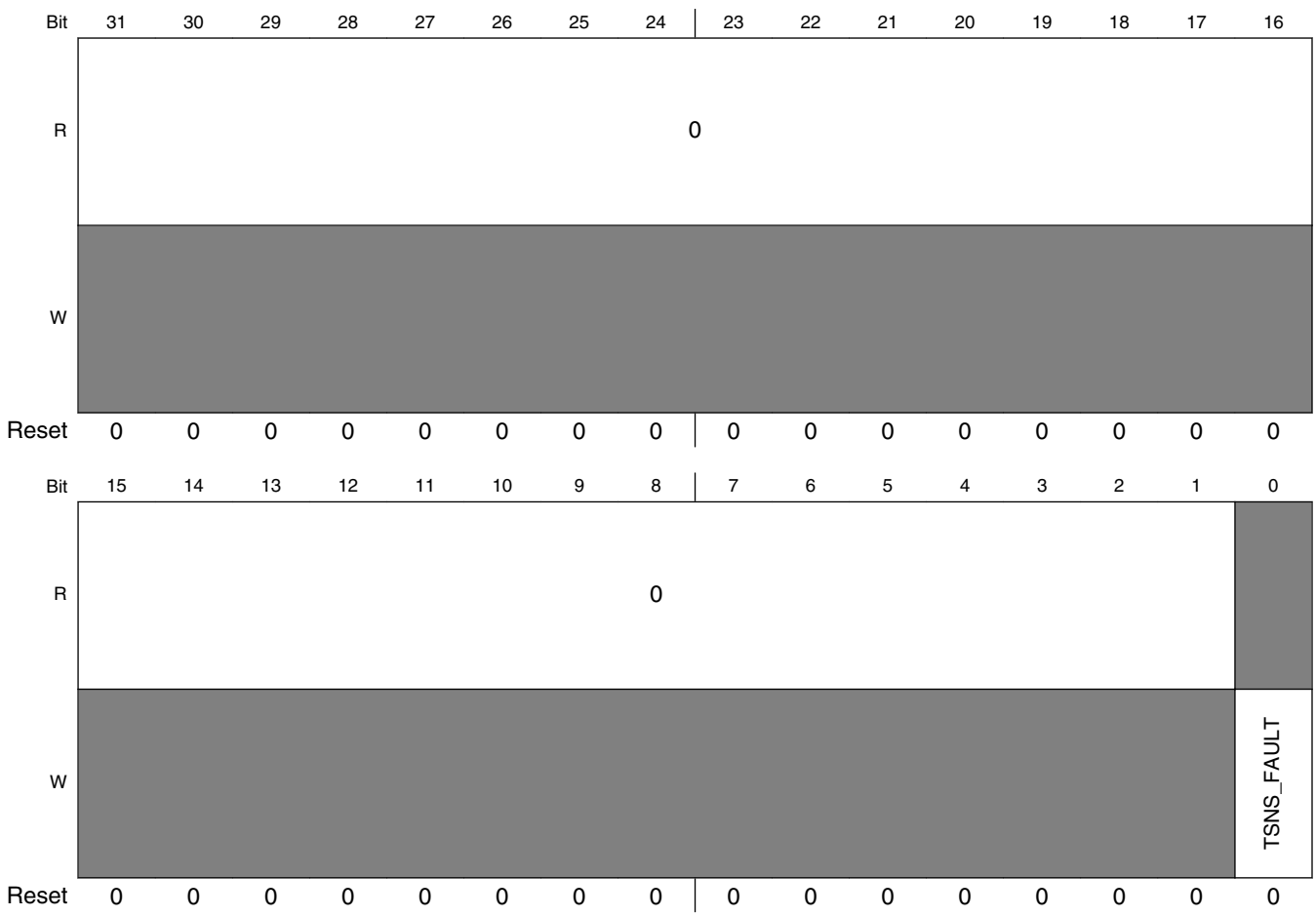
PMC_CTL_TD field descriptions (continued)

Field	Description
0 Reserved	This field is reserved. **NOTE:** Always write the reset value to this field.

60.3.6 Temperature Fault Injection Register (PMC_TS_FIR)

This FCCU Fault Injection register enables the user s/w to artificially generate FCCU faults corresponding to TEMP SENS. A write ‘1’ to these bits generates a single cycle pulsed fault to the FCCU block, this bit automatically gets cleared in the next cycle.

Address: 0h base + C0h offset = C0h



PMC_TS_FIR field descriptions

Field	Description
31–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

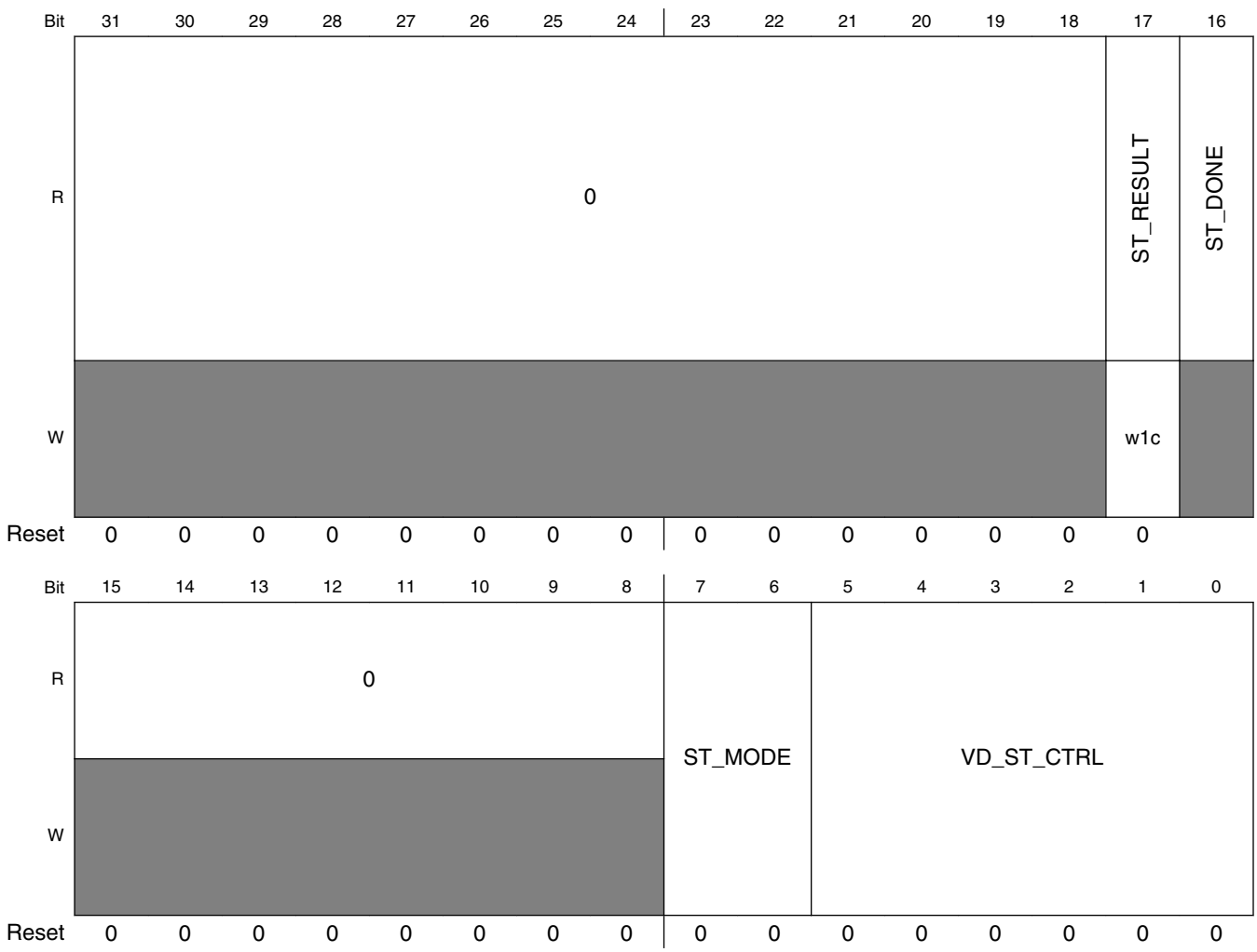
PMC_TS_FIR field descriptions (continued)

Field	Description
0 TSNS_FAULT	Temperature Sensor Fault injection. A write to this bit sends a FCCU fault (single cycle) for TEMP SENSOR. 0 No fault is sent. 1 Single cycle fault injected.

60.3.7 Voltage Detect User Mode Test Register (PMC_SELF_TEST_UM_VD_REG)

This register is used for testing of the LVDs and HVDs during user mode.

Address: 0h base + D0h offset = D0h



PMC_SELF_TEST_UM_VD_REG field descriptions

Field	Description
31–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 ST_RESULT	ST_RESULT. Self Test result status. (Valid only when ST_DONE is High). This bit indicates whether self test has passed or failed. Logic 0 means self test has failed. Logic High menas self test has passed. It is cleared on reset or writing ‘1’ by the host. 0 Self Test Failed 1 Self Test Passed
16 ST_DONE	Self Test Done status. Self Test completed indication status. If this bit is set to logic high, iit indicates that self test is completed. Logic High on this bit doesn't indicate whether self test has passed or failed. This bit gets clear auotmatically every time the self test gets triggered. 0 Self test running 1 Self test completed.
15–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7–6 ST_MODE	Self Test mode bits for testing of LVDs and HVDs. 00 Default 01 S/W triggered self test. 10 Single VD test as indicated by the VD_ST_CTRL bits. 11 Reserved
5–0 VD_ST_CTRL	Voltage Detect Self Test Control. User mode Test bits for testing of LVDs and HVDs. See the chip-specific module information at the beginning of this chapter for the test bit definition.

60.3.8 FCCU Fault Injection Register (PMC_FIR)

This FCCU Fault Injection register enables the user s/w to artificially generate FCCU faults corresponding to various Events, namely LVD, HVD, TEMP SENS and Self Test. A write '1' to these bits generates a single cycle pulsed fault to the FCCU block, this bit automatically gets cleared in the next cycle.

Address: 0h base + D4h offset = D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0													Reserved		
W													STEST_FAULT		HVD_FAULT	LVD_FAULT
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

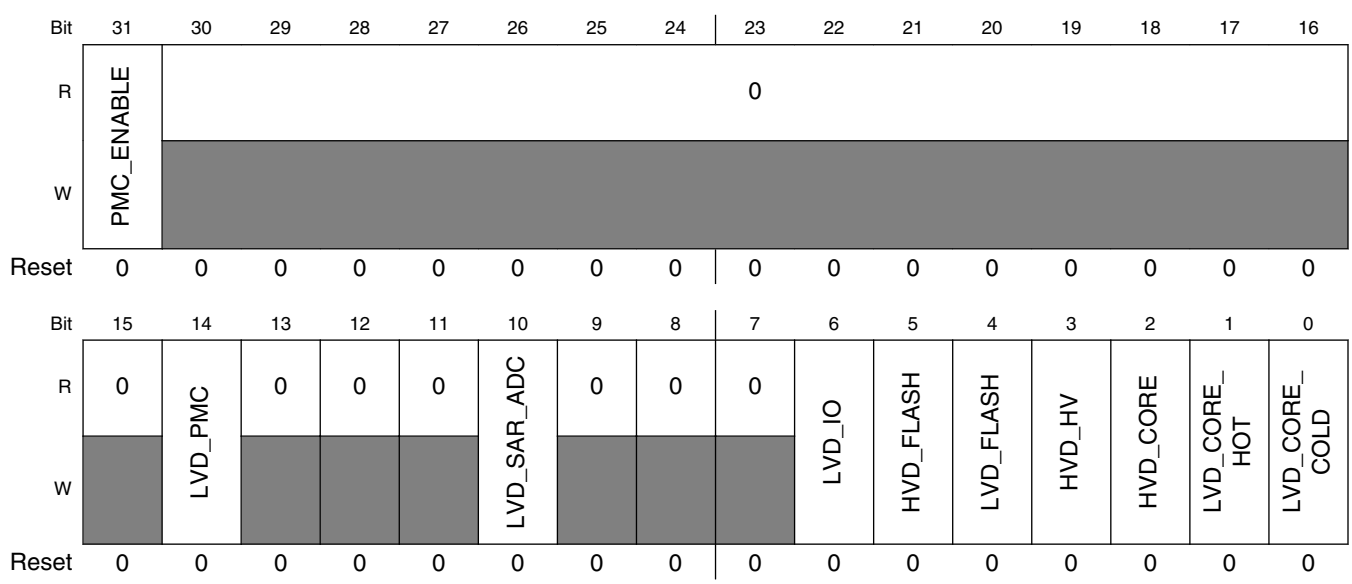
PMC_FIR field descriptions

Field	Description
31–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 STEST_FAULT	A write to this bit sends a FCCU fault (single cycle) for LVD SELF TEST. This is self Clearing bit 0 No fault is sent 1 Single cycle fault injected.
2 Reserved	This field is reserved.
1 HVD_FAULT	High Voltage Detect Fault injection. A write to this bit sends a FCCU fault (single cycle) for HVD. This is self clearing bit. 0 No fault is sent 1 Single cycle fault injected.
0 LVD_FAULT	Low Voltage Detect Fault injection. A write to this bit sends a FCCU fault (single cycle) for LVD. This is self clearing bit. 0 No fault is sent 1 Single cycle fault injected.

60.3.9 LVD HVD Interrupt Enable Control Register (PMC_LVD_HVD_INTR_ENB)

This Register contains the various control bits of PMC block.

Address: 0h base + D8h offset = D8h



PMC_LVD_HVD_INTR_ENB field descriptions

Field	Description
31 PMC_ENABLE	PMC Enable Control This bit determines whether any of the PMC bits can be written or not. 0 No control bit can be written in this(PMC enable control) register.To do a write on this register, first enable this bit and then start writing on other control bits. 1 This control register can be written.
30–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0. 0 Interrupt is disabled. 1 Interrupt is enabled.
14 LVD_PMC	LVD_PMC Control This bit controls the enable/disable on this supply. 0 Interrupt is disabled. 1 Interrupt is enabled.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0. 0 Interrupt is disabled. 1 Interrupt is enabled.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0. 0 Interrupt is disabled. 1 Interrupt is enabled.
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0. 0 Interrupt is disabled. 1 Interrupt is enabled.
10 LVD_SAR_ADC	LVD_SAR_ADC Control This bit controls the enable/disable on this supply. 0 Interrupt is disabled. 1 Interrupt is enabled.
9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0. 0 Interrupt is disabled. 1 Interrupt is enabled.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0. 0 Interrupt is disabled. 1 Interrupt is enabled.

Table continues on the next page...

PMC_LVD_HVD_INTR_ENB field descriptions (continued)

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0. 0 Interrupt is disabled. 1 Interrupt is enabled.
6 LVD_IO	LVD_IO Control This bit controls the enable/disable on this supply. 0 Interrupt is disabled. 1 Interrupt is enabled.
5 HVD_FLASH	HVD FLASH Control This bit controls the enable/disable on the LVD PLL supply. 0 Interrupt is disabled. 1 Interrupt is enabled.
4 LVD_FLASH	LVD FLASH Control This bit controls the enable/disable on this supply. 0 Interrupt is disabled. 1 Interrupt is enabled.
3 HVD_HV	HVD HV Control This bit controls the enable/disable on this supply. 0 Interrupt is disabled. 1 Interrupt is enabled.
2 HVD_CORE	HVD Core Control This bit controls the enable/disable on this supply. 0 Interrupt is disabled. 1 Interrupt is enabled.
1 LVD_CORE_ HOT	LVD Core Hot Control This bit controls the enable/disable on this supply. 0 Interrupt is disabled. 1 Interrupt is enabled.
0 LVD_CORE_ COLD	LVD Core Cold Control This bit controls the enable/disable on this supply. 0 Interrupt is disabled. 1 Interrupt is enabled.

60.3.10 LVD HVD Event Status Register (PMC_LVD_HVD_EVENT_STATUS)

This status register contains the latched status of the Voltage detect events. These bits indicates the state of the voltage detect events including whether a voltage level event has occurred at any time in the past but has not been cleared (via the writing '1'). These bits are read only, and they are cleared by writing a one to the corresponding ESR register bit.

NOTE

This register is cleared on POR.

Address: 0h base + DCh offset = DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															LVD_BUDDY
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	LVD_PMC	0	0	0	LVD_SAR_ADC	0	0	0	LVD_IO	HVD_FLASH	LVD_FLASH	HVD_HV	HVD_CORE	LVD_CORE_HOT	LVD_CORE_COLD
W		w1c				w1c				w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMC_LVD_HVD_EVENT_STATUS field descriptions

Field	Description
31–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 LVD_BUDDY	This bit indicates the status of LVD Buddy. **NOTE:** LVD buddy status should be completely ignored on PD-only parts. 0 No occurrence has happened. 1 An occurrence has happened.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0. 0 No occurrence has happened. 1 An occurrence has happened.
14 LVD_PMC	Indicates whether an occurrence has happened on this supply. 0 No occurrence has happened. 1 An occurrence has happened.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0. 0 No occurrence has happened. 1 An occurrence has happened.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0. 0 No occurrence has happened. 1 An occurrence has happened.
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0. 0 No occurrence has happened. 1 An occurrence has happened.
10 LVD_SAR_ADC	Indicates whether an occurrence has happened on this supply. 0 No occurrence has happened.. 1 An occurrence has happened.
9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0. 0 No occurrence has happened.. 1 An occurrence has happened.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0. 0 No occurrence has happened.. 1 An occurrence has happened.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PMC_LVD_HVD_EVENT_STATUS field descriptions (continued)

Field	Description
	0 No occurrence has happened. 1 An occurrence has happened.
6 LVD_IO	Indicates whether an occurrence has happened on this supply. 0 No occurrence has happened. 1 An occurrence has happened.
5 HVD_FLASH	Indicates whether an occurrence has happened on this supply. 0 No occurrence has happened. 1 An occurrence has happened.
4 LVD_FLASH	Indicates whether an occurrence has happened on this supply. 0 No occurrence has happened. 1 An occurrence has happened.
3 HVD_HV	Indicates whether an occurrence has happened on this supply. 0 No occurrence has happened. 1 An occurrence has happened.
2 HVD_CORE	Indicates whether an occurrence has happened on this supply. 0 No occurrence has happened. 1 An occurrence has happened.
1 LVD_CORE_ HOT	Indicates whether an occurrence has happened on this supply. 0 No occurrence has happened. 1 An occurrence has happened.
0 LVD_CORE_ COLD	Indicates whether an occurrence has happened on this supply. 0 No occurrence has happened. 1 An occurrence has happened.

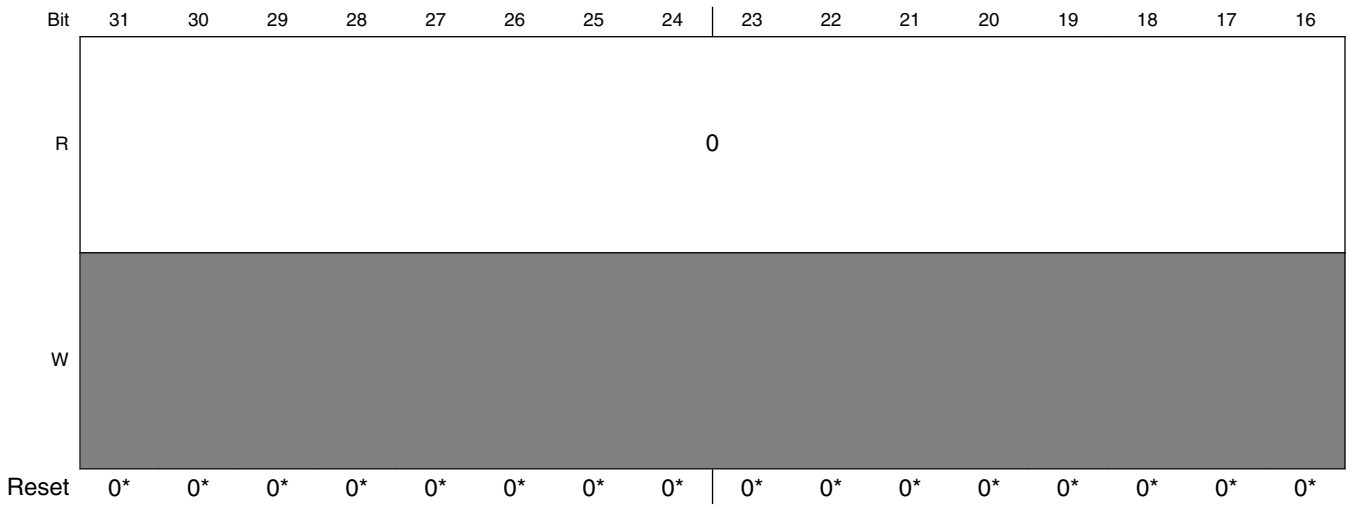
60.3.11 Reset Enable Control Register (PMC_REE)

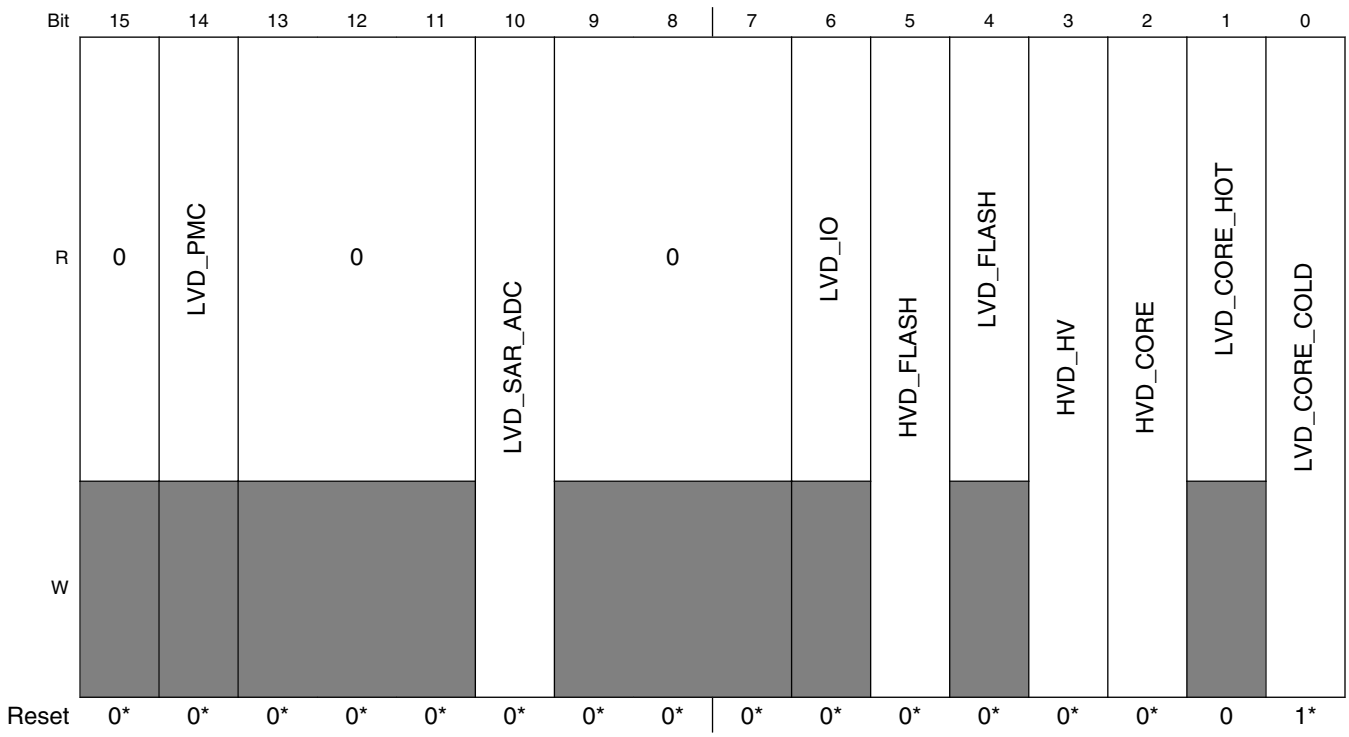
This Reset Event Enable Register controls whether the voltage detect signal event will cause a reset or not. If the desired flag bit is enabled, the voltage detect event causes a reset to be generated when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence. If the flash loaded bits are programmed to be enabled (set), these bits cannot be disabled (cleared).

NOTE

This register is cleared on POR.

Address: 0h base + E0h offset = E0h





* Notes:
- The reset value is chip-specific. Refer to the PMC_dig chip-specific module information for the reset value.

PMC_REE field descriptions

Field	Description
31–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 LVD_PMC	This bit enable/disable the generation of Reset.

Table continues on the next page...

PMC_REE field descriptions (continued)

Field	Description
	0 Reset is disabled. NO reset either functional or destructive, will be generated if an event is detected on this supply 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply
13–11 Reserved	Reserved. Read only zero. This field is reserved. This read-only field is reserved and always has the value 0. 0 Reset is disabled, no reset will be generated even if this event occurred. 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply
10 LVD_SAR_ADC	Reset Enable/Disable Control This bit enable/disable the generation of Reset. **NOTE:** Reset value is defined by the DCF_PMC_REE_CTRL DCF client. 0 Reset is disabled, no reset will be generated even if this event occurred. 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply
9–7 Reserved	Read This field is reserved. This read-only field is reserved and always has the value 0. 0 Reset is disabled, no reset will be generated even if this event occurred. 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply
6 LVD_IO	This bit enable/disable the generation of Reset. 0 Reset is Disabled. NO reset either functional or destructive, will be generated if an event is detected on this supply 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply
5 HVD_FLASH	Reset Enable/Disable Control This bit enable/disable the generation of Reset. **NOTE:** Reset value is defined by the DCF_PMC_REE_CTRL DCF client. 0 Reset is disabled, no reset will be generated even if this event occurred. 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply
4 LVD_FLASH	Reset Enable/Disable Control 0 Reset is Disabled. NO reset either functional or destructive, will be generated if an event is detected on this supply 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply
3 HVD_HV	Reset Enable/Disable Control This bit enable/disable the generation of Reset.

Table continues on the next page...

PMC_REE field descriptions (continued)

Field	Description
	NOTE: Reset value is defined by the DCF_PMC_REE_CTRL DCF client. 0 Reset is disabled, no reset will be generated even if this event occurred. 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply
2 HVD_CORE	Reset Enable/Disable Control This bit enable/disable the generation of Reset. **NOTE:** Reset value is defined by the DCF_PMC_REE_CTRL DCF client. 0 Reset is disabled, no reset will be generated even if this event occurred. 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply
1 LVD_CORE_ HOT	Reset Enable/Disable Control This bit enable/disable the generation of Reset. **NOTE:** Reset value is defined by the DCF_PMC_REE_CTRL DCF client. 0 Reset is disabled, no reset will be generated even if this event occurred. 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply
0 LVD_CORE_ COLD	Reset Enable/Disable Control This bit enable/disable the generation of Reset. **NOTE:** Reset value is defined by the DCF_PMC_REE_CTRL DCF client. 0 Reset is disabled, no reset will be generated even if there is event occurred on this supply. 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply.

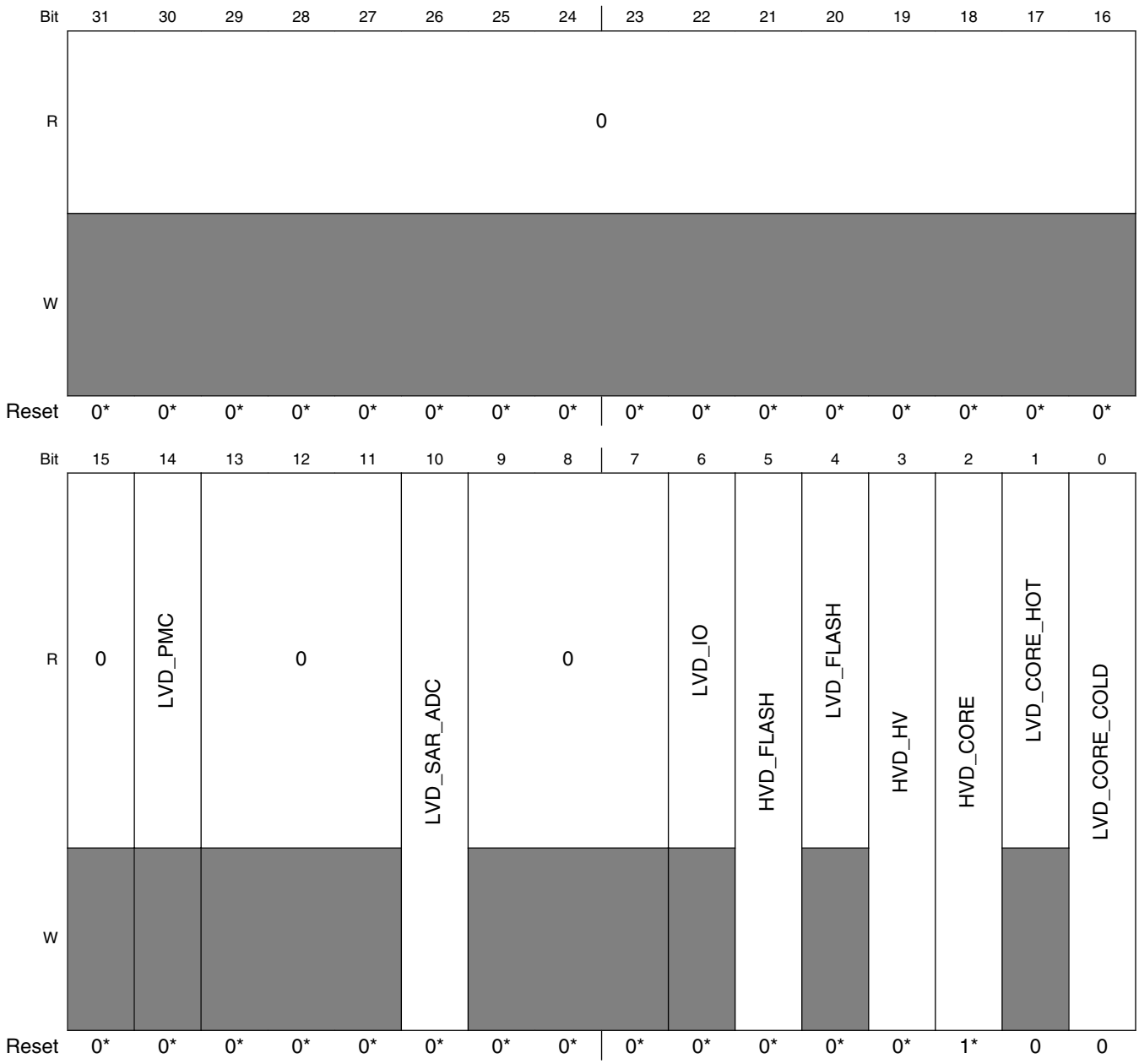
60.3.12 Reset Type (Destructive / Functional) Selection Register (PMC_RES)

This Reset Event Select Register controls whether the voltage detect signal event causes a destructive or functional reset. If the desired flag bit is enbaled in REE regsiter, the voltage detect event causes either a destructive or functional reset, when the selected voltage passes the trigger event. These bits are loaded from the flash during the boot sequence.

NOTE

This register is cleared on POR.

Address: 0h base + E4h offset = E4h



* Notes:

- The reset value is chip-specific. Refer to the PMC_dig chip-specific module information for the reset value.

PMC_RES field descriptions

Field	Description
31–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 LVD_PMC	Reset enabled/disabled.

Table continues on the next page...

PMC_RES field descriptions (continued)

Field	Description
	0 Reset is Disabled. NO reset either functional or destructive, will be generated if an event is detected on this supply 1 Reset is enabled. A reset either functional or destructive, will be generated if an event is detected on this supply
13–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0. 0 If the corresponding reset is enabled, then a Destructive Reset will be generated. 1 If the corresponding reset is enabled, then a Functional Reset will be generated.
10 LVD_SAR_ADC	This bit controls the enable/disable on the LVD SAR ADC supply. **NOTE:** Reset value is defined by the DCF_PMC_RES_CTRL DCF client. 0 If the corresponding reset is enabled, then a Destructive Reset will be generated. 1 If the corresponding reset is enabled, then a Functional Reset will be generated.
9–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0. 0 If the corresponding reset is enabled, then a Destructive Reset will be generated. 1 If the corresponding reset is enabled, then a Functional Reset will be generated.
6 LVD_IO	This bit enable/disable the generation of Reset. 0 If the corresponding reset is enabled, then a Destructive Reset will be generated. 1 If the corresponding reset is enabled, then a Functional Reset will be generated.
5 HVD_FLASH	This bit controls the enable/disable on the HVD FLASH supply. **NOTE:** Reset value is defined by the DCF_PMC_RES_CTRL DCF client. 0 If the corresponding reset is enabled, then a Destructive Reset will be generated. 1 If the corresponding reset is enabled, then a Functional Reset will be generated.
4 LVD_FLASH	This bit enable/disable the generation of Reset. 0 If the corresponding reset is enabled, then a destructive Reset will be generated. 1 If the corresponding reset is enabled, then a Functional Reset will be generated.
3 HVD_HV	This bit controls the enable/disable on the VDD_HV_PMC supply. **NOTE:** Reset value is defined by the DCF_PMC_RES_CTRL DCF client. 0 If the corresponding reset is enabled, then a Destructive Reset will be generated. 1 If the corresponding reset is enabled, then a Functional Reset will be generated.
2 HVD_CORE	This bit controls the enable/disable on the HVD Core supply. **NOTE:** Reset value is defined by the DCF_PMC_RES_CTRL DCF client. 0 If the corresponding reset is enabled, then a Destructive Reset will be generated. 1 If the corresponding reset is enabled, then a Functional Reset will be generated.
1 LVD_CORE_ HOT	This bit enable/disable the generation of Reset. 0 If the corresponding reset is enabled, then a destructive Reset will be generated. 1 If the corresponding reset is enabled, then a Functional Reset will be generated.

Table continues on the next page...

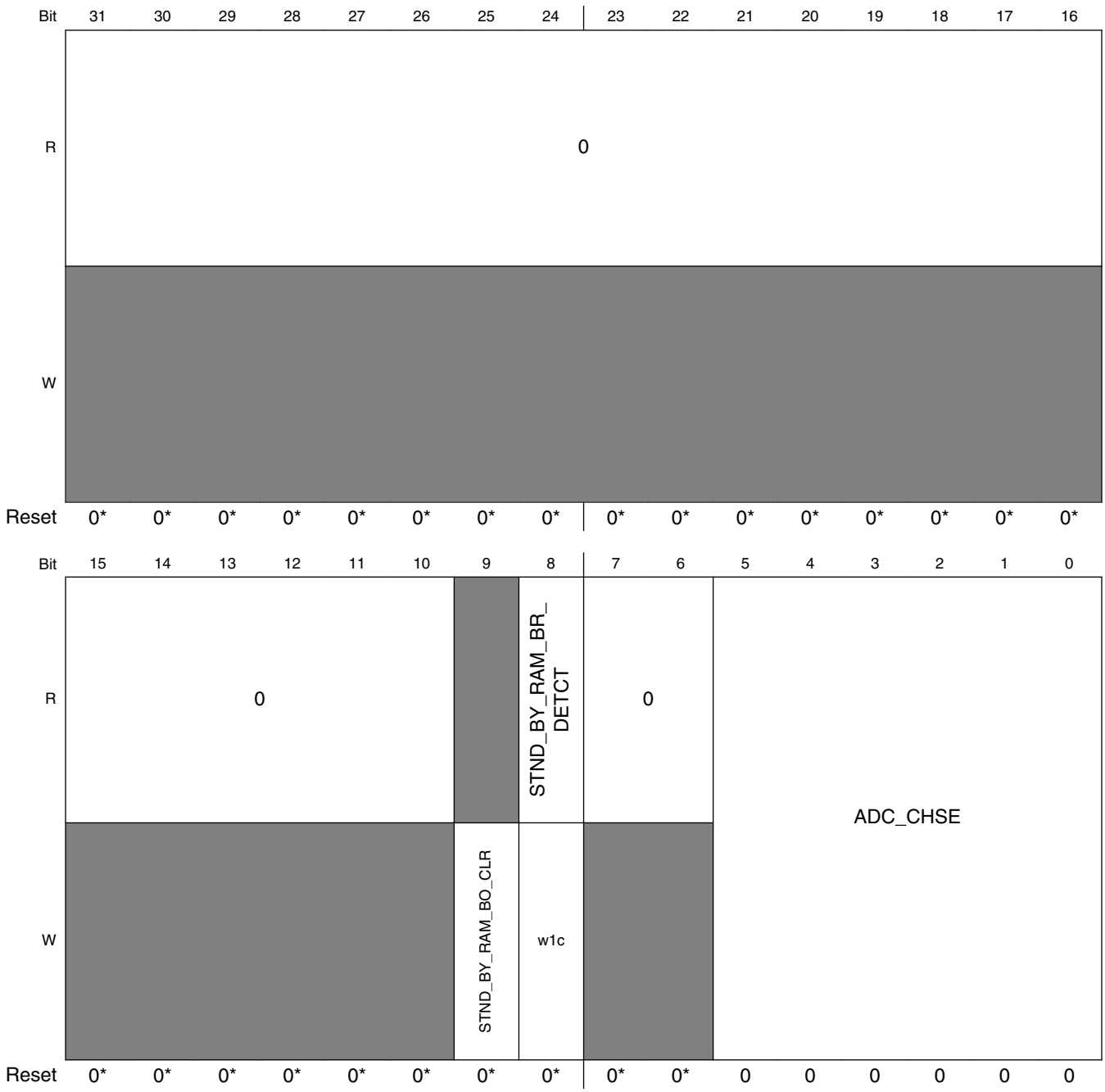
PMC_RES field descriptions (continued)

Field	Description
0 LVD_CORE_ COLD	LVD Core Enable Disable Bit This bit controls the enable/disable on the LVD Core supply. **NOTE:** Reset value is defined by the DCF_PMC_RES_CTRL DCF client. 0 If the corresponding reset is enabled, then a Destructive Reset will be generated. 1 If the corresponding reset is enabled, then a Functional Reset will be generated.

60.3.13 ADC Channel Select Register (PMC_ADC_CS)

This register contains the SRAM Brownout control and the analog channel mux of the analog PMC module.

Address: 0h base + E8h offset = E8h



* Notes:

- The reset value is chip-specific. Refer to the PMC_dig chip-specific module information for the reset value.

PMC_ADC_CS field descriptions

Field	Description
31–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 STND_BY_ RAM_BO_CLR	Standby RAM Brownout clear bit. A write on this bit will generate one clock wide pulse. It is self clear pulse.
8 STND_BY_ RAM_BR_ DETCT	Standby RAM Brownout Detection register bit. This bit is W1C type. To clear this field, first write 1 to STND_BY_RAM_BO_CLR and then write 1 to this field (STND_BY_RAM_BR_DETCT). These two writes need to be separate write instructions.
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–0 ADC_CHSE	Analog-to-Digital Converter Channel Select. These bits are used to control the analog PMC's test mux going to the ADC. This is to probe the internal node voltages of the PMC for testing, calibration and characterization purposes by one of the ADCs of the SoC. To see what voltages are output from PMC in this ADC interface, please refer to Table 60-16. **NOTE:** While changing internal PMC signal exposed on ADC interface for diagnostics, it is required to change mux select lines to '000000' before changing to next valid combination. This is needed to avoid transient short circuit b/w internal nodes in PMC and a false LVD / HVD / POR assertion. This must be handled through software.

60.4 Functional description

60.4.1 Analog PMC interface

The digital block uses the analog PMC interface to configure trim values for the regulators and LVDs that are in the analog block.

60.4.2 Temperature sensor interface logic

The digital PMC block provides control and configuration information to the two temperature sensors. A user adjustable trim control is also provided to fine tune the trimming (for more detail, see the detailed description of the Temperature Sensor module).

60.4.3 POR RGM phase gates

The interface with the RGM block is used during the power on sequence and startup. It provides various phase gates, thereby enabling the MC-RGM to properly complete the reset sequence.

It also provides functional and destructive reset sources, based on the configuration. Two types of resets are generated, based on LVD and HVD faults and the corresponding REE(reset event enable) and RES (Reset event status) enables:

- functional
- destructive

The temperature sensor provides another reset source, based on the under/over temperature faults and the corresponding REE and RES enables.

60.4.4 Flash interface to DCF

During power up, trim values for the regulators, LVDs, and HVDs are read from Flash memory by the SSCM, and are then provided to the digital PMC via the DCF interface.

The reset event enable (REE) configurations that are stored in Flash memory are also provided to the PMC via the DCF interface. These REEs configurations are used, by default, to generate various resets to the RGM.

60.4.5 FCCU interface

The digital block generates various fault indications to the FCCU block:

- Separate corresponding, active-high, fault indications are generated whenever an LVD or HVD defect is detected in the analog block.
- A fault indication is generated when a self test task results in a failure.
- A fault condition is generated whenever over-temperature or under-temperature events occur in the temperature sensor.

The figure below shows the exact logic for the generation of faults to FCCU block.

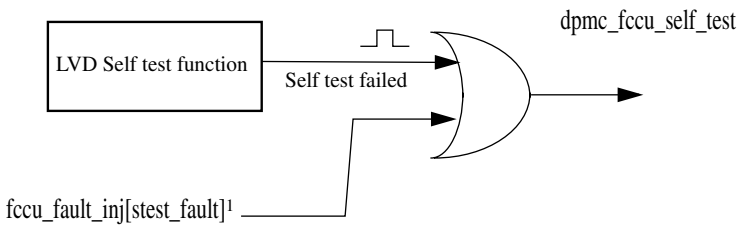


Figure 60-14. FCCU_fault_gen_LVD_Self_Test

[1] See FCCU Fault Injection Register 0(FIR) for details.

60.4.6 LVD-HVD self test mechanism

The PMC implements an LVD/HVD self test mechanism for use in satisfying safety goals. The digital block assists and controls the analog block during it's performance of the self test.

Currently, three self test modes (described in the following sections) are supported by the analog block (based on the mode specified by the VD_UTST[ST_MODE] register bit).

The self test status and result are provided in the VD_UTST[ST_DONE, ST_RESULT] register fields, respectively.

60.4.6.1 Default mode

The self test is always performed during start up (i.e power on). The self test starts during the phase 3 RGM state, when the SSCM has completed reads from the Flash memory. A counter is provided that starts from zero and increments by one after an 8 µs time gap. The counter value is the source for the output “pmc_lvd_self_test” signal, which is provided to the analog block.

Meanwhile, the LVD/HVD indication from the analog block is latched in the VD_UTST register. At the end of the self test (i.e. when all of the LVDs/HVDs have been tested), the VD_UTST[ST_DONE] status bit is set to '1' and all of the bits in the VD_UTST register are checked for ‘1’. One or more bits not set to '1' generates a fault indication to the FCCU block, and the VD_UTST[ST_RESULT] register bit is set to ‘0’. In the case

where all of the register bits are set to '1', the test is considered to be PASSED, the VD_UTST[ST_RESULT] register bit is set to ‘1’, and no fault indications are provided to the FCCU.

This startup self test function also provides gating of the reset phase3 exit in RGM. This means the RGM will not be able to exit from phase3 until the self test is successfully completed.

NOTE

The total run time of the LVD self test is equal to: ‘time required per LVD (20 µs)’ times 'number of LVDs (23)' = 460 µs.

60.4.6.2 Software-triggered self test

This mode is entered by software by writing '01' to the VD_UTST[ST_MODE] register 2-bit field. The complete self test, as is performed in default mode, is repeated. The difference is that the self test in this mode is initiated by the host while the self test in default mode is started automatically during power on.

60.4.6.3 Single voltage detect test

This mode is entered by software writing '10' to the VD_ST_CTRL[ST_MODE] register 2-bit field, which results in the assertion of the “pmc_lvd_self_test” input. The test completes after the prescribed time gap has elapsed. The PASS/FAIL result is then updated based on the corresponding LVD output tripping, or not.

NOTE

During the running of the LVD self test, all of the LVD and HVD outputs are masked including some PORs that are also under self test. Therefore, they will have no effect on POR generation, or in any reset and/or fault generation.

60.4.6.4 ADC Interface

ADC Interface is to probe the internal node voltages of PMC for testing, calibration and characterization purpose by one of the ADCs of the SoC. There will be a dedicated ADC channel for this interface.

PMC signals are buffered to ensure correct ADC sampling.

Node to be probed through ADC can be selected using bit "pmc_adc_ch_sel[5:0]".

Table 60-16. ADC Interface

S.No.	pmc_adc_ch_sel[5:0]	ADC Channel Output	Expected Value
0	6'b000000	ADC Channel OFF	
1	6’b011110	por085_c sense value	1.25V (CORE Supply)
2	6’b011101	por085_c reference value	0.915V
3	6’b011100	por098_c sense value	1.25V (CORE Supply)
4	6’b011011	por098_c reference value	1.01V
5	6'b011010	por085_c2 sense	1.25V (CORE SUPPLY)
6	6'b011001	por085_c2 reference value	0.905V
7	6'b011000	por098_c2 sense value	1.25V (CORE SUPPLY)
8	6'b010111	por098_c2 reference value	0.993V
9	6'b010110	lvd_core_cold sense	1.25V (CORE SUPPLY)
10	6'b010101	lvd_core_cold reference	1.24V
11	6'b010100	lvd_core_hot sense	1.25V (CORE SUPPLY)
12	6'b010011	lvd_core_hot reference	1.21V
13	6'b010010	hvd_core sense	1.25V (CORE SUPPLY)
14	6'b010001	hvd_core reference	1.43V
15	6'b010000	por260_c divider tap point	2.55V
16	6'b001111	por260_c reference	1.2V
17	6’b001110	por260_c2 divider tap point	2.55V
18	6’b001101	por260_c2 reference	1.2V
19	6’b001100	lvd_pmc divider tap point	1.98V
20	6’b001011	lvd_pmc reference	1.2V
21	6’b001010	hvd600 divider tap point	1.13V
22	6’b001001	hvd600 reference	1.2V
23	6b’001000	lvd_flash divider tap point	1.31V
24	6’b000111	lvd_flash reference	1.2V
25	6’b000110	hvd_flash divider tap point	1.13V
26	6’b000101	hvd_flash reference	1.2V
27	6’b000100	lvd_io divider tap point	1.97V
28	6’b000011	lvd_io reference	1.2V
29	6’b000010	Reserved	
30	6’b000001	Reserved	
31	6’b100000	Reserved	
32	6'b111111	Reserved	
33	6’b111110	Reserved	
34	6'b111101	Reserved	
35	6'b111100	Reserved	
36	6'b111011	Reserved	
37	6'b111010	lvd_buddy reference	1.05V

Table continues on the next page...

Table 60-16. ADC Interface (continued)

38	6'b111001	lvd_buddy sense	1.2V(BUDDY Device Supply)
39	6'b111000	lvd_sar_adc divider tap point	2.25V
40	6'b110111	lvd_sar_adc reference	1.2V
41	6'b110110	hvd_sar_adc divider tap point	1.13V
42	6'b110101	hvd_sar_adc reference	1.2V
43	6'b110100	bandgap voltage with only curve trimming	1.205V
44	6'b110011	nwellbias regulator output	1.25V
45	6'b110010	nwellbias regulator reference	1.25V
59	6'b100100	ioseg3_sense	VDD_HV_IO Segment3
60	6'b100011	ioseg2_sense	VDD_HV_IO Segment2
61	6'b100010	vrefh_sd_adc	5V
62	6'b100001	vrefh_sd_adc	5V
63	6'b100000	vss_sd_adc	0
	Others	Reserved	

Chapter 61
Power Control Unit (MC_PCU)

61.1 Introduction

The function of this module is simply to map the PMC digital interface IP into its address space.

The MC_PCU module plays a different role in this microcontroller than it does in other microcontrollers. Instead of directly providing an interface to configure and control power distribution to various segments ("power domains") of the chip, the MC_PCU acts as an integration module.

This microcontroller has a single power domain, Power Domain 0, which is managed by an analog Power Management Controller (PMC) and a PMC digital interface (PMC_dig). The 16K address space consists of a single MC_PCU power domain status register and the PMC_dig registers used to configure and control the analog PMC.

61.2 Memory map and register definition

All registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the PD0 field of the PCU_PSTAT register may be accessed as a word at offset 0x0040, as a half-word at offset 0x0042, or as a byte at offset 0x0043.

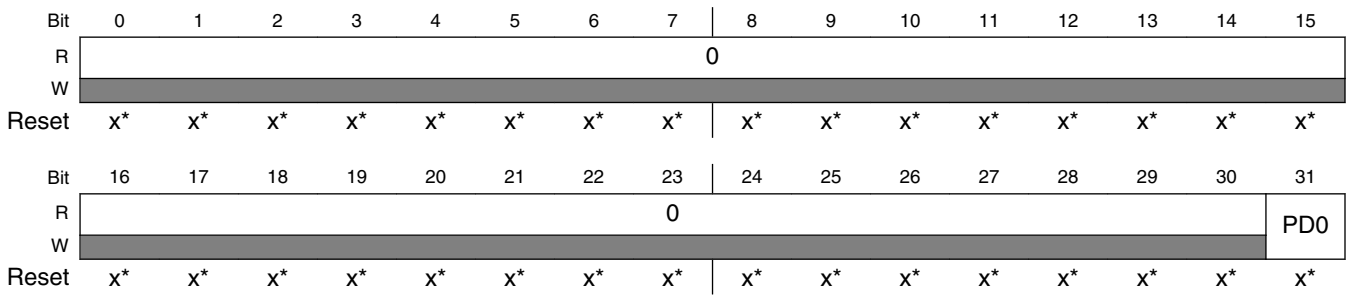
MC_PCU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
40	Power Domain Status Register (MC_PCU_PSTAT)	32	R	Undefined	61.2.1/2608

61.2.1 Power Domain Status Register (MC_PCU_PSTAT)

This register reflects the power status of all available power domains.

Address: 0h base + 40h offset = 40h



* Notes:
- x = Undefined at reset.

MC_PCU_PSTAT field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 PD0	Power status for power domain 0 0 Power domain is inoperable 1 Power domain is operable

61.3 Functional description

Please see the PMC_dig chapter for specific details on the PMC control registers.

Chapter 62
Mode Entry Module (MC_ME)

62.1 Introduction

This chapter describes the Mode Entry module.

62.1.1 Overview

The MC_ME controls the chip mode and mode transition sequences in all functional states. It also contains configuration, control, and status registers accessible for the application.

The purpose of the MC_ME is to minimize the current consumption of device modules by:

- Switching clocks on/off
- Switching clock sources on/off
- Enabling system clock progressive switching when changing to a different power level configuration

The following figure shows the MC_ME block diagram.

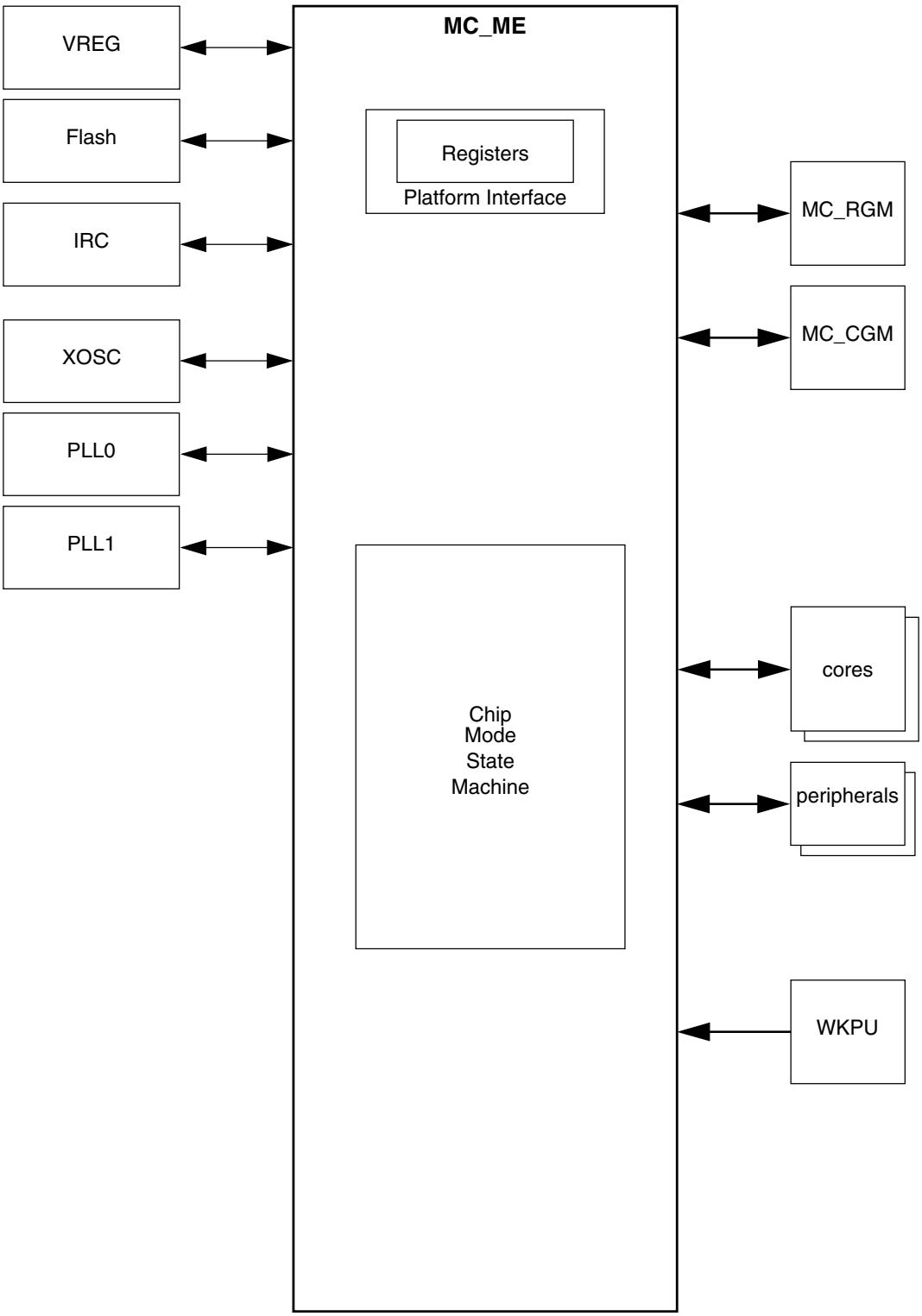


Figure 62-1. MC_ME Block Diagram

62.1.2 Features

The MC_ME includes the following features:

- control of the available modes by the **ME_ME** register
- definition of various chip mode configurations by the **ME_<mode>_MC** registers
- control of the actual chip mode by the **ME_MCTL** register for
  - switching between software-running modes
  - switching from software-running modes to low-power modes
  - switching to a mode to run diagnostics after a failure has been detected
  - triggering a device 'functional' reset which does not cause a start-up self test to run
  - triggering a device 'destructive' reset which may cause a start-up self test to run
- capture of the current mode and various resource status within the contents of the **ME_GS** register
- optional generation of various mode transition interrupts
- status bits for each cause of invalid mode transitions
- peripheral clock gating control based on the **ME_RUN_PC0…7**, **ME_LP_PC0…7**, and **ME_PCTL*n*** registers
- capture of current peripheral clock gated/enabled status
- core clock gating and boot control based on the **ME_CCTL*n*** and **ME_CADDR*n*** registers
- progressive system clock switching when transitioning from a lower power consumption mode to a higher power consumption mode, and vice versa
- capture of current core clock gated/enabled status

NOTE

It is possible to wake up from Stop/Halt mode through an interrupt even if it is not enabled at INTC level but just enabled at the module generating it.

62.1.3 Modes of Operation

The MC_ME is based on several chip modes corresponding to different usage models of the chip. Each mode is configurable and can define a policy for energy and processing power management to fit particular system requirements. An application can easily switch from one mode to another depending on the current needs of the system. The operating modes controlled by the MC_ME are divided into system and user modes. The system modes are modes such as ***RESET***, ***DRUN***, ***SAFE***, and ***TEST***. These modes aim to ease the configuration and monitoring of the system. The user modes are modes such as ***RUN0…3***, ***HALT0***, and ***STOP0*** which can be configured to meet the application requirements in terms of energy management and available processing power. The modes ***DRUN***, ***SAFE***, ***TEST***, and ***RUN0…3*** are the chip software running modes.

The following table describes the MC_ME modes.

Table 62-1. MC_ME Mode Descriptions (Continued)

Name	Description	Entry	Exit
RESET	This is a chip-wide virtual mode during which the application is not active. The system remains in this mode until all resources are available for the embedded software to take control of the chip. It manages hardware initialization of chip configuration, voltage regulators, clock sources, and flash modules.	system reset assertion from MC_RGM	system reset deassertion from MC_RGM
DRUN	This is the entry mode for the embedded software. It provides full accessibility to the system and enables the configuration of the system at startup. It provides the unique gate to enter user modes. BAM is executed in ***DRUN*** mode.	system reset deassertion from MC_RGM, software request from ***SAFE***, ***TEST*** and ***RUN0…3***	system reset assertion, ***RUN0…3***, ***TEST*** via software, ***SAFE*** via software or hardware failure.
SAFE	This is a chip-wide service mode which may be entered on the detection of a recoverable error. It forces the system into a pre-defined safe configuration from which the system may try to recover.	hardware failure, software request from any mode except ***RESET***	system reset assertion, ***DRUN*** via software
TEST	This is a chip-wide service mode which is intended to provide a control environment for chip software testing.	software request from ***DRUN***	system reset assertion, ***DRUN*** via software
RUN0…3	These are software running modes where most processing activity is done. These various run modes allow to enable different clock & power configurations of the system with respect to each other.	software request from ***DRUN*** or other ***RUN0…3***, interrupt event from ***HALT0***, interrupt or wakeup event from ***STOP0***	system reset assertion, ***SAFE*** via software or hardware failure, other ***RUN0…3*** modes, ***HALT0***, ***STOP0*** via software
HALT0	This is a reduced-activity low-power mode during which the clock to the core is disabled. It can be configured to switch off analog peripherals like clock sources, flash, main regulator, etc. for efficient power management at the cost of higher wakeup latency.	software request from ***RUN0…3***	system reset assertion, ***SAFE*** via software or hardware failure, ***RUN0…3*** on interrupt event

Table continues on the next page...

Table 62-1. MC_ME Mode Descriptions (Continued) (continued)

Name	Description	Entry	Exit
STOP0	This is an advanced low-power mode during which the clock to the core is disabled. It may be configured to switch off most of the peripherals including clock sources for efficient power management at the cost of higher wakeup latency.	software request from ***RUN0...3***	system reset assertion, ***SAFE*** via software or hardware failure, ***RUN0...3*** on interrupt event or wakeup event

62.2 External Signal Description

The MC_ME has no connections to any external pins.

62.3 MC_ME Registers

The following subsections show the MC_ME registers memory map and the detailed definition of each register.

The MC_ME contains registers for:

- mode selection and status reporting
- mode configuration
- mode transition interrupts status and mask control
- scalable number of peripheral sub-mode selection and status reporting
- scalable number of core enabling and disabling control and status reporting

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the ME_RUN_PC0 register may be accessed as a word at address offset 0x0080, as a half-word at address offset 0x0082, or as a byte at address offset 0x0083.

Some fields may be read-only, and their reset value of '1' or '0' and the corresponding behavior cannot be changed.

MC_ME memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Global Status (MC_ME_GS)	32	R	3013_0010h	62.3.1/2618
4	Mode Control (MC_ME_MCTL)	32	R/W	3000_A50Fh	62.3.2/2620

Table continues on the next page...

MC_ME memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8	Mode Enable (MC_ME_ME)	32	R/W	0000_801Dh	62.3.3/2622
C	Interrupt Status (MC_ME_IS)	32	R/W	0000_0000h	62.3.4/2624
10	Interrupt Mask (MC_ME_IM)	32	R/W	0000_0000h	62.3.5/2625
14	Invalid Mode Transition Status (MC_ME_IMTS)	32	R/W	0000_0000h	62.3.6/2627
18	Debug Mode Transition Status (MC_ME_DMTS)	32	R	0000_0000h	62.3.7/2628
20	RESET Mode Configuration (MC_ME_RESET_MC)	32	R	0013_0010h	62.3.8/2633
24	TEST Mode Configuration (MC_ME_TEST_MC)	32	R/W	0013_0010h	62.3.9/2635
28	SAFE Mode Configuration (MC_ME_SAFE_MC)	32	R/W	0093_0010h	62.3.10/ 2636
2C	DRUN Mode Configuration (MC_ME_DRUN_MC)	32	R/W	0013_0010h	62.3.11/ 2638
30	RUN0 Mode Configuration (MC_ME_RUN0_MC)	32	R/W	0013_0010h	62.3.12/ 2640
34	RUN1 Mode Configuration (MC_ME_RUN1_MC)	32	R/W	0013_0010h	62.3.13/ 2641
38	RUN2 Mode Configuration (MC_ME_RUN2_MC)	32	R/W	0013_0010h	62.3.14/ 2643
3C	RUN3 Mode Configuration (MC_ME_RUN3_MC)	32	R/W	0013_0010h	62.3.15/ 2645
40	HALT0 Mode Configuration (MC_ME_HALT0_MC)	32	R/W	0013_0010h	62.3.16/ 2646
48	STOP0 Mode Configuration (MC_ME_STOP0_MC)	32	R/W	0013_0010h	62.3.17/ 2648
60	Peripheral Status 0 (MC_ME_PS0)	32	R	0000_8000h	62.3.18/ 2650
64	Peripheral Status 1 (MC_ME_PS1)	32	R	0000_0000h	62.3.19/ 2652
68	Peripheral Status 2 (MC_ME_PS2)	32	R	0000_0000h	62.3.20/ 2654
6C	Peripheral Status 3 (MC_ME_PS3)	32	R	0000_0000h	62.3.21/ 2656
70	Peripheral Status 4 (MC_ME_PS4)	32	R	0000_0000h	62.3.22/ 2658
74	Peripheral Status 5 (MC_ME_PS5)	32	R	0000_0000h	62.3.23/ 2659
78	Peripheral Status 6 (MC_ME_PS6)	32	R	0000_0000h	62.3.24/ 2661
7C	Peripheral Status 7 (MC_ME_PS7)	32	R	0000_0000h	62.3.25/ 2663
80	Run Peripheral Configuration 0 (MC_ME_RUN_PC0)	32	R/W	0000_0000h	62.3.26/ 2665
84	Run Peripheral Configuration 0 (MC_ME_RUN_PC1)	32	R/W	0000_0000h	62.3.26/ 2665

Table continues on the next page...

MC_ME memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
88	Run Peripheral Configuration 0 (MC_ME_RUN_PC2)	32	R/W	0000_0000h	62.3.26/ 2665
8C	Run Peripheral Configuration 0 (MC_ME_RUN_PC3)	32	R/W	0000_0000h	62.3.26/ 2665
90	Run Peripheral Configuration 0 (MC_ME_RUN_PC4)	32	R/W	0000_0000h	62.3.26/ 2665
94	Run Peripheral Configuration 0 (MC_ME_RUN_PC5)	32	R/W	0000_0000h	62.3.26/ 2665
98	Run Peripheral Configuration 0 (MC_ME_RUN_PC6)	32	R/W	0000_0000h	62.3.26/ 2665
9C	Run Peripheral Configuration 0 (MC_ME_RUN_PC7)	32	R/W	0000_0000h	62.3.26/ 2665
A0	Low-Power Peripheral Configuration 0 (MC_ME_LP_PC0)	32	R/W	0000_0000h	62.3.27/ 2666
A4	Low-Power Peripheral Configuration 0 (MC_ME_LP_PC1)	32	R/W	0000_0000h	62.3.27/ 2666
A8	Low-Power Peripheral Configuration 0 (MC_ME_LP_PC2)	32	R/W	0000_0000h	62.3.27/ 2666
AC	Low-Power Peripheral Configuration 0 (MC_ME_LP_PC3)	32	R/W	0000_0000h	62.3.27/ 2666
B0	Low-Power Peripheral Configuration 0 (MC_ME_LP_PC4)	32	R/W	0000_0000h	62.3.27/ 2666
B4	Low-Power Peripheral Configuration 0 (MC_ME_LP_PC5)	32	R/W	0000_0000h	62.3.27/ 2666
B8	Low-Power Peripheral Configuration 0 (MC_ME_LP_PC6)	32	R/W	0000_0000h	62.3.27/ 2666
BC	Low-Power Peripheral Configuration 0 (MC_ME_LP_PC7)	32	R/W	0000_0000h	62.3.27/ 2666
C9	LFAST_0 Control (MC_ME_PCTL9)	8	R/W	00h	62.3.28/ 2667
CB	SIPI_0 Control (MC_ME_PCTL11)	8	R/W	00h	62.3.29/ 2668
DC	DECFILT_1 Control (MC_ME_PCTL28)	8	R/W	00h	62.3.30/ 2669
DD	DECFILT_0 Control (MC_ME_PCTL29)	8	R/W	00h	62.3.31/ 2669
DE	PIT_RTI_0 Control (MC_ME_PCTL30)	8	R/W	00h	62.3.32/ 2670
DF	PIT_RTI_1 Control (MC_ME_PCTL31)	8	R/W	00h	62.3.33/ 2670
E4	DMA_CH_MUX_0-2 Control (MC_ME_PCTL36)	8	R/W	00h	62.3.34/ 2671
E6	CRC_0 Control (MC_ME_PCTL38)	8	R/W	00h	62.3.35/ 2671

Table continues on the next page...

MC_ME memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
ED	JTAGM Control (MC_ME_PCTL45)	8	R/W	00h	62.3.36/ 2672
FB	ADCSD_2 Control (MC_ME_PCTL59)	8	R/W	00h	62.3.37/ 2672
FC	ADCSD_0 Control (MC_ME_PCTL60)	8	R/W	00h	62.3.38/ 2673
105	ADCSAR_2 Control (MC_ME_PCTL69)	8	R/W	00h	62.3.39/ 2673
107	ADCSAR_0 Control (MC_ME_PCTL71)	8	R/W	00h	62.3.40/ 2674
111	FLEXCAN_2 Control (MC_ME_PCTL81)	8	R/W	00h	62.3.41/ 2674
113	FLEXCAN_0 Control (MC_ME_PCTL83)	8	R/W	00h	62.3.42/ 2675
115	LINFLEX_M0 Control (MC_ME_PCTL85)	8	R/W	00h	62.3.43/ 2675
11B	LINFLEX_2 Control (MC_ME_PCTL91)	8	R/W	00h	62.3.44/ 2676
11C	LINFLEX_0 Control (MC_ME_PCTL92)	8	R/W	00h	62.3.45/ 2677
120	DSPI_M0 Control (MC_ME_PCTL96)	8	R/W	00h	62.3.46/ 2677
121	DSPI_4 Control (MC_ME_PCTL97)	8	R/W	00h	62.3.47/ 2678
122	DSPI_2 Control (MC_ME_PCTL98)	8	R/W	00h	62.3.48/ 2678
123	DSPI_0 Control (MC_ME_PCTL99)	8	R/W	00h	62.3.49/ 2679
126	EMIOS Control (MC_ME_PCTL102)	8	R/W	00h	62.3.50/ 2680
128	SENT_0 Control (MC_ME_PCTL104)	8	R/W	00h	62.3.51/ 2680
14F	DMA_CH_MUX_3 Control (MC_ME_PCTL143)	8	R/W	00h	62.3.52/ 2681
166	CRC_1 Control (MC_ME_PCTL166)	8	R/W	00h	62.3.53/ 2681
17C	ADCSD_1 Control (MC_ME_PCTL188)	8	R/W	00h	62.3.54/ 2682
184	ADCSAR_3 Control (MC_ME_PCTL196)	8	R/W	00h	62.3.55/ 2682
185	IGF Control (MC_ME_PCTL197)	8	R/W	00h	62.3.56/ 2683
186	ADCSAR_1 Control (MC_ME_PCTL198)	8	R/W	00h	62.3.57/ 2683

Table continues on the next page...

MC_ME memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
18A	ETPU_SCRAM Control (MC_ME_PCTL202)	8	R/W	00h	62.3.58/ 2684
18B	ETPU_SCRAM Control (MC_ME_PCTL203)	8	R/W	00h	62.3.59/ 2685
18C	ETPU_SPRAM Control (MC_ME_PCTL204)	8	R/W	00h	62.3.60/ 2685
18D	ETPU_SPRAM Control (MC_ME_PCTL205)	8	R/W	00h	62.3.61/ 2686
18E	REACM Control (MC_ME_PCTL206)	8	R/W	00h	62.3.62/ 2686
18F	ETPU Control (MC_ME_PCTL207)	8	R/W	00h	62.3.63/ 2687
190	FLEXCAN_3 Control (MC_ME_PCTL208)	8	R/W	00h	62.3.64/ 2688
192	FLEXCAN_1 Control (MC_ME_PCTL210)	8	R/W	00h	62.3.65/ 2688
195	LINFLEX_M1 Control (MC_ME_PCTL213)	8	R/W	00h	62.3.66/ 2689
19B	LINFLEX_3 Control (MC_ME_PCTL219)	8	R/W	00h	62.3.67/ 2689
19C	LINFLEX_1 Control (MC_ME_PCTL220)	8	R/W	00h	62.3.68/ 2690
1A1	DSPI_M1 Control (MC_ME_PCTL225)	8	R/W	00h	62.3.69/ 2690
1A2	DSPI_3 Control (MC_ME_PCTL226)	8	R/W	00h	62.3.70/ 2691
1A3	DSPI_1 Control (MC_ME_PCTL227)	8	R/W	00h	62.3.71/ 2691
1A6	EMIOS_1 Control (MC_ME_PCTL230)	8	R/W	00h	62.3.72/ 2692
1A8	SENT_1 Control (MC_ME_PCTL232)	8	R/W	00h	62.3.73/ 2692
1BB	BCTU Control (MC_ME_PCTL251)	8	R/W	00h	62.3.74/ 2693
1C0	Core Status (MC_ME_CS)	32	R	0000_0001h	62.3.75/ 2693
1C4	CORE1 Control (MC_ME_CCTL0)	16	R/W	00FEh	62.3.76/ 2695
1C6	CORE0 Control (MC_ME_CCTL1)	16	R/W	0000h	62.3.77/ 2696
1C8	CORE2 Control (MC_ME_CCTL2)	16	R/W	0000h	62.3.78/ 2697
1E0	CORE1 Address (MC_ME_CADDR0)	32	R/W	0100_0010h	62.3.79/ 2699

Table continues on the next page...

MC_ME memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1E4	CORE0 Address (MC_ME_CADDR1)	32	R/W	0000_0000h	62.3.80/ 2700
1E8	CORE2 Address (MC_ME_CADDR2)	32	R/W	0000_0000h	62.3.81/ 2700

62.3.1 Global Status (MC_ME_GS)

This register contains global mode status.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_CURRENT_MODE				S_MTRANS	1	0		S_PDO	0		S_MVR	Reserved		S_FLA	
W																
Reset	0	0	1	1	0	0	0	0	0	0	0	1	0	0	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								S_PLL1	S_PLL0	S_XOSC	S_IRC	S_SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

MC_ME_GS field descriptions

Field	Description
0–3 S_CURRENT_MODE	Current chip mode status 0000 RESET 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0
4 S_MTRANS	Mode transition status 0 Mode transition process is not active 1 Mode transition is ongoing
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
6–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 S_PDO	Output power-down status - This bit specifies output power-down status of I/Os. This bit is asserted whenever outputs of pads are forced to high impedance state or the pads power sequence driver is switched off. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and the pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 S_MVR	Main voltage regulator status 0 Main voltage regulator is not ready 1 Main voltage regulator is ready for use
12–13 Reserved	This field is reserved.
14–15 S_FLA	Flash availability status 00 Flash is not available 01 reserved 10 reserved 11 Flash is in normal mode and available for use
16–23 Reserved	This field is reserved.

Table continues on the next page...

MC_ME_GS field descriptions (continued)

Field	Description
24 S_PLL1	Secondary PLL status 0 secondary PLL is not stable 1 secondary PLL is providing a stable clock
25 S_PLL0	System PLL status 0 system PLL is not stable 1 system PLL is providing a stable clock
26 S_XOSC	4-40 MHz crystal oscillator status 0 4-40 MHz crystal oscillator is not stable 1 4-40 MHz crystal oscillator is providing a stable clock
27 S_IRC	16 MHz internal RC oscillator status 0 16 MHz internal RC oscillator is not stable 1 16 MHz internal RC oscillator is providing a stable clock
28–31 S_SYSCLK	System clock switch status - These bits specify the system clock currently used by the system. 0000 16 MHz int. RC osc. 0001 external osc. 0010 PLL0 0011 reserved 0100 PLL1 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled

62.3.2 Mode Control (MC_ME_MCTL)

This register is used to trigger software-controlled mode changes. Depending on the modes as enabled by ME_ME register bits, configurations corresponding to unavailable modes are reserved and access to ME_<mode>_MC registers must respect this for successful mode requests.

NOTE

Byte and half-word write accesses are not allowed for this register as a predefined key is required to change its value.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TARGET_MODE				0												KEY															
W																																
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1

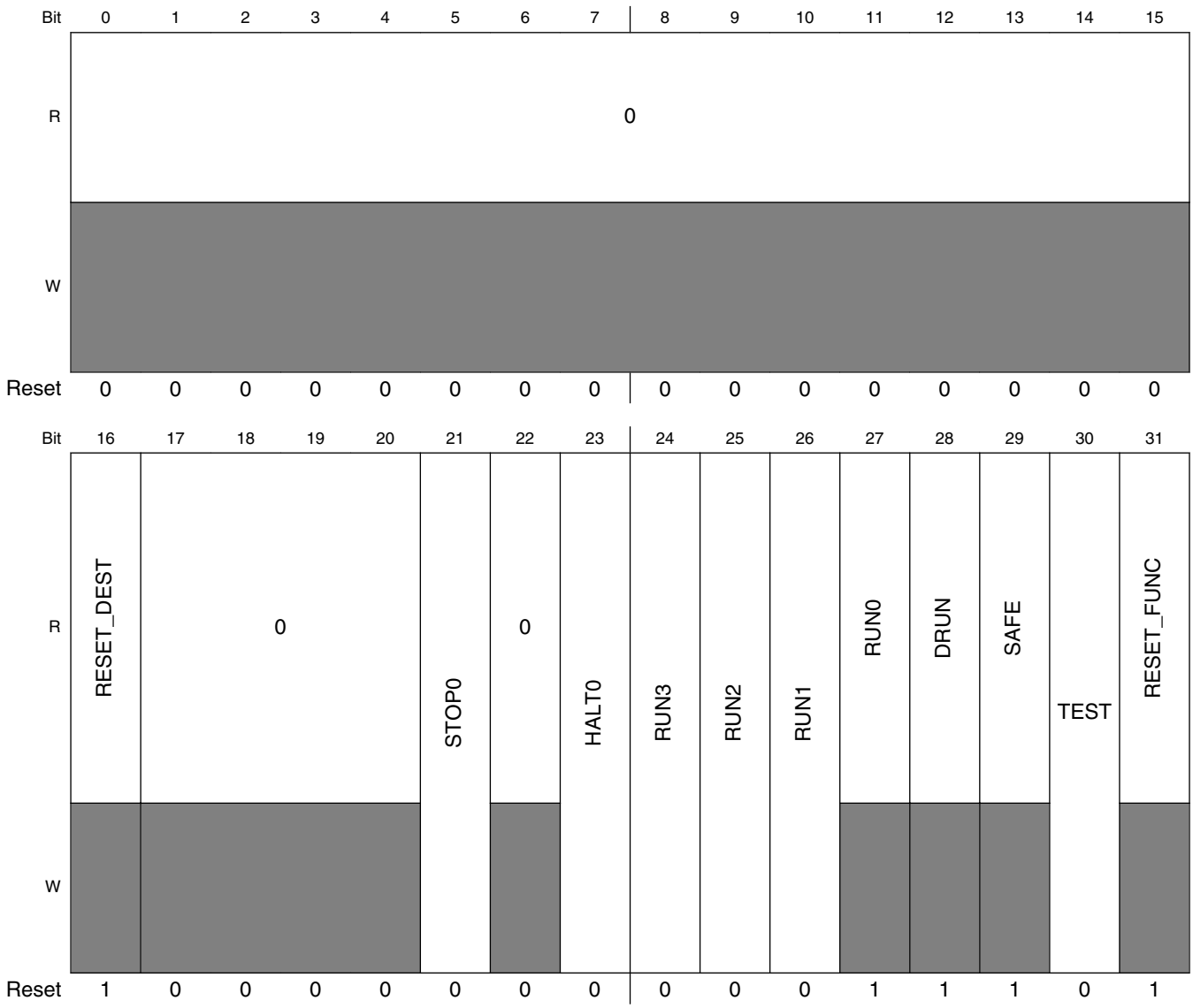
MC_ME_MCTL field descriptions

Field	Description
0–3 TARGET_MODE	Target chip mode -These bits provide the target chip mode to be entered by software programming. The mechanism to enter into any mode by software requires the write operation twice: first time with key, and second time with inverted key. These bits are automatically updated by hardware while entering SAFE on hardware request. Also, while exiting from the HALT0 and STOP0 modes on hardware exit events, these are updated with the appropriate RUN0-3 mode value. 0000 RESET (triggers a 'functional' reset event) 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 RESET (triggers a 'destructive' reset event)
4–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 KEY	Control key -These bits enable write access to this register. Any write access to the register with a value different from the keys is ignored. Read access will always return inverted key. KEY: 0101101011110000 (0x5AF0) INVERTED KEY: 1010010100001111 (0xA50F)

62.3.3 Mode Enable (MC_ME_ME)

This register allows a way to disable the chip modes which are not required for a given chip. RESET, SAFE, DRUN, and RUN0 modes are always enabled.

Address: 0h base + 8h offset = 8h



MC_ME_ME field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 RESET_DEST	'Destructive' RESET mode enable 0 'Destructive' RESET mode is disabled

Table continues on the next page...

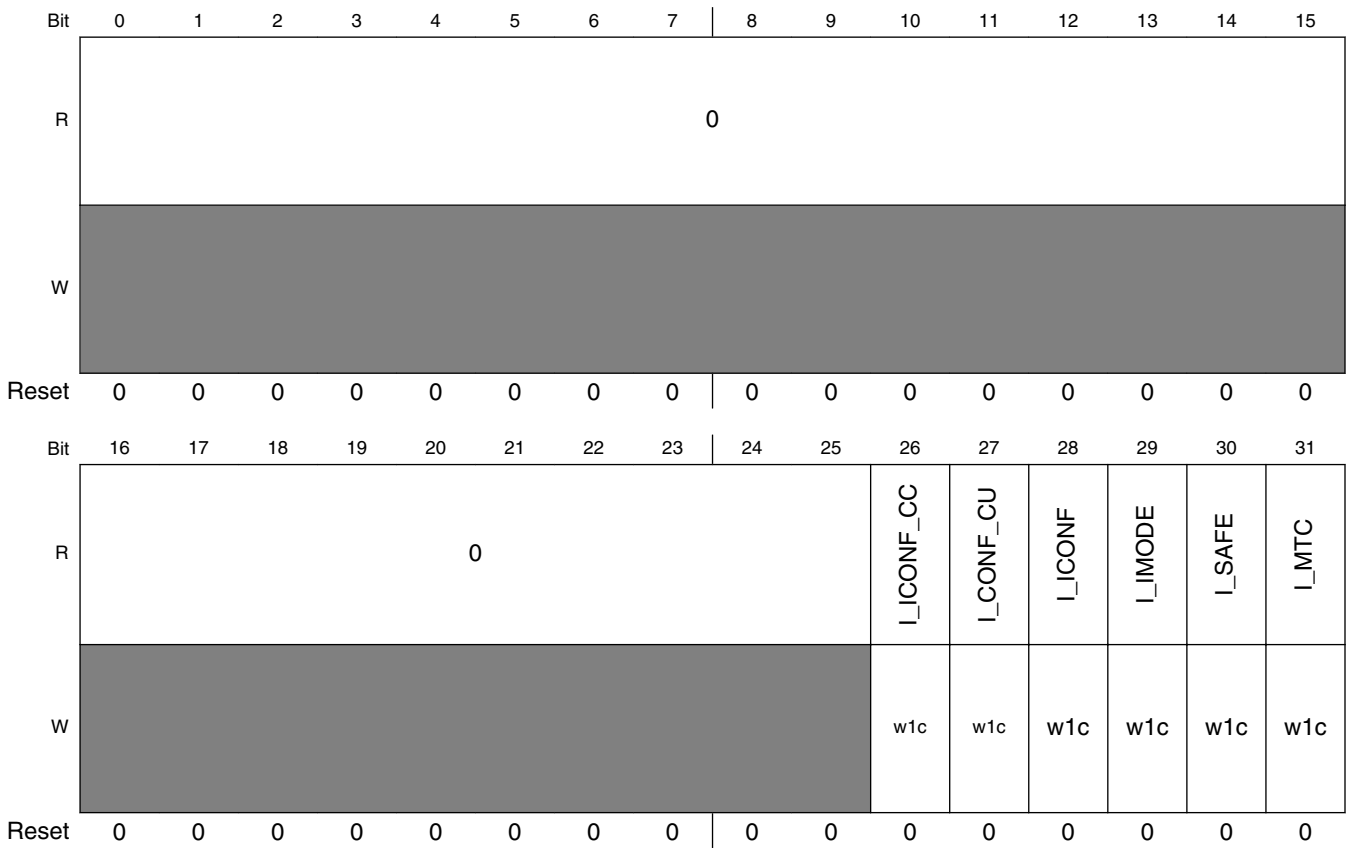
MC_ME_ME field descriptions (continued)

Field	Description
	1 'Destructive' RESET mode is enabled
17–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 STOP0	STOP0 mode enable 0 STOP0 mode is disabled 1 STOP0 mode is enabled
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 HALT0	HALT0 mode enable 0 HALT0 mode is disabled 1 HALT0 mode is enabled
24 RUN3	RUN3 mode enable 0 RUN3 mode is disabled 1 RUN3 mode is enabled
25 RUN2	RUN2 mode enable 0 RUN2 mode is disabled 1 RUN2 mode is enabled
26 RUN1	RUN1 mode enable 0 RUN1 mode is disabled 1 RUN1 mode is enabled
27 RUN0	RUN0 mode enable 1 RUN0 mode is enabled
28 DRUN	DRUN mode enable 1 DRUN mode is enabled
29 SAFE	SAFE mode enable 1 SAFE mode is enabled
30 TEST	TEST mode enable 0 TEST mode is disabled 1 TEST mode is enabled
31 RESET_FUNC	'functional' RESET mode enable 0 'functional' RESET mode is disabled 1 'functional' RESET mode is enabled

62.3.4 Interrupt Status (MC_ME_IS)

This register provides the current interrupt status.

Address: 0h base + Ch offset = Ch



MC_ME_IS field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 I_ICONF_CC	Invalid mode configuration interrupt (core configuration) - This bit is set if a write access to one of the ME_CADDRn registers is attempted while a mode transition is in progress. 0 No write to an ME_CADDRn register was attempted during an ongoing mode transition 1 A write to an ME_CADDRn register was attempted during an ongoing mode transition
27 I_CONF_CU	Invalid mode configuration interrupt (clock usage) - This bit is set during a mode transition if a clock which is required to be on by an enabled peripheral is configured to be turned off. It is cleared by writing a '1' to this bit. 0 No invalid mode configuration (clock usage) interrupt occurred 1 Invalid mode configuration (clock usage) interrupt is pending
28 I_ICONF	Invalid mode configuration interrupt - This bit is set whenever a write operation to ME_<mode>_MC registers with invalid mode configuration is attempted. It is cleared by writing a '1' to this bit.

Table continues on the next page...

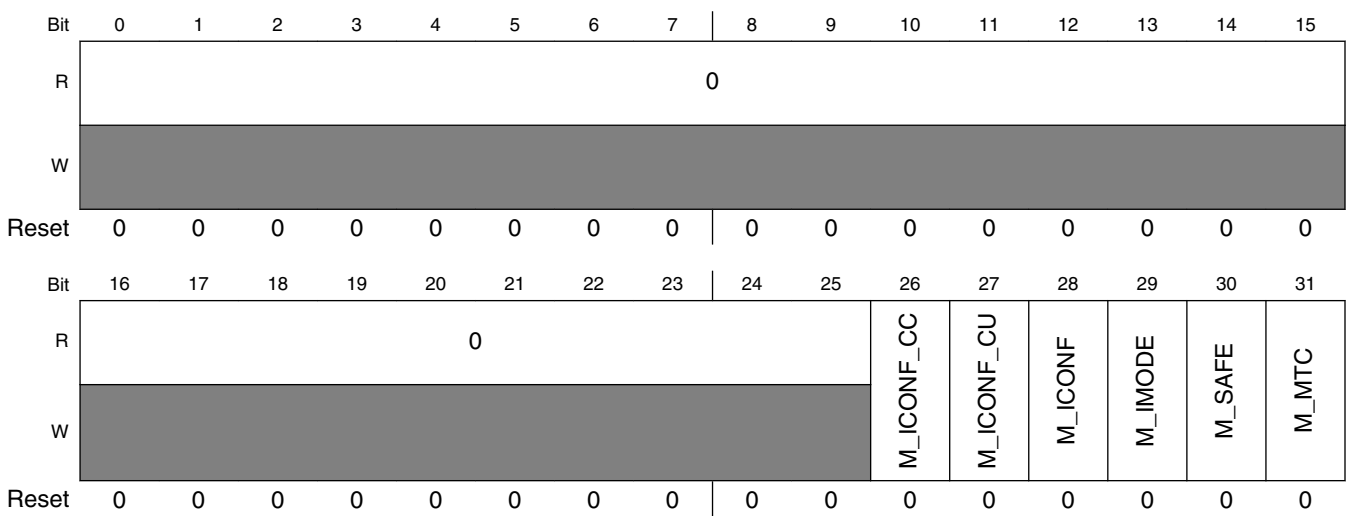
MC_ME_IS field descriptions (continued)

Field	Description
	0 No invalid mode configuration interrupt occurred 1 Invalid mode configuration interrupt is pending
29 I_IMODE	Invalid mode interrupt - This bit is set whenever an invalid mode transition is requested. It is cleared by writing a '1' to this bit. 0 No invalid mode interrupt occurred 1 Invalid mode interrupt is pending
30 I_SAFE	SAFE mode interrupt - This bit is set whenever the chip enters SAFE mode on hardware requests generated in the system. It is cleared by writing a '1' to this bit. 0 No SAFE mode interrupt occurred 1 SAFE mode interrupt is pending
31 I_MTC	Mode transition complete interrupt - This bit is set whenever the mode transition process completes (S_MTRANS transits from 1 to 0). It is cleared by writing a '1' to this bit. This mode transition interrupt bit will not be set while entering low-power modes HALT0, or STOP0. 0 No mode transition complete interrupt occurred 1 Mode transition complete interrupt is pending

62.3.5 Interrupt Mask (MC_ME_IM)

This register controls whether an event generates an interrupt or not.

Address: 0h base + 10h offset = 10h



MC_ME_IM field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

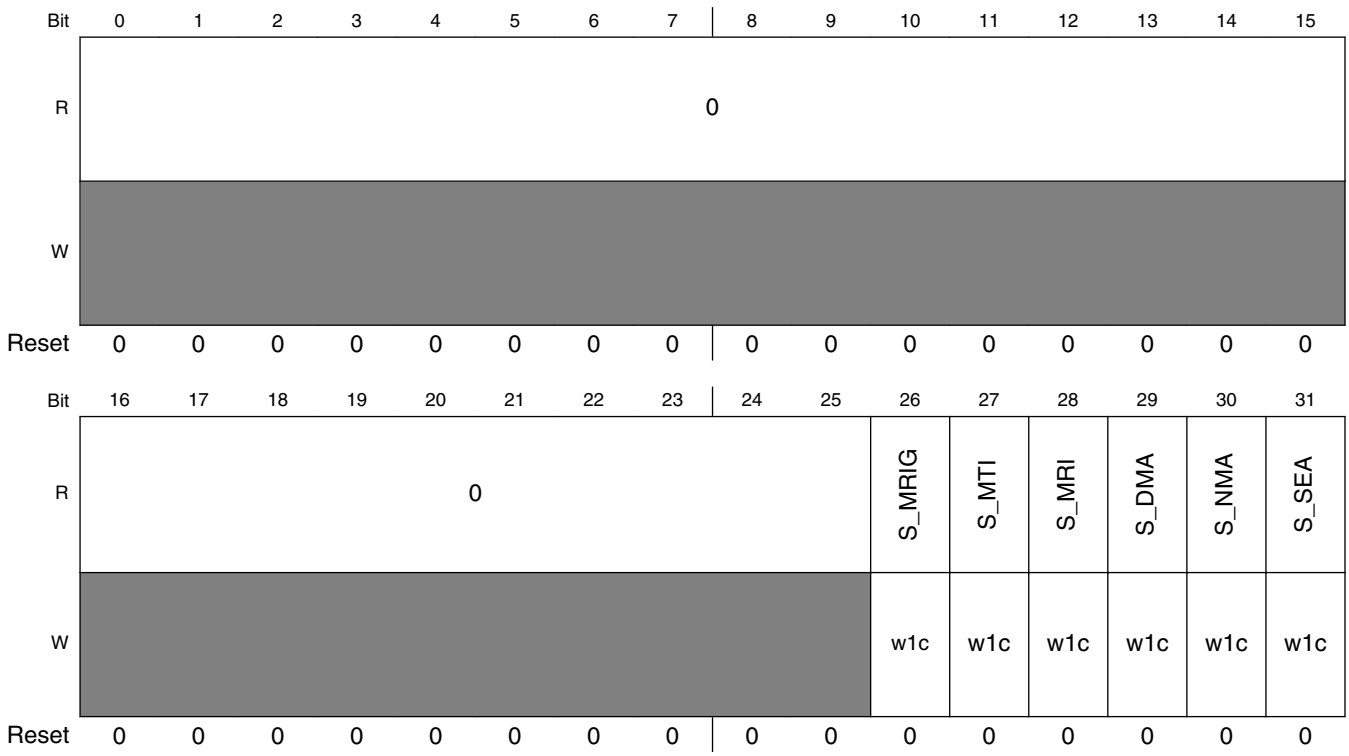
MC_ME_IM field descriptions (continued)

Field	Description
26 M_ICONF_CC	Invalid mode configuration (core configuration) interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
27 M_ICONF_CU	Invalid mode configuration (clock usage) interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
28 M_ICONF	Invalid mode configuration interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
29 M_IMODE	Invalid mode interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
30 M_SAFE	SAFE mode interrupt mask 0 SAFE mode interrupt is masked 1 SAFE mode interrupt is enabled
31 M_MTC	Mode transition complete interrupt mask 0 Mode transition complete interrupt is masked 1 Mode transition complete interrupt is enabled

62.3.6 Invalid Mode Transition Status (MC_ME_IMTS)

This register provides the status bits for the possible causes of an invalid mode interrupt.

Address: 0h base + 14h offset = 14h



MC_ME_IMTS field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 S_MRIG	Mode Request Ignored status - This bit is set whenever a new mode is requested while a transition to the SAFE mode is in progress. This bit is also set when a transition to STOP0/HALT0 mode is made when a wakeup is active. It is cleared by writing a '1' to this bit. 0 Mode transition requested is not illegal 1 Mode transition requested is illegal
27 S_MTI	Mode Transition Illegal status - This bit is set whenever a new mode is requested while some other non-SAFE mode transition process is active (S_MTRANS is '1' . Please refer to Section "Mode Transition Interrupts" for the exceptions to this behavior. It is cleared by writing a '1' to this bit. 0 Mode transition requested is not illegal 1 Mode transition requested is illegal
28 S_MRI	Mode Request Illegal status - This bit is set whenever the target mode requested is not a valid mode with respect to current mode. It is cleared by writing a '1' to this bit. 0 Target mode requested is not illegal with respect to current mode 1 Target mode requested is illegal with respect to current mode

Table continues on the next page...

MC_ME_IMTS field descriptions (continued)

Field	Description
29 S_DMA	Disabled Mode Access status - This bit is set whenever the target mode requested is one of those disabled modes determined by ME_ME register. It is cleared by writing a '1' to this bit. 0 Target mode requested is not a disabled mode 1 Target mode requested is a disabled mode
30 S_NMA	Non-existing Mode Access status - This bit is set whenever the target mode requested is one of those non existing modes determined by ME_ME register. It is cleared by writing a '1' to this bit. 0 Target mode requested is an existing mode 1 Target mode requested is a non-existing mode
31 S_SEA	SAFE Event Active status - This bit is set whenever the chip is in SAFE mode, SAFE event bit is pending and a new mode requested other than RESET/SAFE modes. It is cleared by writing a '1' to this bit. 0 No new mode requested other than RESET/SAFE while SAFE event is pending 1 New mode requested other than RESET/SAFE while SAFE event is pending

62.3.7 Debug Mode Transition Status (MC_ME_DMTS)

This register provides the status of different factors which influence mode transitions. It is used to give an indication of why a mode transition indicated by ME_GS[S_MTRANS] may be taking longer than expected.

NOTE

The ME_DMTS register does not indicate whether a mode transition is ongoing. Therefore, some ME_DMTS bits may still be asserted after the mode transition has completed.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PREVIOUS_MODE				0				MPH_BUSY	0		PMC_PROG	DBG_MODE	CCKL_PROG	PCS_PROG	SMR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDP_PRPH_0_255	VREG_CSRC_SC	CSRC_CSRC_SC	IRC_SC	SCSRC_SC	SYSCLK_SW	0	FLASH_SC	CDP_PRPH_244_255	CDP_PRPH_192_223	CDP_PRPH_160_191	CDP_PRPH_128_159	CDP_PRPH_96_127	CDP_PRPH_64_95	CDP_PRPH_32_63	CDP_PRPH_0_31
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_DMTS field descriptions

Field	Description
0–3 PREVIOUS_ MODE	Previous chip mode - These bits show the mode in which the chip was prior to the latest change to the current mode. 0000 RESET 0001 TEST 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 HALT0 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 reserved
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 MPH_BUSY	MC_ME/MC_PCU Handshake Busy indicator - This bit is set if the MC_ME has requested a mode change from the MC_PCU and the MC_PCU has not yet completed its power-up/down sequencing. It is cleared when the MC_PCU has completed power-up/down sequencing. 0 Handshake is not busy 1 Handshake is busy
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 PMC_PROG	MC_PCU Mode Change in Progress indicator - This bit is set if the MC_PCU is in the process of powering up or down power domains. It is cleared when all power-up/down processes have completed. 0 Power-up/down transition is not in progress 1 Power-up/down transition is in progress
12 DBG_MODE	Debug mode indicator - This bit is set while the chip is in debug mode. 0 The chip is not in debug mode 1 The chip is in debug mode
13 CCKL_PROG	Core Clock Enable/Disable in Progress - This bit is set while any core's clock is in the process of being enabled or disabled. 0 No core clock is being enabled or disabled 1 A core clock is being enabled or disabled
14 PCS_PROG	Progressive System Clock Switching in Progress - This bit is set when PCS is in progress, peripheral clock switching is ongoing, core clock switching is ongoing, or power sequence process is ongoing.

Table continues on the next page...

MC_ME_DMTS field descriptions (continued)

Field	Description
	0 PCS is not in progress 1 PCS is in progress
15 SMR	SAFE mode request from MC_RGM is active indicator - This bit is set if a hardware SAFE mode request has been triggered. It is cleared when the hardware SAFE mode request has been cleared. 0 A SAFE mode request is not active 1 A SAFE mode request is active
16 CDP_PRPH_0_255	Clock Disable Process Pending status for Peripherals 0-255 - This bit is set when any peripheral appearing has been requested to have its clock disabled. It is cleared when all the peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
17 VREG_CSRC_SC	Main VREG dependent Clock Source State Change during mode transition indicator - This bit is set when a clock source which depends on the main voltage regulator to be powered-up is requested to change its power up/down state. It is cleared when the clock source has completed its state change. 0 No state change is taking place 1 A state change is taking place
18 CSRC_CSRC_SC	(Other) Clock Source dependent Clock Source State Change during mode transition indicator - This bit is set when a clock source which depends on another clock source to be powered-up is requested to change its power up/down state. It is cleared when the clock source has completed its state change. 0 No state change is taking place 1 A state change is taking place
19 IRC_SC	IRC State Change during mode transition indicator - This bit is set when the 16 MHz internal RC oscillator is requested to change its power up/down state. It is cleared when the 16 MHz internal RC oscillator has completed its state change. 0 No state change is taking place 1 A state change is taking place
20 SCSRC_SC	Secondary Clock Sources State Change during mode transition indicator - This bit is set when a secondary clock source is requested to change its power up/down state. It is cleared when all secondary system clock sources have completed their state changes. (A secondary clock source is a clock source other than IRC.) 0 No state change is taking place 1 A state change is taking place
21 SYSCLK_SW	System Clock Switching pending status 0 No system clock source switching is pending 1 A system clock source switching is pending
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 FLASH_SC	FLASH State Change during mode transition indicator - This bit is set when the FLASH is requested to change its power up/down state. It is cleared when the DFLASH has completed its state change. 0 No state change is taking place 1 A state change is taking place

Table continues on the next page...

MC_ME_DMTS field descriptions (continued)

Field	Description
24 CDP_PRPH_ 244_255	Clock Disable Process Pending status for Peripherals 224 - 2552 - This bit is set when any peripheral appearing in ME_PS7 has been requested to have its clock disabled. It is cleared when all the peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
25 CDP_PRPH_ 192_223	Clock Disable Process Pending status for Peripherals 192 - 223 - This bit is set when any peripheral appearing in ME_PS6 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 1 Clock disabling is pending for at least one peripheral
26 CDP_PRPH_ 160_191	Clock Disable Process Pending status for Peripherals 160 - 191 - This bit is set when any peripheral appearing in ME_PS5 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 1 Clock disabling is pending for at least one peripheral
27 CDP_PRPH_ 128_159	Clock Disable Process Pending status for Peripherals 128 - 159 - This bit is set when any peripheral appearing in ME_PS4 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
28 CDP_PRPH_96_ 127	Clock Disable Process Pending status for Peripherals 96 - 127 - This bit is set when any peripheral appearing in ME_PS3 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
29 CDP_PRPH_64_ 95	Clock Disable Process Pending status for Peripherals 64 - 95 - This bit is set when any peripheral appearing in ME_PS2 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
30 CDP_PRPH_32_ 63	Clock Disable Process Pending status for Peripherals 32 - 63 - This bit is set when any peripheral appearing in ME_PS1 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
31 CDP_PRPH_0_ 31	Clock Disable Process Pending status for Peripherals 0 - 31 - This bit is set when any peripheral appearing in ME_PS0 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending

Table continues on the next page...

MC_ME_DMTS field descriptions (continued)

Field	Description
	1 Clock disabling is pending for at least one peripheral

62.3.8 RESET Mode Configuration (MC_ME_RESET_MC)

This register configures system behavior during RESET mode.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								PLL1ON	PLL0ON	XOSCON	IRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

MC_ME_RESET_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_RESET_MC field descriptions (continued)

Field	Description
1–3 PWRLVL	Power level - These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control - This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE/TEST modes, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control - This bit specifies whether main voltage regulator is switched off or not while entering this mode.
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control - This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 reserved 10 reserved 11 Flash is in normal mode
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	secondary PLL control 0 secondary PLL is switched off 1 secondary PLL is switched on
25 PLL0ON	system PLL control 0 system PLL is switched off 1 system PLL is switched on
26 XOSCON	4-40 MHz crystal oscillator control 0 4-40 MHz crystal oscillator is switched off 1 4-40 MHz crystal oscillator is switched on
27 IRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control - These bits specify the system clock to be used by the system. 0000 16 MHz int. RC osc. 0001 external osc. 0010 PLL0 0011 reserved

Table continues on the next page...

MC_ME_RESET_MC field descriptions (continued)

Field	Description
	0100 PLL1
	0101 reserved
	0110 reserved
	0111 reserved
	1000 reserved
	1001 reserved
	1010 reserved
	1011 reserved
	1100 reserved
	1101 reserved
	1110 reserved
	1111 system clock is disabled in TEST mode, reserved in all other modes

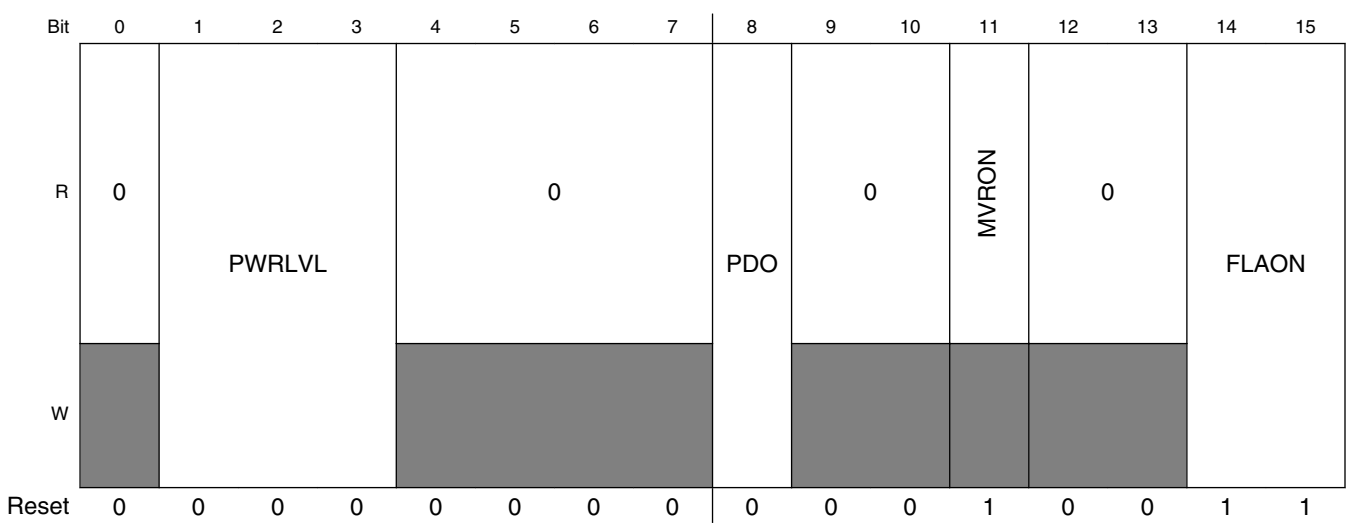
62.3.9 TEST Mode Configuration (MC_ME_TEST_MC)

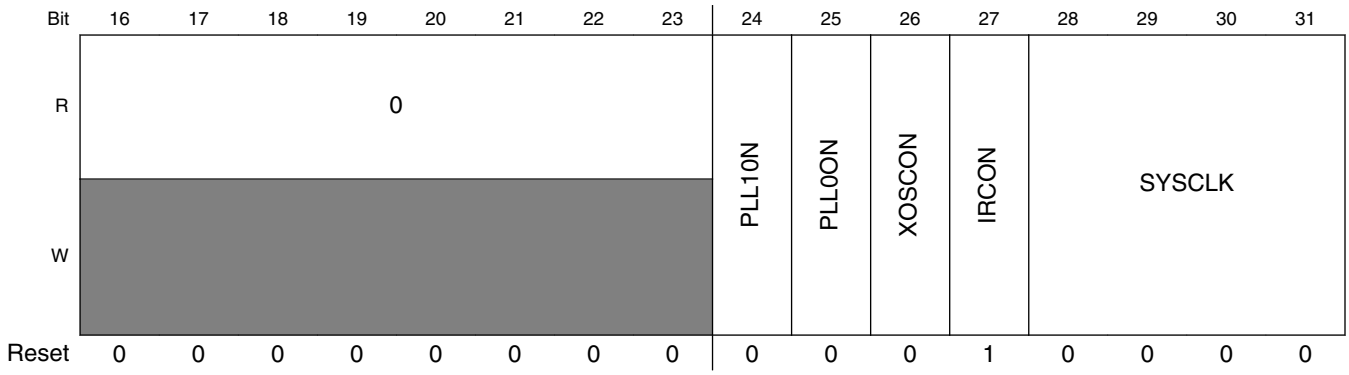
This register configures system behavior during TEST mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 24h offset = 24h





MC_ME_TEST_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	See Description for the ME_RESET_MC register.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	See Description for the ME_RESET_MC register.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	See Description for the ME_RESET_MC register.
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	See Description for the ME_RESET_MC register.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL10N	See Description for the ME_RESET_MC register.
25 PLL0ON	See Description for the ME_RESET_MC register.
26 XOSCON	See Description for the ME_RESET_MC register.
27 IRCON	See Description for the ME_RESET_MC register.
28–31 SYSCLK	See Description for the ME_RESET_MC register.

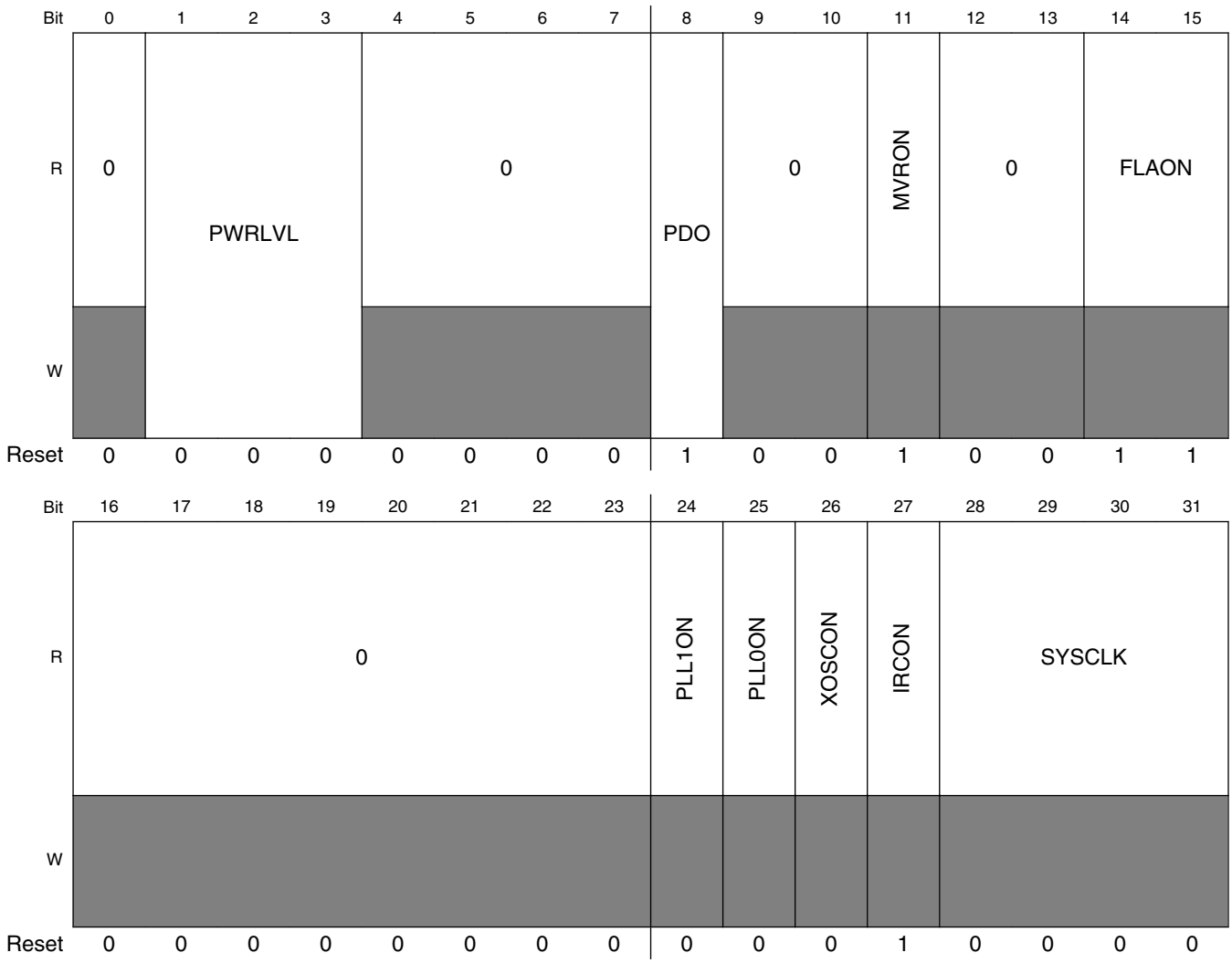
62.3.10 SAFE Mode Configuration (MC_ME_SAFE_MC)

This register configures system behavior during SAFE mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 28h offset = 28h



MC_ME_SAFE_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	See Description for the ME_RESET_MC register.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	See Description for the ME_RESET_MC register.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_SAFE_MC field descriptions (continued)

Field	Description
11 MVRON	See Description for the ME_RESET_MC register.
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	See Description for the ME_RESET_MC register.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	See Description for the ME_RESET_MC register.
25 PLL0ON	See Description for the ME_RESET_MC register.
26 XOSCON	See Description for the ME_RESET_MC register.
27 IRCON	See Description for the ME_RESET_MC register.
28–31 SYSCLK	See Description for the ME_RESET_MC register.

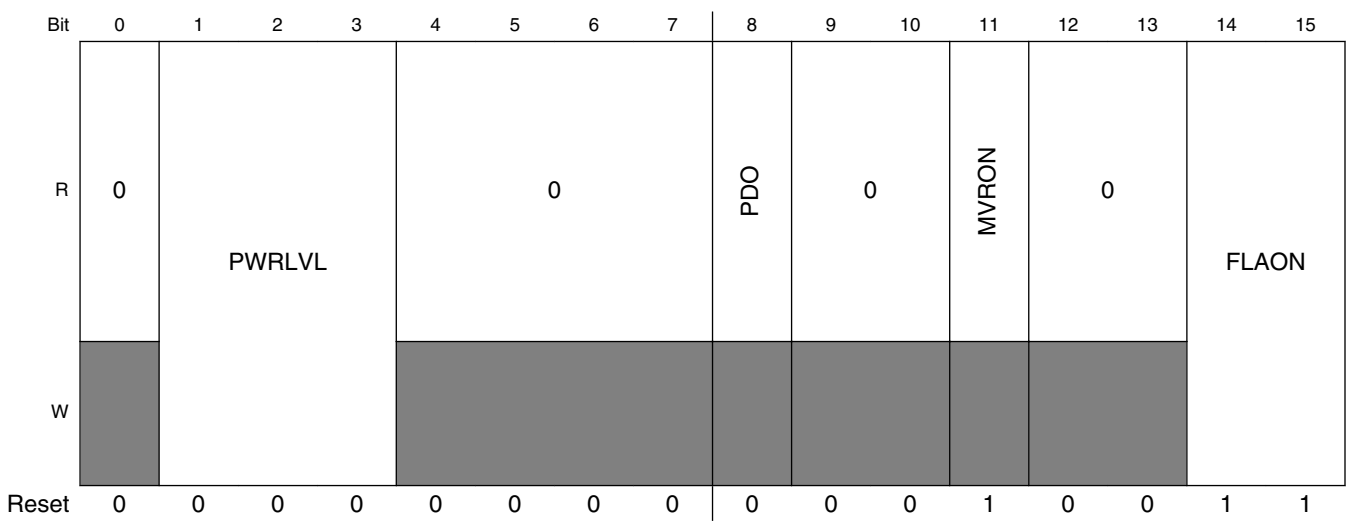
62.3.11 DRUN Mode Configuration (MC_ME_DRUN_MC)

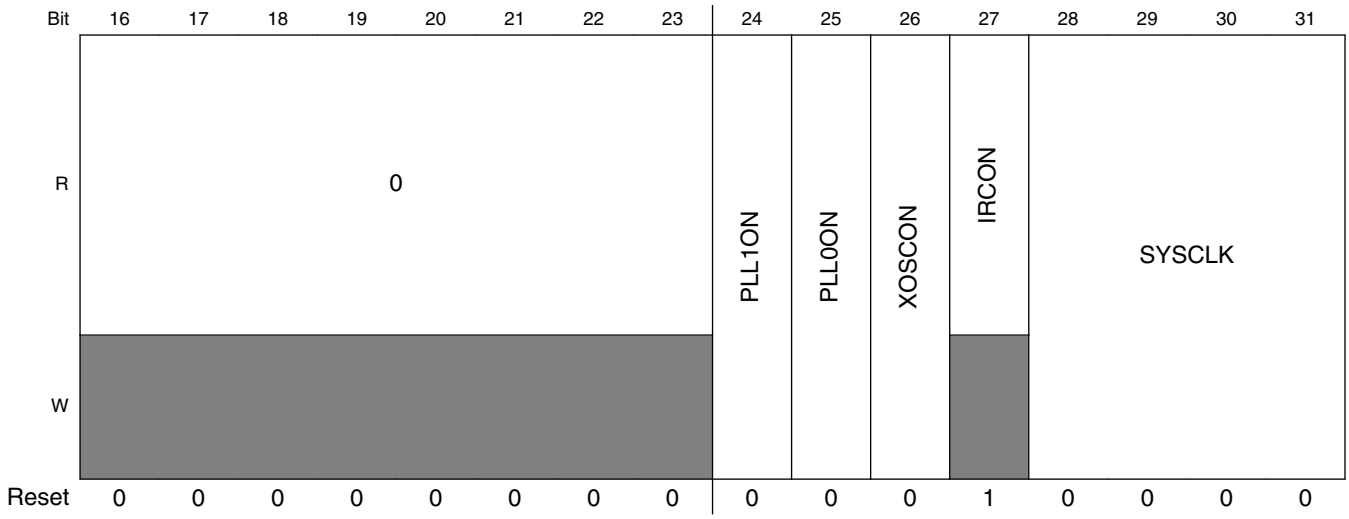
This register configures system behavior during DRUN mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 2Ch offset = 2Ch





MC_ME_DRUN_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	See Description for the ME_RESET_MC register.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	See Description for the ME_RESET_MC register.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	See Description for the ME_RESET_MC register.
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	See Description for the ME_RESET_MC register.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	See Description for the ME_RESET_MC register.
25 PLL0ON	See Description for the ME_RESET_MC register.
26 XOSCON	See Description for the ME_RESET_MC register.
27 IRCON	See Description for the ME_RESET_MC register.
28–31 SYSCLK	See Description for the ME_RESET_MC register.

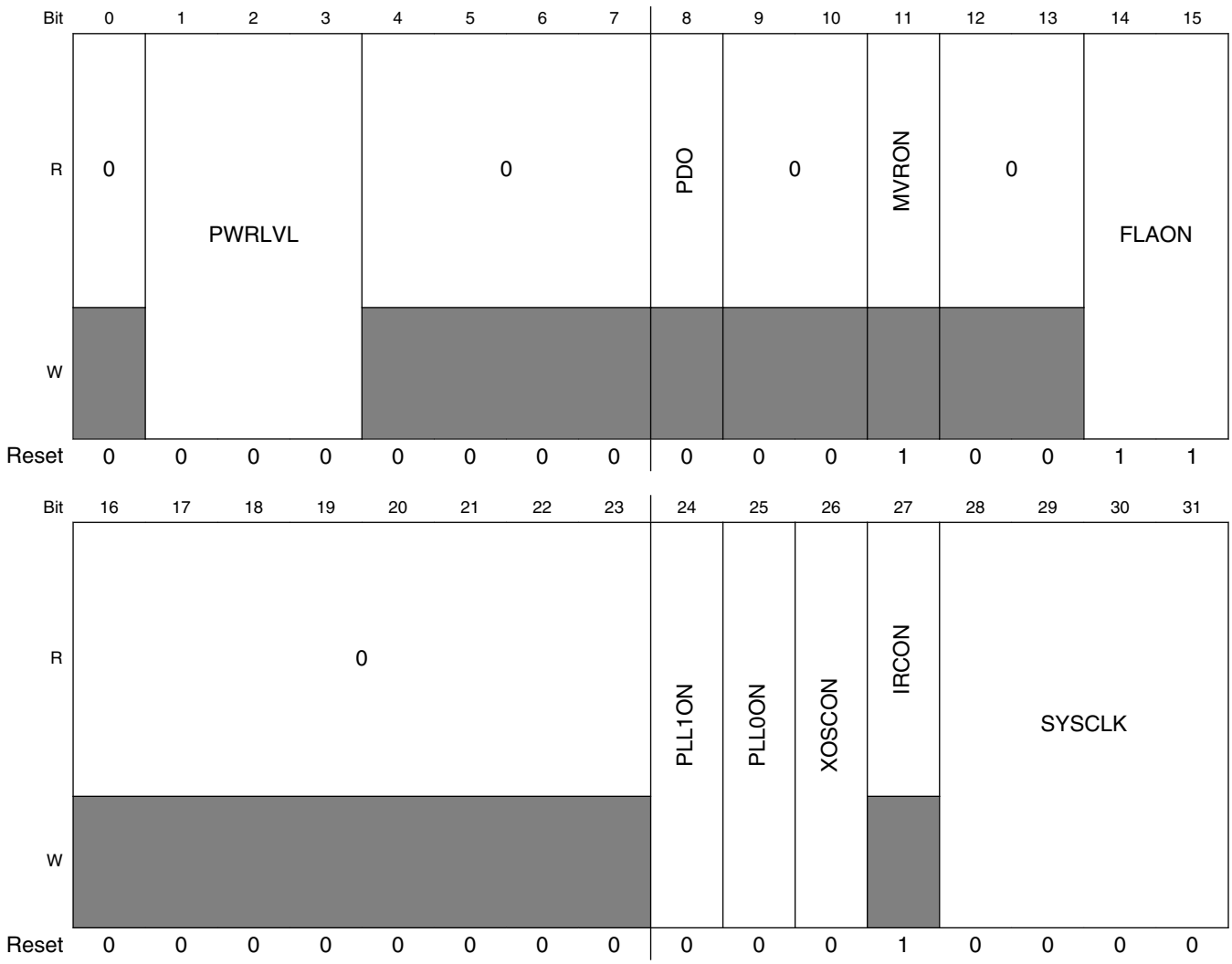
62.3.12 RUN0 Mode Configuration (MC_ME_RUN0_MC)

This register configures system behavior during RUN0 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 30h offset = 30h



MC_ME_RUN0_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	See Description for the ME_RESET_MC register.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_RUN0_MC field descriptions (continued)

Field	Description
8 PDO	See Description for the ME_RESET_MC register.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	See Description for the ME_RESET_MC register.
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	See Description for the ME_RESET_MC register.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	See Description for the ME_RESET_MC register.
25 PLL0ON	See Description for the ME_RESET_MC register.
26 XOSCON	See Description for the ME_RESET_MC register.
27 IRCON	See Description for the ME_RESET_MC register.
28–31 SYSCLK	See Description for the ME_RESET_MC register.

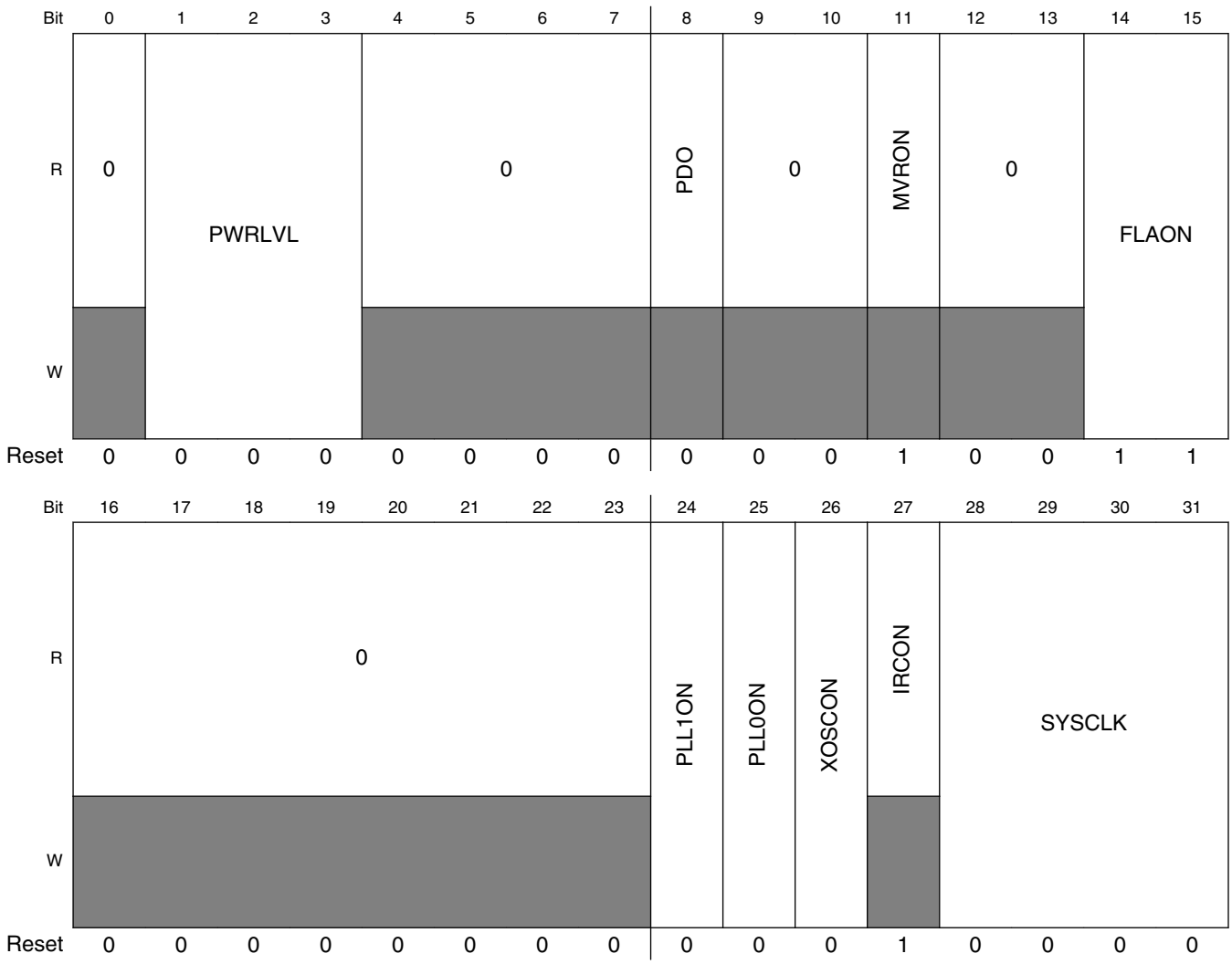
62.3.13 RUN1 Mode Configuration (MC_ME_RUN1_MC)

This register configures system behavior during RUN1 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 34h offset = 34h



MC_ME_RUN1_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	See Description for the ME_RESET_MC register.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	See Description for the ME_RESET_MC register.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	See Description for the ME_RESET_MC register.
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_RUN1_MC field descriptions (continued)

Field	Description
14–15 FLAON	See Description for the ME_RESET_MC register.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	See Description for the ME_RESET_MC register.
25 PLL0ON	See Description for the ME_RESET_MC register.
26 XOSCON	See Description for the ME_RESET_MC register.
27 IRCON	See Description for the ME_RESET_MC register.
28–31 SYSCLK	See Description for the ME_RESET_MC register.

62.3.14 RUN2 Mode Configuration (MC_ME_RUN2_MC)

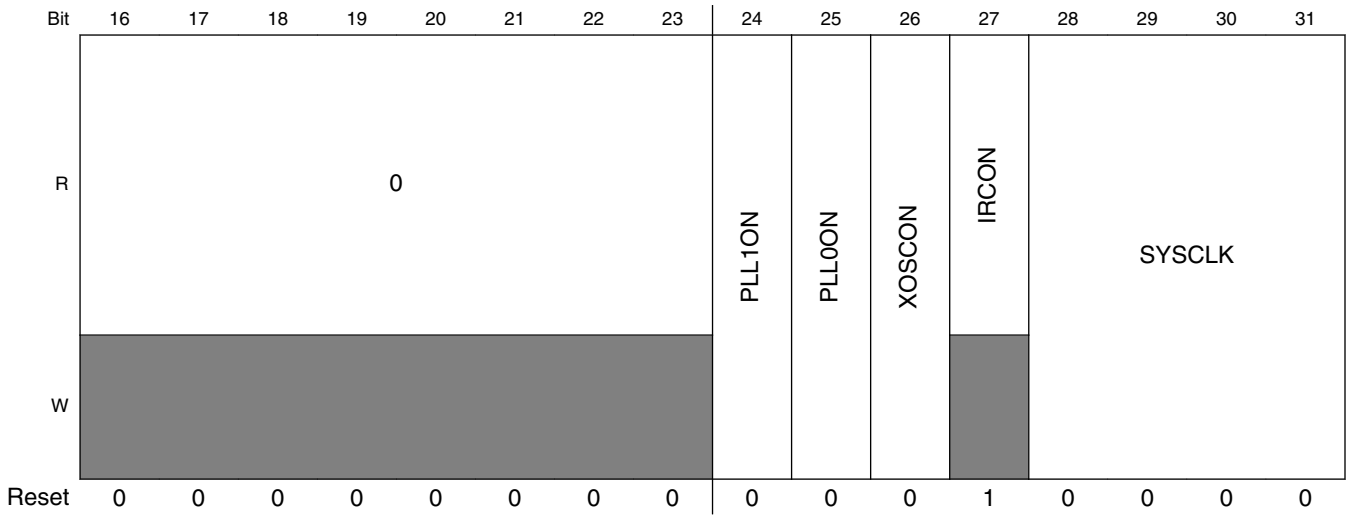
This register configures system behavior during RUN2 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1



MC_ME_RUN2_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	See Description for the ME_RESET_MC register.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	See Description for the ME_RESET_MC register.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	See Description for the ME_RESET_MC register.
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	See Description for the ME_RESET_MC register.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	See Description for the ME_RESET_MC register.
25 PLL0ON	See Description for the ME_RESET_MC register.
26 XOSCON	See Description for the ME_RESET_MC register.
27 IRCON	See Description for the ME_RESET_MC register.
28–31 SYSCLK	See Description for the ME_RESET_MC register.See Description for the ME_RESET_MC register.

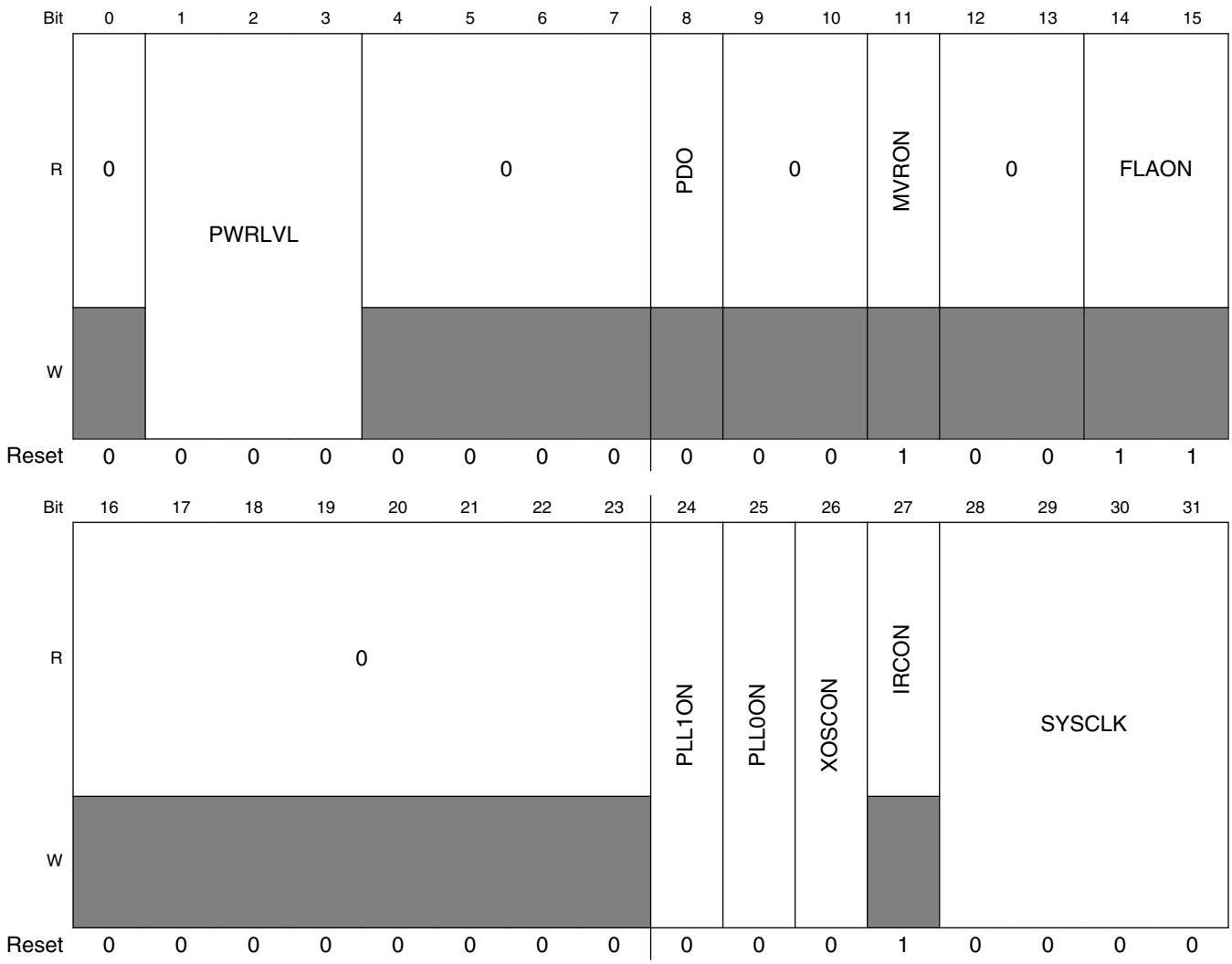
62.3.15 RUN3 Mode Configuration (MC_ME_RUN3_MC)

This register configures system behavior during RUN3 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 3Ch offset = 3Ch



MC_ME_RUN3_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	See Description for the ME_RESET_MC register.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_RUN3_MC field descriptions (continued)

Field	Description
8 PDO	See Description for the ME_RESET_MC register.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	See Description for the ME_RESET_MC register.
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	See Description for the ME_RESET_MC register.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	See Description for the ME_RESET_MC register.
25 PLL0ON	See Description for the ME_RESET_MC register.
26 XOSCON	See Description for the ME_RESET_MC register.
27 IRCON	See Description for the ME_RESET_MC register.
28–31 SYSCLK	See Description for the ME_RESET_MC register.

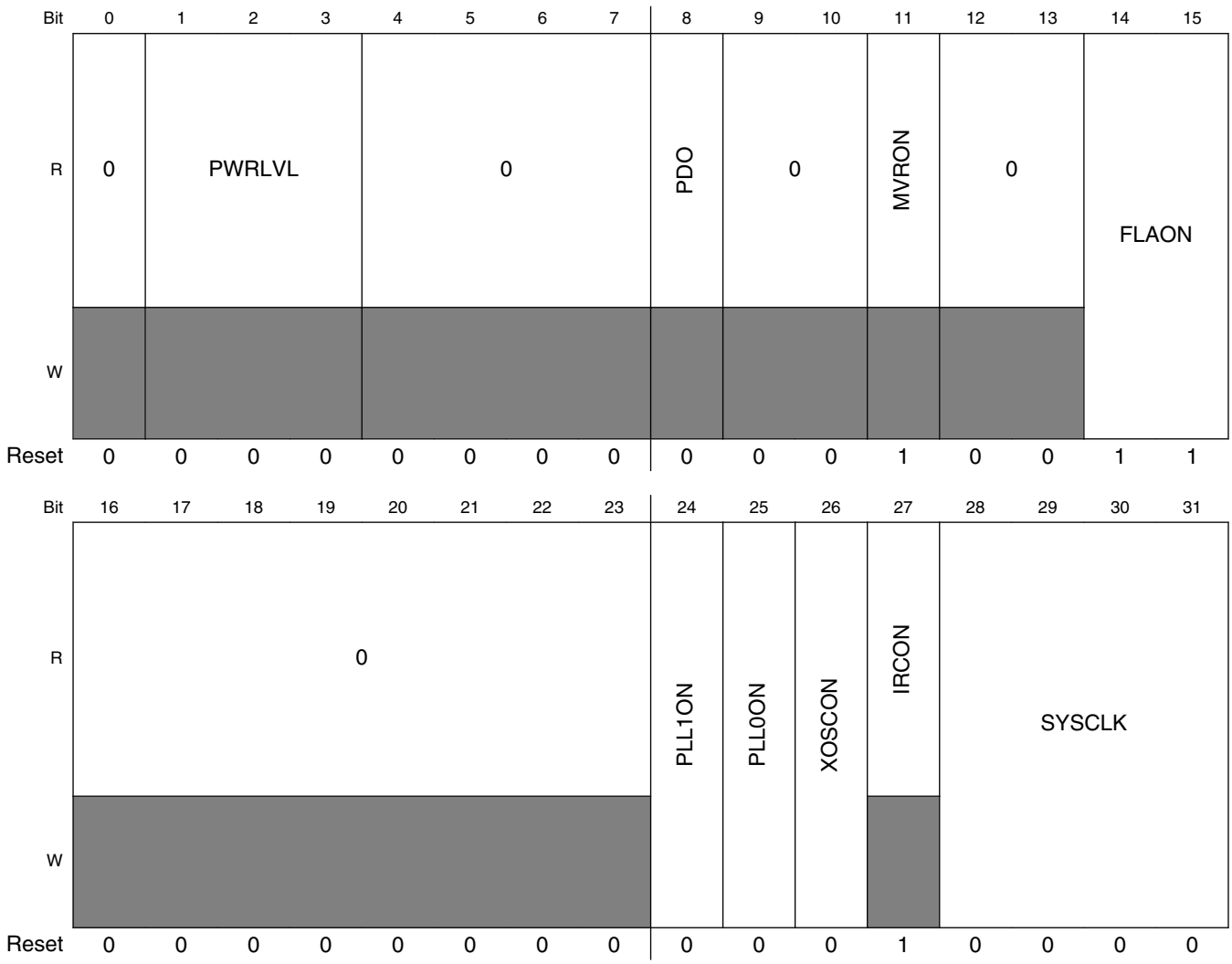
62.3.16 HALT0 Mode Configuration (MC_ME_HALT0_MC)

This register configures system behavior during HALT0 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 40h offset = 40h



MC_ME_HALT0_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	See Description for the ME_RESET_MC register.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	See Description for the ME_RESET_MC register.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	See Description for the ME_RESET_MC register.
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_HALT0_MC field descriptions (continued)

Field	Description
14–15 FLAON	See Description for the ME_RESET_MC register.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	See Description for the ME_RESET_MC register.
25 PLL0ON	See Description for the ME_RESET_MC register.
26 XOSCON	See Description for the ME_RESET_MC register.
27 IRCON	See Description for the ME_RESET_MC register.
28–31 SYSCLK	See Description for the ME_RESET_MC register.

62.3.17 STOP0 Mode Configuration (MC_ME_STOP0_MC)

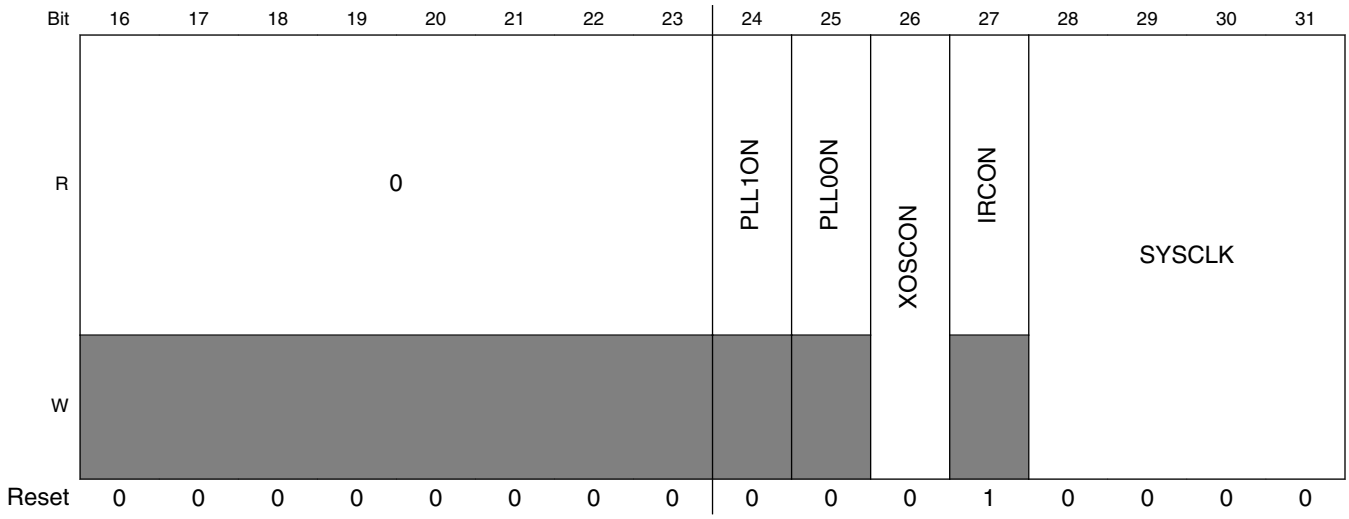
This register configures system behavior during STOP0 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1



MC_ME_STOP0_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	See Description for the ME_RESET_MC register.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	See Description for the ME_RESET_MC register.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	See Description for the ME_RESET_MC register.
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	See Description for the ME_RESET_MC register.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PLL1ON	See Description for the ME_RESET_MC register.
25 PLL0ON	See Description for the ME_RESET_MC register.
26 XOSCON	See Description for the ME_RESET_MC register.
27 IRCON	See Description for the ME_RESET_MC register.
28–31 SYSCLK	See Description for the ME_RESET_MC register.

62.3.18 Peripheral Status 0 (MC_ME_PS0)

This register provides the status of the peripherals.

Address: 0h base + 60h offset = 60h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_PIT_RTI_1	S_PIT_RTI_0	S_DECFILT_0	S_DECFILT_1	reserved											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	S_SIUL2_0	0			S_SIPI_0	0	S_LFAST_0	0								
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

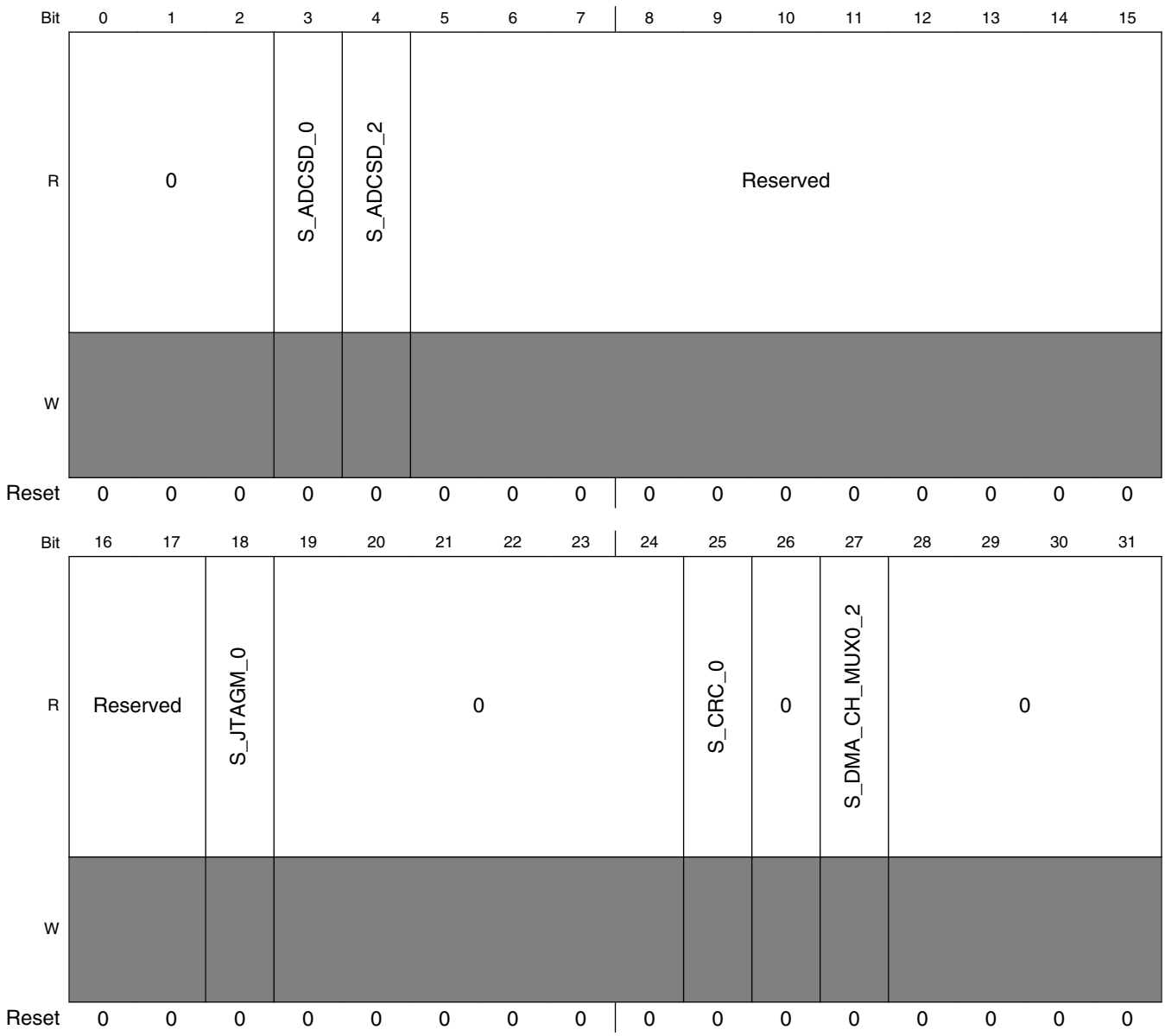
MC_ME_PS0 field descriptions

Field	Description
0 S_PIT_RTI_1	Peripheral status - These bits specify the current status of each peripheral which is controlled by the MC_ME 0 Peripheral is frozen 1 Peripheral is active
1 S_PIT_RTI_0	(See the Description for the S_PIT_RTI_1 bit.)
2 S_DECFILT_0	(See the Description for the S_PIT_RTI_1 bit.)
3 S_DECFILT_1	(See the Description for the S_PIT_RTI_1 bit.)
4–15 reserved	This field is reserved.
16 S_SIUL2_0	SIUL2 is always on.
17–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 S_SIPI_0	(See the Description for the S_PIT_RTI_1 bit.)
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 S_LFAST_0	(See the Description for the S_PIT_RTI_1 bit.)
23–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

62.3.19 Peripheral Status 1 (MC_ME_PS1)

This register provides the status of the peripherals.

Address: 0h base + 64h offset = 64h



MC_ME_PS1 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 S_ADCSD_0	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)

Table continues on the next page...

MC_ME_PS1 field descriptions (continued)

Field	Description
4 S_ADCSD_2	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
5–17 Reserved	This field is reserved.
18 S_JTAGM_0	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
19–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 S_CRC_0	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 S_DMA_CH_ MUX0_2	(See the Description for the ME_PS0[S_PIT_RTI_1] register.) This controls only DMAMUX0, 1, and 2.
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

62.3.20 Peripheral Status 2 (MC_ME_PS2)

This register provides the status of the peripherals.

Address: 0h base + 68h offset = 68h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0			S_LINFLEX_0	S_LINFLEX_2	0					S_LINFLEX_M0	0	S_FLEXCAN_0	0	S_FLEXCAN_2	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								S_ADCSAR_0	0	S_ADCSAR_2	0				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

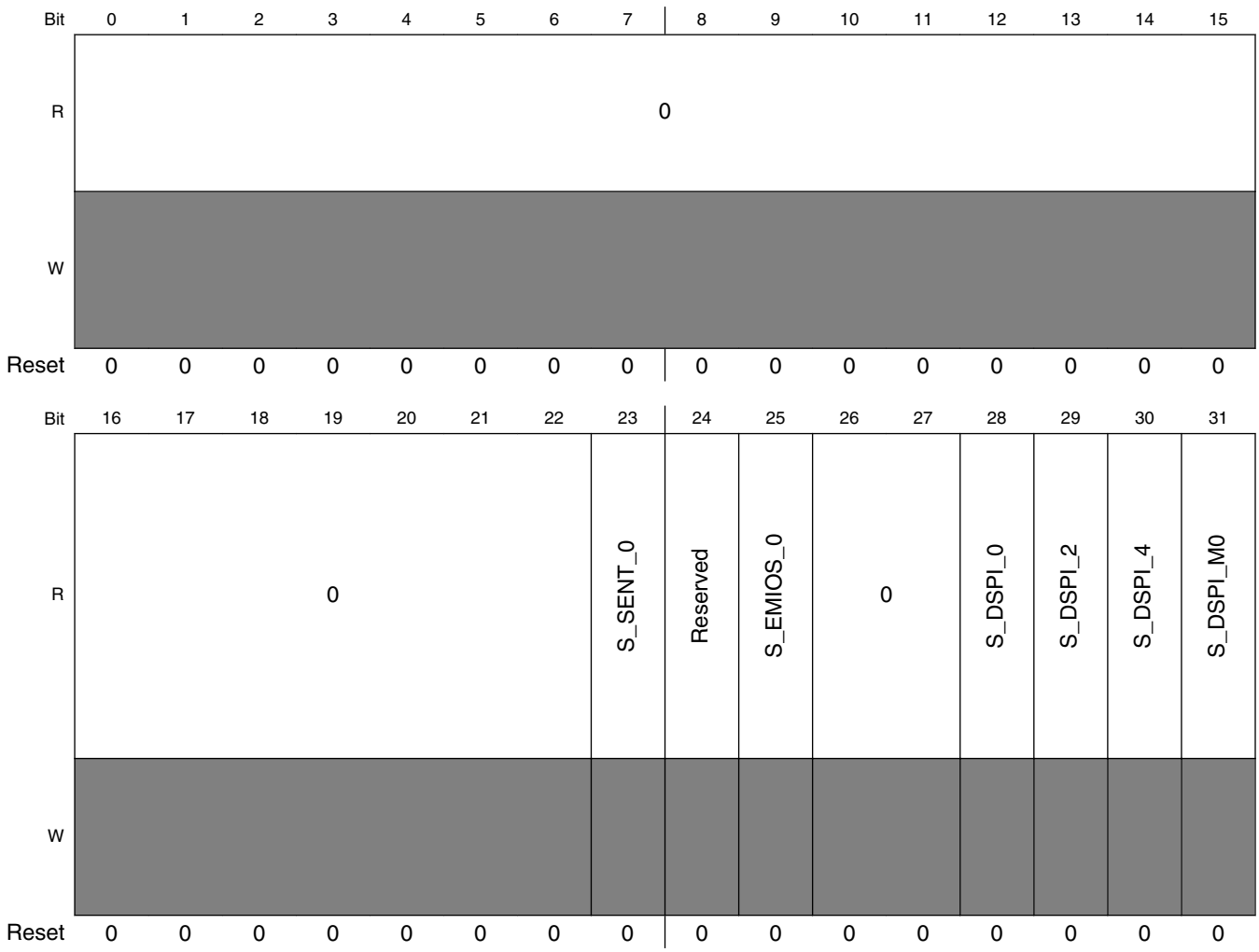
MC_ME_PS2 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 S_LINFLEX_0	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
4 S_LINFLEX_2	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
5–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 S_LINFLEX_M0	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 S_FLEXCAN_0	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 S_FLEXCAN_2	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
15–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 S_ADCSAR_0	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 S_ADCSAR_2	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
27–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

62.3.21 Peripheral Status 3 (MC_ME_PS3)

This register provides the status of the peripherals.

Address: 0h base + 6Ch offset = 6Ch



MC_ME_PS3 field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 S_SENT_0	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
24 Reserved	This field is reserved.
25 S_EMIOS_0	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)

Table continues on the next page...

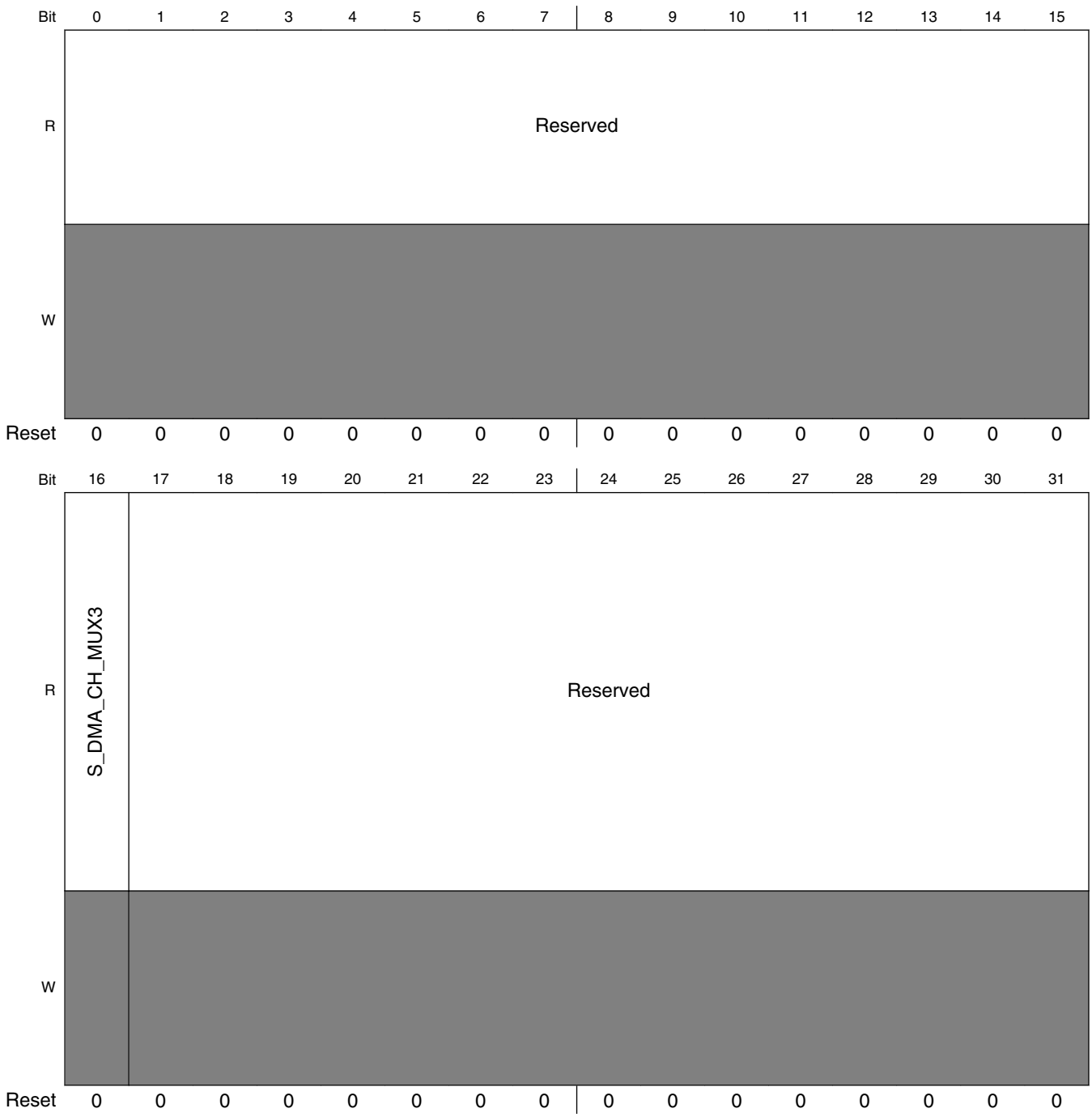
MC_ME_PS3 field descriptions (continued)

Field	Description
26–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 S_DSPI_0	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
29 S_DSPI_2	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
30 S_DSPI_4	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
31 S_DSPI_M0	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)

62.3.22 Peripheral Status 4 (MC_ME_PS4)

This register provides the status of the peripherals.

Address: 0h base + 70h offset = 70h



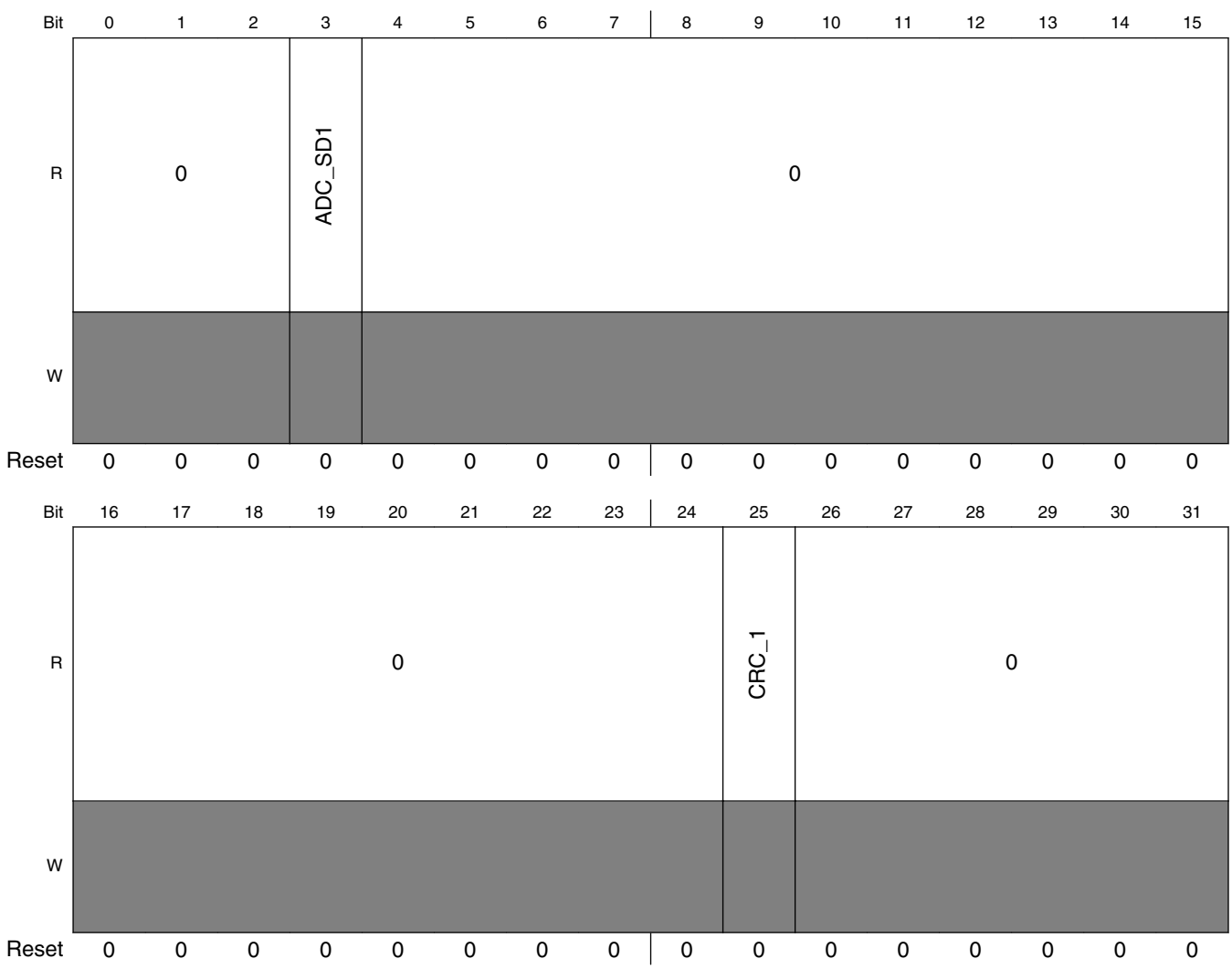
MC_ME_PS4 field descriptions

Field	Description
0–15 Reserved	This field is reserved.
16 S_DMA_CH_MUX3	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
17–31 Reserved	This field is reserved.

62.3.23 Peripheral Status 5 (MC_ME_PS5)

This register provides the status of the peripherals.

Address: 0h base + 74h offset = 74h



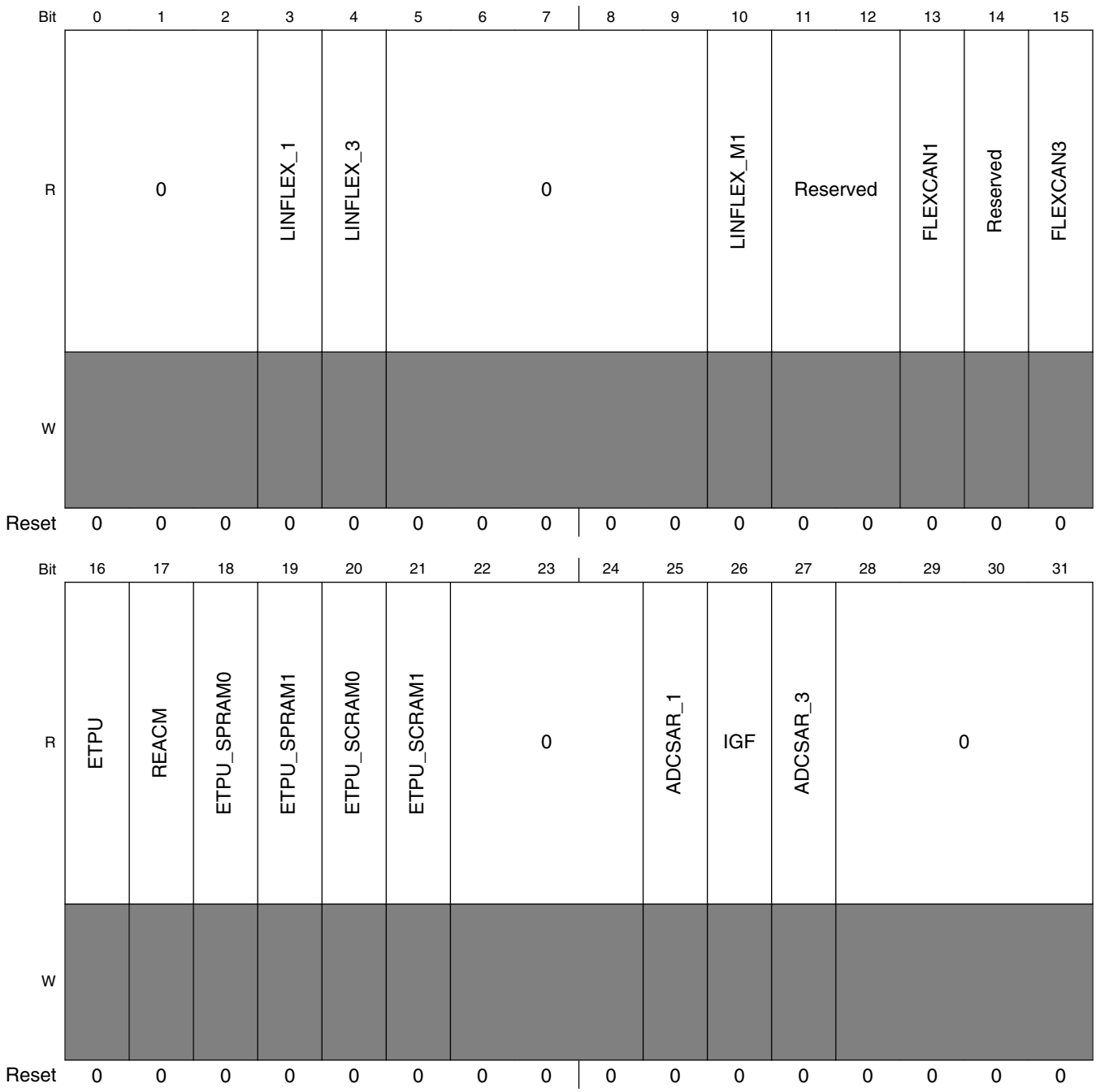
MC_ME_PS5 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 ADC_SD1	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
4–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 CRC_1	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
26–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

62.3.24 Peripheral Status 6 (MC_ME_PS6)

This register provides the status of the peripherals.

Address: 0h base + 78h offset = 78h



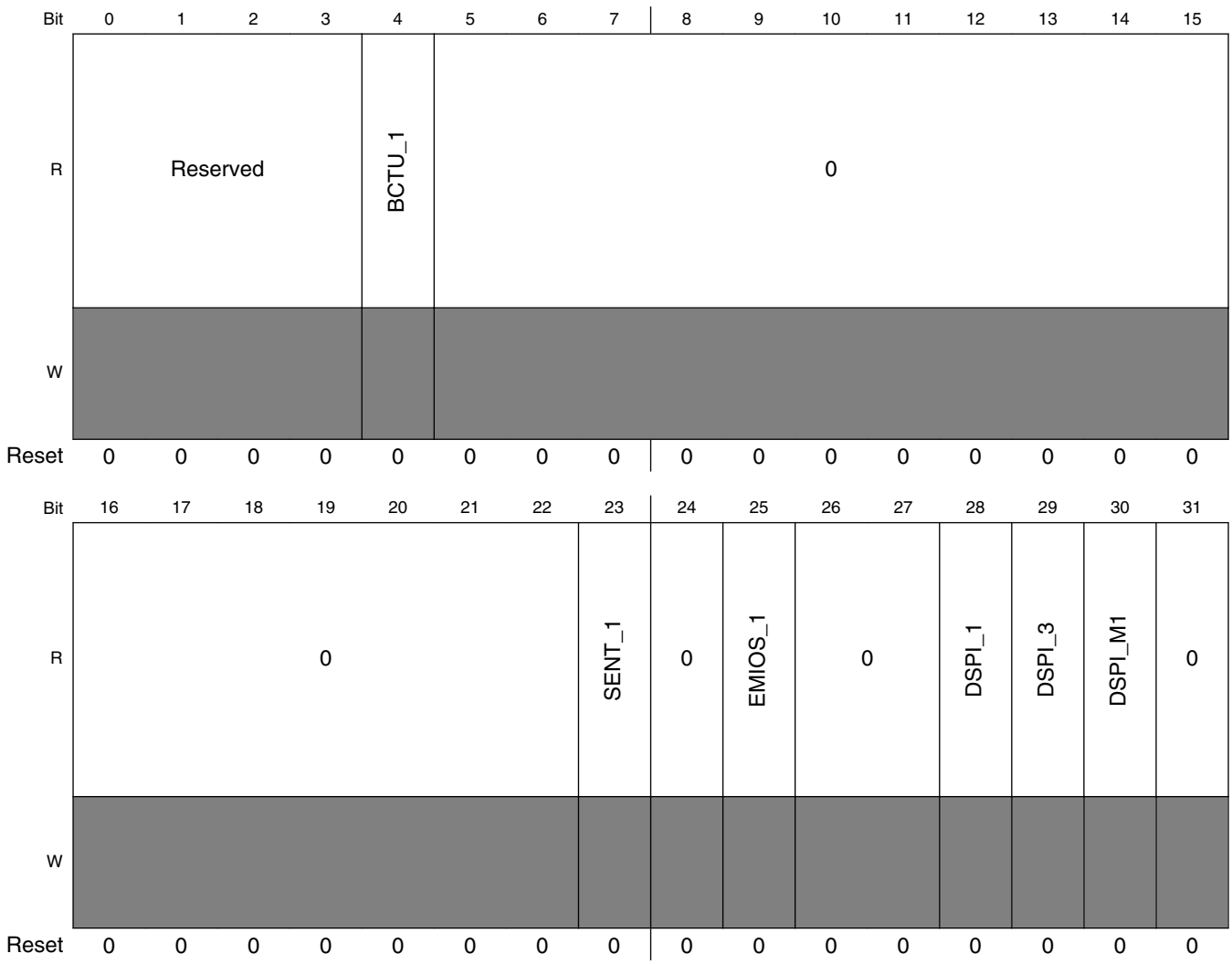
MC_ME_PS6 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 LINFLEX_1	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
4 LINFLEX_3	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
5–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 LINFLEX_M1	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
11–12 Reserved	This field is reserved.
13 FLEXCAN1	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
14 Reserved	This field is reserved.
15 FLEXCAN3	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
16 ETPU	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
17 REACM	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
18 ETPU_SPRAM0	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
19 ETPU_SPRAM1	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
20 ETPU_SCRAM0	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
21 ETPU_SCRAM1	(See the Description for the ME_PS0[S_PIT_RTI_1] register.
22–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 ADCSAR_1	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
26 IGF	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
27 ADCSAR_3	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

62.3.25 Peripheral Status 7 (MC_ME_PS7)

This register provides the status of the peripherals.

Address: 0h base + 7Ch offset = 7Ch



MC_ME_PS7 field descriptions

Field	Description
0–3 Reserved	This field is reserved.
4 BCTU_1	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
5–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SENT_1	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)

Table continues on the next page...

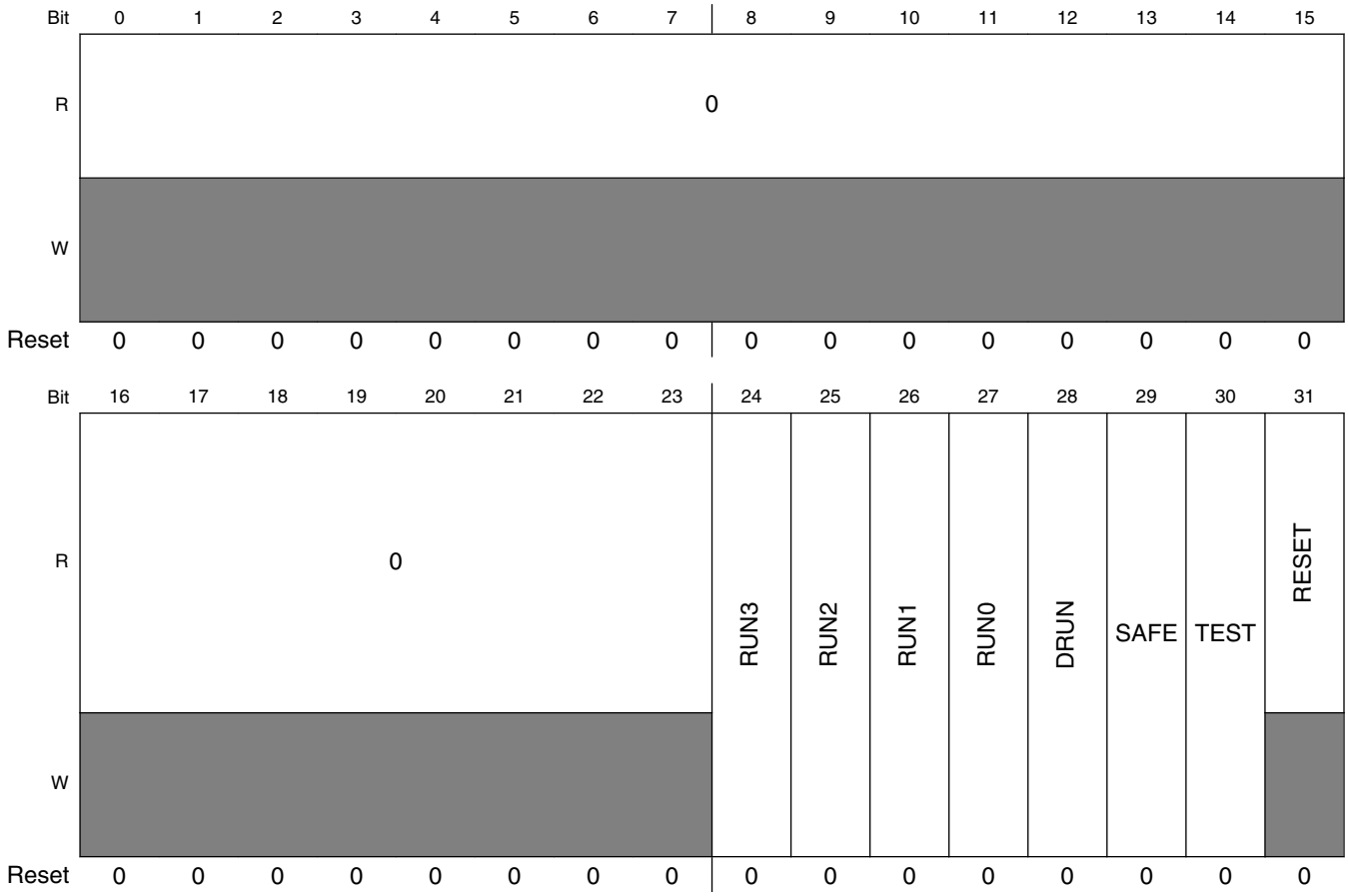
MC_ME_PS7 field descriptions (continued)

Field	Description
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 EMIOS_1	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
26–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 DSPI_1	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
29 DSPI_3	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
30 DSPI_M1	(See the Description for the ME_PS0[S_PIT_RTI_1] register.)
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

62.3.26 Run Peripheral Configuration 0 (MC_ME_RUN_PC*n*)

The RUN_PCn registers configure eight different types of peripheral behavior during run modes.

Address: 0h base + 80h offset + (4d × i), where i=0d to 7d



MC_ME_RUN_PC*n* field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 RUN3	Peripheral control during RUN3 0 Peripheral is frozen with clock gated 1 Peripheral is active
25 RUN2	Peripheral control during RUN2 0 Peripheral is frozen with clock gated 1 Peripheral is active
26 RUN1	Peripheral control during RUN1 0 Peripheral is frozen with clock gated

Table continues on the next page...

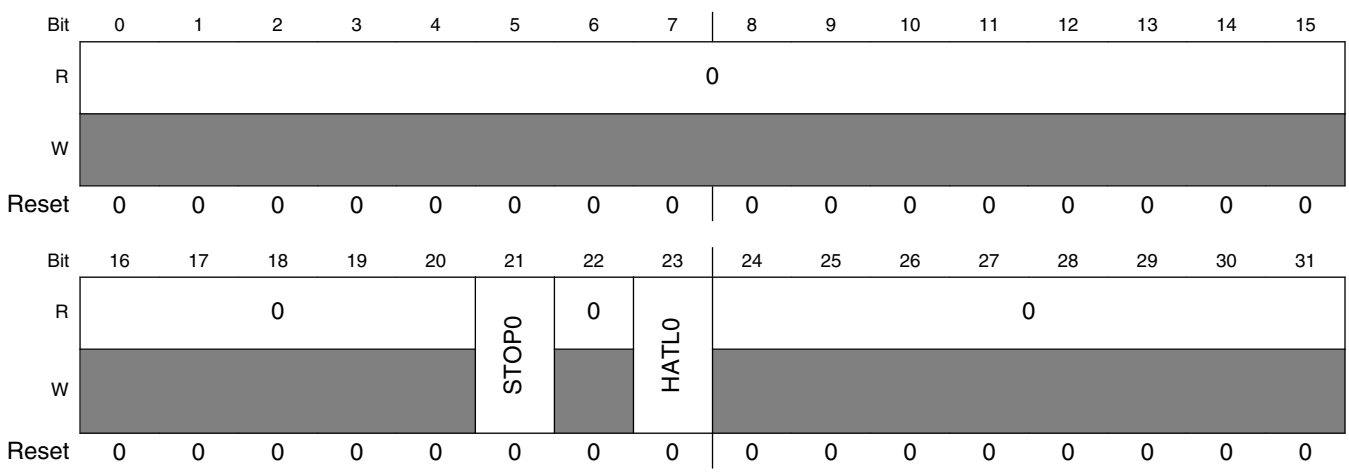
MC_ME_RUN_PC*n* field descriptions (continued)

Field	Description
	1 Peripheral is active
27 RUN0	Peripheral control during RUN0 0 Peripheral is frozen with clock gated 1 Peripheral is active
28 DRUN	Peripheral control during DRUN 0 Peripheral is frozen with clock gated 1 Peripheral is active
29 SAFE	Peripheral control during SAFE 0 Peripheral is frozen with clock gated 1 Peripheral is active
30 TEST	Peripheral control during TEST 0 Peripheral is frozen with clock gated 1 Peripheral is active
31 RESET	Peripheral control during RESET 0 Peripheral is frozen with clock gated 1 Peripheral is active

62.3.27 Low-Power Peripheral Configuration 0 (MC_ME_LP_PC*n*)

The LP_PCn registers configure eight different types of peripheral behavior during non-run modes.

Address: 0h base + A0h offset + (4d × i), where i=0d to 7d



MC_ME_LP_PC*n* field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 STOP0	Peripheral control during STOP0 0 Peripheral is frozen with clock gated 1 Peripheral is active
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 HATL0	Peripheral control during HALT0 0 Peripheral is frozen with clock gated 1 Peripheral is active
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

62.3.28 LFAST_0 Control (MC_ME_PCTL9)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + C9h offset = C9h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL9 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode - This bit controls the state of the peripheral in debug mode 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes. **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses.
2–4 LP_CFG	Peripheral configuration select for non-run modes - These bits associate a configuration as defined in the ME_LP_PC0 - 7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration

Table continues on the next page...

MC_ME_PCTL9 field descriptions (continued)

Field	Description
	100 Selects ME_LP_PC4 configuration
	101 Selects ME_LP_PC5 configuration
	110 Selects ME_LP_PC6 configuration
	111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes - These bits associate a configuration as defined in the ME_RUN_PC0 - 7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

62.3.29 SIPI_0 Control (MC_ME_PCTL11)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + CBh offset = CBh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL11 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.30 DECFILT_1 Control (MC_ME_PCTL28)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + DCh offset = DCh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL28 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.31 DECFILT_0 Control (MC_ME_PCTL29)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + DDh offset = DDh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL29 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.32 PIT_RTI_0 Control (MC_ME_PCTL30)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + DEh offset = DEh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL30 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.33 PIT_RTI_1 Control (MC_ME_PCTL31)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + DFh offset = DFh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL31 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.34 DMA_CH_MUX_0-2 Control (MC_ME_PCTL36)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

NOTE

DMA is a platform IP and in low power modes the clock to all platform IPs is shut off. Therefore the DMA cannot be kept ON in low power modes.

Address: 0h base + E4h offset = E4h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL36 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.35 CRC_0 Control (MC_ME_PCTL38)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + E6h offset = E6h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL38 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.36 JTAGM Control (MC_ME_PCTL45)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + EDh offset = EDh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL45 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.37 ADCSD_2 Control (MC_ME_PCTL59)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + FBh offset = FBh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL59 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.38 ADCSD_0 Control (MC_ME_PCTL60)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + FCh offset = FCh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL60 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.39 ADCSAR_2 Control (MC_ME_PCTL69)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 105h offset = 105h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL69 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.40 ADCSAR_0 Control (MC_ME_PCTL71)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 107h offset = 107h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL71 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.41 FLEXCAN_2 Control (MC_ME_PCTL81)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 111h offset = 111h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL81 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.42 FLEXCAN_0 Control (MC_ME_PCTL83)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 113h offset = 113h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL83 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.43 LINFLEX_M0 Control (MC_ME_PCTL85)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 115h offset = 115h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL85 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.44 LINFLEX_2 Control (MC_ME_PCTL91)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 11Bh offset = 11Bh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL91 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.45 LINFLEX_0 Control (MC_ME_PCTL92)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 11Ch offset = 11Ch

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL92 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.46 DSPI_M0 Control (MC_ME_PCTL96)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 120h offset = 120h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL96 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)

Table continues on the next page...

MC_ME_PCTL96 field descriptions (continued)

Field	Description
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.47 DSPI_4 Control (MC_ME_PCTL97)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 121h offset = 121h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL97 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.48 DSPI_2 Control (MC_ME_PCTL98)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 122h offset = 122h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL98 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.49 DSPI_0 Control (MC_ME_PCTL99)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 123h offset = 123h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL99 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.50 EMIOS Control (MC_ME_PCTL102)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 126h offset = 126h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL102 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.51 SENT_0 Control (MC_ME_PCTL104)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 128h offset = 128h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL104 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.52 DMA_CH_MUX_3 Control (MC_ME_PCTL143)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

NOTE

DMA is a platform IP and in low power modes the clock to all platform IPs is shut off. Therefore the DMA cannot be kept ON in low power modes.

Address: 0h base + 14Fh offset = 14Fh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL143 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.53 CRC_1 Control (MC_ME_PCTL166)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 166h offset = 166h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL166 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_PCTL166 field descriptions (continued)

Field	Description
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.54 ADCSD_1 Control (MC_ME_PCTL188)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 17Ch offset = 17Ch

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL188 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.55 ADCSAR_3 Control (MC_ME_PCTL196)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 184h offset = 184h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL196 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.56 IGF Control (MC_ME_PCTL197)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 185h offset = 185h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL197 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.57 ADCSAR_1 Control (MC_ME_PCTL198)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 186h offset = 186h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL198 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.58 ETPU_SCRAM Control (MC_ME_PCTL202)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 18Ah offset = 18Ah

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL202 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.59 ETPU_SCRAM Control (MC_ME_PCTL203)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 18Bh offset = 18Bh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL203 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.60 ETPU_SPRAM Control (MC_ME_PCTL204)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 18Ch offset = 18Ch

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL204 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)

Table continues on the next page...

MC_ME_PCTL204 field descriptions (continued)

Field	Description
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.61 ETPU_SPRAM Control (MC_ME_PCTL205)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 18Dh offset = 18Dh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL205 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.62 REACM Control (MC_ME_PCTL206)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

NOTE

REACM module can not have its clock enabled if the eTPU clock is disabled. Therefore, if MC_ME_PCTL207 register is configured to disable the eTPU clock, MC_ME_PCTL206 must always be configured to disable the REACM clock as well.

Address: 0h base + 18Eh offset = 18Eh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL206 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.63 ETPU Control (MC_ME_PCTL207)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 18Fh offset = 18Fh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL207 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.64 FLEXCAN_3 Control (MC_ME_PCTL208)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 190h offset = 190h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL208 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.65 FLEXCAN_1 Control (MC_ME_PCTL210)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 192h offset = 192h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL210 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.66 LINFLEX_M1 Control (MC_ME_PCTL213)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 195h offset = 195h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL213 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.67 LINFLEX_3 Control (MC_ME_PCTL219)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 19Bh offset = 19Bh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL219 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.68 LINFLEX_1 Control (MC_ME_PCTL220)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 19Ch offset = 19Ch

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL220 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.69 DSPI_M1 Control (MC_ME_PCTL225)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 1A1h offset = 1A1h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL225 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.70 DSPI_3 Control (MC_ME_PCTL226)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 1A2h offset = 1A2h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL226 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.71 DSPI_1 Control (MC_ME_PCTL227)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 1A3h offset = 1A3h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL227 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.72 EMIOS_1 Control (MC_ME_PCTL230)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 1A6h offset = 1A6h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL230 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.73 SENT_1 Control (MC_ME_PCTL232)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 1A8h offset = 1A8h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL232 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.74 BCTU Control (MC_ME_PCTL251)

The PCTLn registers select the configurations during run and non-run modes for each peripheral. The unoccupied locations contain a read-only byte value of 0x00.

Address: 0h base + 1BBh offset = 1BBh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL251 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	(See the Description for the ME_PCTL9 register.)
2–4 LP_CFG	(See the Description for the ME_PCTL9 register.)
5–7 RUN_CFG	(See the Description for the ME_PCTL9 register.)

62.3.75 Core Status (MC_ME_CS)

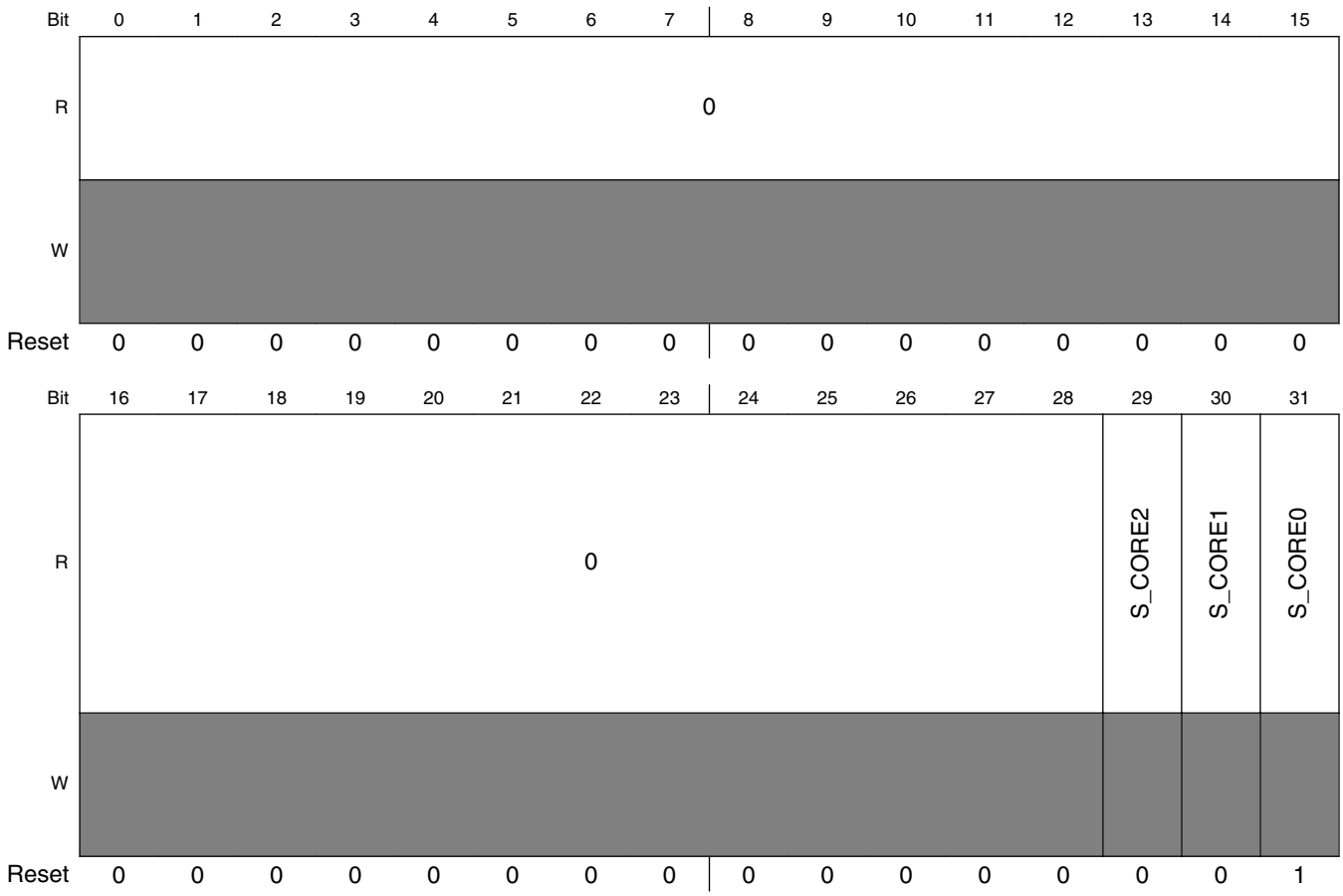
This register provides the status of each core.

The following table shows the correspondence between the core nomenclature in the block diagram and the ME module.

Table 62-94. Core Mapping with ME Registers

Core per Block Diagram	Core Description	Corresponding ME Control and Address Registers
Core 1	Computational Core	ME_CTTL0, ME_CADDR0
Core 0	Safety core which works with lockstep core	ME_CTTL1, ME_CADDR1
Core 0s	Lockstep core for Core 0	ME_CTTL2, ME_CADDR2

Address: 0h base + 1C0h offset = 1C0h



MC_ME_CS field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 S_CORE2	Core Status - This bit specifies the current status of an additional core which is controlled by the MC_ME. 0 Core is disabled 1 Core is running
30 S_CORE1	Core Status - This bit specifies the current status of an additional core which is controlled by the MC_ME. 0 Core is disabled 1 Core is running
31 S_CORE0	Core Status - This bit specifies the current status of an additional core which is controlled by the MC_ME. 0 Core is disabled 1 Core is running

62.3.76 CORE1 Control (MC_ME_CCTL0)

The CCTLn registers control whether each core is disabled or running during run modes. The unoccupied locations contain a read-only half-word value of 0x0000.

The run mode bits (TEST, SAFE, DRUN, and RUNx) of the ME_CCTL0 register are pre-loaded with the value provided by the device configuration just before reset exist.

Address: 0h base + 1C4h offset = 1C4h

Bit	0	1	2	3	4	5	6	7
Read	0					STOP0	0	HALT0
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
Write								
Reset	1	1	1	1	1	1	1	0

MC_ME_CCTL0 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 STOP0	Core control during STOP0 - CORE1 is always disabled during STOP mode 0 COREn is disabled with clock gated
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 HALT0	Core control during HALT0- CORE1 is always disabled during HALT mode 0 COREn is disabled with clock gated
8 RUN3	Core control during RUN3 0 CORE1 is disabled with clock gated 1 CORE1 is running
9 RUN2	Core control during RUN2 0 CORE1 is disabled with clock gated 1 CORE1 is running
10 RUN1	Core control during RUN1 0 CORE1 is disabled with clock gated 1 CORE1 is running
11 RUN0	Core control during RUN0 0 CORE1 is frozen with clock gated 1 CORE1 is running

Table continues on the next page...

MC_ME_CCTL0 field descriptions (continued)

Field	Description
12 DRUN	Core control during DRUN 0 CORE1 is frozen with clock gated 1 CORE1 is running
13 SAFE	Core control during SAFE 0 CORE1 is frozen with clock gated 1 CORE1 is running
14 TEST	Core control during TEST 0 CORE1 is frozen with clock gated 1 CORE1 is running
15 RESET	Core control during RESET - The core is always disabled during RESET.

62.3.77 CORE0 Control (MC_ME_CCTL1)

The CCTLn registers control whether each core is disabled or running during run modes. The unoccupied locations contain a read-only half-word value of 0x0000.

The run mode bits (TEST, SAFE, DRUN, and RUNx) of the ME_CCTL1 register are pre-loaded with the value provided by the device configuration just before reset exist.

Address: 0h base + 1C6h offset = 1C6h

Bit	0	1	2	3	4	5	6	7
Read	0					STOP0	0	HALT0
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_CCTL1 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 STOP0	Core control during STOP0 - CORE0 is always disabled during STOP mode
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_CCTL1 field descriptions (continued)

Field	Description
7 HALT0	Core control during HALT0 - CORE0 is always disabled during HALT mode
8 RUN3	Core control during RUN3 0 CORE0 is disabled with clock gated 1 CORE0 is running
9 RUN2	Core control during RUN2 0 CORE0 is disabled with clock gated 1 CORE0 is running
10 RUN1	Core control during RUN1 0 CORE0 is disabled with clock gated 1 CORE0 is running
11 RUN0	Core control during RUN0 0 CORE0 is frozen with clock gated 1 CORE0 is running
12 DRUN	Core control during DRUN 0 CORE0 is frozen with clock gated 1 CORE0 is running
13 SAFE	Core control during SAFE 0 CORE0 is frozen with clock gated 1 CORE0 is running
14 TEST	Core control during TEST 0 CORE0 is frozen with clock gated 1 CORE0 is running
15 RESET	Core control during RESET - The core is always disabled during RESET.

62.3.78 CORE2 Control (MC_ME_CCTL2)

The CCTLn registers control whether each core is disabled or running during run modes. The unoccupied locations contain a read-only half-word value of 0x0000.

The run mode bits (TEST, SAFE, DRUN, and RUNx) of the ME_CCTL2 register are pre-loaded with the value provided by the device configuration just before reset exist.

Address: 0h base + 1C8h offset = 1C8h

Bit	0	1	2	3	4	5	6	7
Read	0					STOP0	0	HALT0
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	TEST	RESET
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_CCTL2 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 STOP0	Core control during STOP0 - CORE2 is always disabled during STOP mode
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 HALT0	Core control during HALT0 - CORE2 is always disabled during HALT mode
8 RUN3	Core control during RUN3 0 CORE2 is disabled with clock gated 1 CORE2 is running
9 RUN2	Core control during RUN2 0 CORE2 is disabled with clock gated 1 CORE2 is running
10 RUN1	Core control during RUN1 0 CORE2 is disabled with clock gated 1 CORE2 is running
11 RUN0	Core control during RUN0 0 CORE2 is frozen with clock gated 1 CORE2 is running
12 DRUN	Core control during DRUN 0 CORE2 is frozen with clock gated 1 CORE2 is running
13 SAFE	Core control during SAFE 0 CORE2 is frozen with clock gated 1 CORE2 is running
14 TEST	Core control during TEST 0 CORE2 is frozen with clock gated 1 CORE2 is running
15 RESET	Core control during RESET - The core is always disabled during RESET.

62.3.79 CORE1 Address (MC_ME_CADDR0)

This register contains the boot address of CORE1 and a bit for controlling whether the core is to be reset on the next mode change that has the core configured to be running in the target mode.

These registers can be written only as words and cannot be written after a mode change request has been made until the mode transition has completed (i.e., while ME_GS[S_MTRANS] = ‘1’). A write access to any of these registers during this time will result in the ME_IS[ICONF_CC] flag being asserted.

ME_CADDR0 is pre-loaded with the value provided by the device configuration just before reset exit in order to ensure that CORE1 boots from the correct address after any chip reset if MC_CCTL0[DRUN] has a default value of '1’.

NOTE

The reset value of ME_CADDR0[ADDR] is determined by the chip configuration and boot mode.

Address: 0h base + 1E0h offset = 1E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADDR															
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ADDR														0	RMC
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

MC_ME_CADDR0 field descriptions

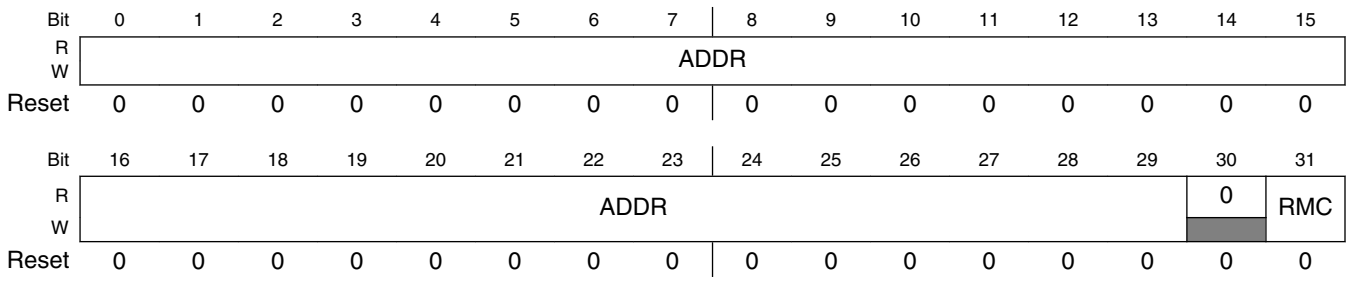
Field	Description
0–29 ADDR	Core Address - This field is used by CORE1 as the boot address (32-bit word aligned) when CORE1 next exits reset.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RMC	Reset on Mode Change - The core will be reset on the next mode change that has the core configured to be running in the target mode. 0 CORE1 will not be reset on the next mode change 1 CORE1 will be reset on the next mode change that has the core configured to be running in the target mode

62.3.80 CORE0 Address (MC_ME_CADDR1)

This register contains the boot address of CORE0 and a bit for controlling whether the core is to be reset on the next mode change that has the core configured to be running in the target mode.

ME_CADDR1 is pre-loaded with the value provided by the device configuration just before reset exit in order to ensure that CORE0 boots from the correct address after any chip reset if MC_CCTL1[DRUN] has a default value of ‘1’.

Address: 0h base + 1E4h offset = 1E4h



MC_ME_CADDR1 field descriptions

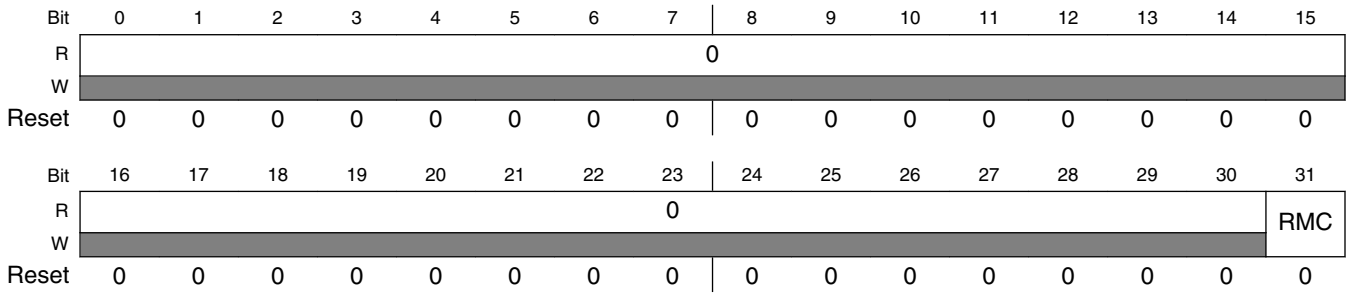
Field	Description
0–29 ADDR	Core Address - This field is used by CORE0 as the boot address (32-bit word aligned) when CORE0 next exits reset.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RMC	Reset on Mode Change - The core will be reset on the next mode change that has the core configured to be running in the target mode. 0 CORE0 will not be reset on the next mode change 1 CORE0 will be reset on the next mode change that has the core configured to be running in the target mode

62.3.81 CORE2 Address (MC_ME_CADDR2)

This register contains a bit for controlling whether the core is to be reset on the next mode change that has the core configured to be running in the target mode.

These registers can be written only as words and cannot be written after a mode change request has been made until the mode transition has completed (i.e., while ME_GS[S_MTRANS] =‘1’). A write access to any of these registers during this time will result in the ME_IS[ICONF_CC] being asserted.

Address: 0h base + 1E8h offset = 1E8h



MC_ME_CADDR2 field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RMC	Reset on Mode Change - The core will be reset on the next mode change that has the core configured to be running in the target mode. 0 CORE2 will not be reset on the next mode change 1 CORE2 will be reset on the next mode change that has the core configured to be running in the target mode

62.4 Functional Description

62.4.1 Mode Transition Request

The transition from one mode to another mode is normally handled by software by accessing the mode control register **ME_MCTL**. But the in case of special events, the mode transition can be automatically managed by hardware. In order to switch from one mode to another, the application should access the **ME_MCTL** register twice by writing

- the first time with the value of the key (0x5AF0) into the **KEY** bit field and the required target mode into the **TARGET_MODE** bit field,
- and the second time with the inverted value of the key (0xA50F) into the **KEY** bit field and the required target mode into the **TARGET_MODE** bit field.

Once a valid mode transition request is detected, the target mode configuration information is loaded from the corresponding **ME_<mode>_MC** register. The mode transition request may require a number of cycles depending on the programmed configuration, and software should check the **S_CURRENT_MODE** bit field and the

S_MTRANS bit of the global status register **ME_GS** to verify when the mode has been correctly entered and the transition process has completed. For a description of valid mode requests, please refer to Mode Transition Interrupts.

Any modification of the mode configuration register of the currently selected mode will not be taken into account immediately but on the next request to enter this mode. This means that transition requests such as ***RUN0…3*** → ***RUN0…3***, ***DRUN*** → ***DRUN***, ***SAFE*** → ***SAFE***, and ***TEST*** → ***TEST*** are considered valid mode transition requests. As soon as the mode request is accepted as valid, the **S_MTRANS** bit is set till the status in the **ME_GS** register matches the configuration programmed in the respective **ME_<mode>_MC** register.

Note

It is recommended that software polls the **S_MTRANS** bit in the **ME_GS** register after requesting a transition to ***HALT0*** or ***STOP0*** modes.

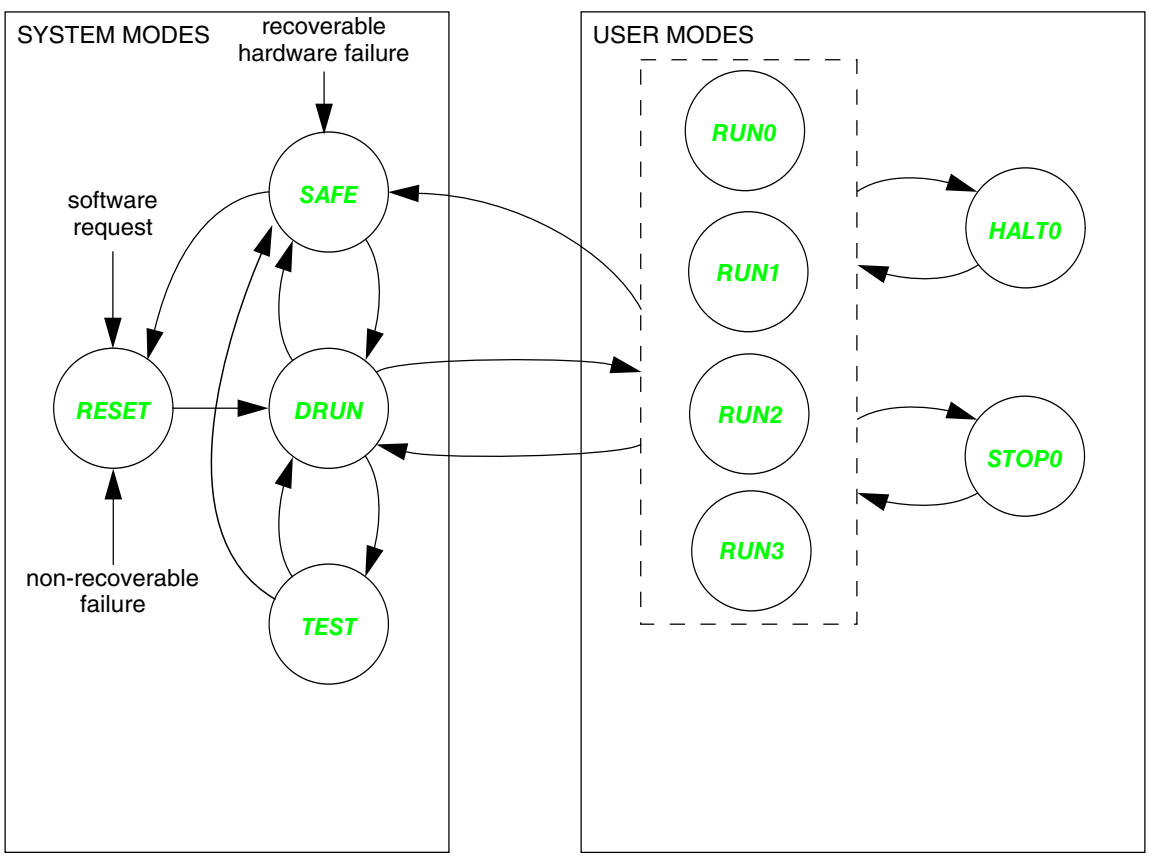


Figure 62-99. MC_ME Mode Diagram

All mode transitions shown in figure Figure 62-99 are initiated by software except for those from ***RESET*** and the low-power modes or to ***SAFE*** and ***RESET*** on a hardware failure. See Table 62-1 for a more detailed summary of mode entry and exit triggers.

62.4.2 Modes Details

62.4.2.1 *RESET* Mode

The chip enters this mode on the following events:

- from ***SAFE***, ***DRUN***, ***RUN0…3***, or ***TEST*** mode when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with either "0000" for a 'functional' reset or "1111" for a 'destructive' reset
- from any mode due to a system reset by the MC_RGM because of some non-recoverable hardware failure in the system (see the MC_RGM chapter for details)

Transition to this mode is instantaneous, and the system remains in this mode until the reset sequence is finished. The mode configuration information for this mode is provided by the **ME_RESET_MC** register. This mode has a pre-defined configuration, and the 16 MHz internal RC oscillator is selected as the system clock.

62.4.2.2 *DRUN* Mode

The chip enters this mode on the following events:

- automatically from ***RESET*** mode after completion of the reset sequence
- from ***RUN0…3***, ***SAFE***, or ***TEST*** mode when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with "0011"

As soon as any of the above events has occurred, a ***DRUN*** mode transition request is generated. The mode configuration information for this mode is provided by the **ME_DRUN_MC** register. In this mode, the flash, all clock sources, and the system clock configuration can be controlled by software as required. After system reset, the software execution starts with the default configuration selecting the 16 MHz internal RC oscillator as the system clock. The clock source configuration except system clock source 0 can be chosen by pre-loaded information in flash memory. This information is loaded into the **ME_DRUN_MC** register during ***PHASE3*** of the reset sequence.

This mode is intended to be used by software

- to initialize all registers as per the system needs

62.4.2.3 *SAFE* Mode

The chip enters this mode on the following events:

- from any mode except ***RESET*** when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with "0010"
- from any mode except ***RESET*** due to a ***SAFE*** mode request generated by the MC_RGM because of some potentially recoverable hardware failure in the system (see the MC_RGM chapter for details)

Note

If a hardware ***SAFE*** mode request occurs during ***RESET***, depending on the timing of the ***SAFE*** mode request, ***SAFE*** mode may be entered immediately after the normal completion of the reset sequence or several system clock cycles after ***DRUN*** entry. The ***SAFE*** mode request does not have any influence on the execution of the reset sequence itself.

As soon as any of the above events has occurred, a ***SAFE*** mode transition request is generated. The mode configuration information for this mode is provided by the **ME_SAFE_MC** register. This mode has a pre-defined configuration, and the 16 MHz internal RC oscillator is selected as the system clock.

If the ***SAFE*** mode is requested by hardware, a ***SAFE*** mode interrupt is generated. See *SAFE* Mode Transition Interrupt for details.

If the ***SAFE*** mode is requested by software while some other mode transition process is ongoing, the new target mode becomes the ***SAFE*** mode regardless of other pending requests or new requests during the mode transition. No new mode request made during a transition to the ***SAFE*** mode will cause an invalid mode interrupt.

Note

If software requests to change to the ***SAFE*** mode and then requests to change back to the parent mode before the mode transition is completed, the chip's final mode after mode transition will be the ***SAFE*** mode, and an invalid mode transition interrupt will be generated.

As long as a ***SAFE*** event is active, the system remains in the ***SAFE*** mode, and any software mode request during this time is ignored and lost.

This mode is intended to be used by software

- to assess the severity of the cause of failure and then to either
  - re-initialize the chip via the ***DRUN*** mode, or
  - completely reset the chip via the ***RESET*** mode.

If the outputs of the system I/Os need to be forced to a high impedance state upon entering this mode, the **PDO** bit of the **ME_SAFE_MC** register should be set. The input levels remain unchanged.

62.4.2.4 *TEST* Mode

The chip enters this mode on the following events:

- from the ***DRUN*** mode when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with "0001"

As soon as any of the above events has occurred, a ***TEST*** mode transition request is generated. The mode configuration information for this mode is provided by the **ME_TEST_MC** register. Except for the main voltage regulator, all resources of the system are configurable in this mode. The system clock to the whole system can be stopped by programming the **SYSCLK** bit field to "1111", and in this case, the only way to exit this mode is via a chip reset.

This mode is intended to be used by software

- to execute software test routines

62.4.2.5 *RUN0…3* Modes

The chip enters one of these modes on the following events:

- from the ***DRUN*** or another ***RUN0…3*** mode when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with "0100…0111"
- from the ***HALT0*** mode due to an interrupt event
- from the ***STOP0*** mode due to an interrupt or wakeup event

As soon as any of the above events has occurred, a ***RUN0…3*** mode transition request is generated. The mode configuration information for these modes is provided by the **ME_RUN0…3_MC** registers. In these modes, the flash, all clock sources, and the system clock configuration can be controlled by software as required.

These modes are intended to be used by software

- to execute application routines

62.4.2.6 *HALT0* Mode

The chip enters this mode on the following events:

- from one of the ***RUN0…3*** modes when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with "1000".

As soon as any of the above events has occurred, a ***HALT0*** mode transition request is generated. The mode configuration information for this mode is provided by **ME_HALT0_MC** register. This mode is quite configurable, and the **ME_HALT0_MC** register should be programmed according to the system needs. If there is a ***HALT0*** mode request while an interrupt request is active, the transition to ***HALT0*** is aborted with the resultant mode being the current mode, ***SAFE*** (on ***SAFE*** mode request), or ***DRUN*** (on reset), and an invalid mode interrupt is not generated.

This mode is intended as a first-level low-power mode with

- the core clocks frozen the additional core clocks frozen
- only a few peripherals running

and to be used by software

- to wait until it is required to do something and then to react quickly (i.e., within a few system clock cycles of an interrupt event)

The normal exit of the ***HALT0*** mode is from an interrupt event. Wakeup events will not cause ***HALT0*** mode exit.

Note

It is good practice for software to ensure that the **S_MTRANS** bit in the **ME_GS** register has been cleared on ***HALT0*** mode exit to ensure that the previous ***RUN0…3*** mode configuration has been fully restored before executing critical code.

62.4.2.7 *STOP0* Mode

The chip enters this mode on the following events:

- from one of the ***RUN0…3*** modes when the **TARGET_MODE** bit field of the **ME_MCTL** register is written with "1010".

As soon as any of the above events has occurred, a ***STOP0*** mode transition request is generated. The mode configuration information for this mode is provided by the **ME_STOP0_MC** register. This mode is fully configurable, and the **ME_STOP0_MC** register should be programmed according to the system needs. The following clock sources are switched off in this mode:

- the system PLL
- the secondary PLL

If there is a ***STOP0*** mode request while any interrupt or wakeup event is active, the transition to ***STOP0*** is aborted with the resultant mode being the current mode, ***SAFE*** (on ***SAFE*** mode request), or ***DRUN*** (on reset), and an invalid mode interrupt is not generated.

This can be used as an advanced low-power mode with the core clock frozen and almost all peripherals stopped.

This mode is intended as an advanced low-power mode with

- the core clock frozen
- the additional core clocks frozen
- almost all peripherals stopped

and to be used by software

- to wait until it is required to do something with no need to react quickly (e.g., allow for system clock source to be re-started)

The normal exit from the ***STOP0*** mode is by an interrupt or a wakeup event.

This mode can be used to stop all clock sources and thus preserve the chip status. When exiting the ***STOP0*** mode, the 16 MHz internal RC oscillator clock is selected as the system clock until the target clock is available.

Note

It is good practice for software to ensure that the **S_MTRANS** bit in the **ME_GS** register has been cleared on ***STOP0*** mode exit to ensure that the previous ***RUN0…3*** mode configuration has been fully restored before executing critical code.

The normal exit of the mode is from a wakeup event. Interrupt events will not cause mode exit.

62.4.3 Mode Transition Process

The process of mode transition follows the following steps in a pre-defined manner depending on the current chip mode and the requested target mode. In many cases of mode transition, not all steps need to be executed based on the mode control information, and some steps may not be applicable according to the mode definition itself.

Since the MC_ME ensures that all resources are in the proper state during the mode transition process, the following, for example, do not need to be checked by software.

- When the 4-40 MHz crystal oscillator is enabled in a mode configuration register, a mode transition to that mode will not complete until the End Of Count timeout occurs for the oscillator. Software does not need to interrogate the End Of Count timeout status bit.
- When a PLL is enabled in a mode configuration register, a mode transition to that mode will not complete until the PLL is locked. Software does not need to interrogate the PLL's lock status bit.

NOTE

After enabling the XOSC, an initiated mode transition is always completed, irrespective of whether there is a crystal on the board, whether the XTAL and EXTAL pins are open (floating) or whether the XTAL and EXTAL pins are shorted with a wire.

62.4.3.1 Target Mode Request

The target mode is requested by accessing the **ME_MCTL** register with the required keys. This mode transition request by software must be a valid request satisfying a set of pre-defined rules to initiate the process. If the request fails to satisfy these rules, it is ignored, and the **TARGET_MODE** bit field is not updated. An optional interrupt can be generated for invalid mode requests. Refer to Mode Transition Interrupts for details.

In the case of mode transitions occurring because of hardware events such as a reset, a ***SAFE*** mode request, or interrupt requests and wakeup events to exit from low-power modes, the **TARGET_MODE** bit field of the **ME_MCTL** register is automatically updated with the appropriate target mode. The mode change process start is indicated by the setting of the mode transition status bit **S_MTRANS** of the **ME_GS** register.

A ***RESET*** mode requested via the **ME_MCTL** register is passed to the MC_RGM, which generates a global system reset and initiates the reset sequence. The ***RESET*** mode request has the highest priority, and the MC_ME is kept in the ***RESET*** mode during the entire reset sequence.

The ***SAFE*** mode request has the next highest priority after reset. It can be generated either by software via the **ME_MCTL** register from all software running modes or by the MC_RGM after the detection of system hardware failures, which may occur in any mode.

62.4.3.2 Target Mode Configuration Loading

On completion of the Mode Transition Process step, the target mode configuration from the **ME_<target mode>_MC** register is loaded to start the resources (voltage sources, clock sources, flash, pads, etc.) control process.

An overview of resource control possibilities for each mode is shown in the following table. A '√' indicates that a given resource is configurable for a given mode.

Table 62-101. MC_ME Resource Control Overview

Resource	Mode						
	RESET	***TEST***	***SAFE***	***DRUN***	***RUN0...3***	***HALT0***	***STOP0***
IRC	on	√ on	on	on	on	√ on	√ on
XOSC	off	√ off	off	√ off	√ off	√ off	√ off
PLL0	off	√ off	off	√ off	√ off	√ off	off
PLL1	off	√ off	off	√ off	√ off	√ off	√ off
FLASH	normal	normal	normal	normal	normal	normal	normal
MVREG	on	on	on	on	on	√ on	√ on
PDO	off	√ off	√ on	off	off	off	√ off

62.4.3.3 Peripheral Clocks Disable

On completion of the Mode Transition Process step and the system clock frequency ramp-down of the System Clock Switching step, the MC_ME requests each peripheral to enter its stop mode when:

- the peripheral is configured to be disabled via the target mode, the peripheral configuration registers **ME_RUN_PC0…7** and **ME_LP_PC0…7**, and the peripheral control registers **ME_PCTL*n***

Note

The MC_ME automatically requests peripherals to enter their stop modes if the power domains in which they are residing are to be turned off due to a mode change. However, it is good practice for software to ensure that those peripherals that are to be powered down are configured in the MC_ME to be frozen.

Each peripheral acknowledges its stop mode request after closing its internal activity. The MC_ME then disables the corresponding clock(s) to this peripheral.

In the case of a ***SAFE*** mode transition request, the MC_ME does not wait for the peripherals to acknowledge the stop requests. The ***SAFE*** mode clock gating configuration is applied immediately regardless of the status of the peripherals' stop acknowledges.

Please refer to Peripheral Clock Gating for more details.

Each peripheral that may block or disrupt a communication bus to which it is connected ensures that these outputs are forced to a safe or recessive state when the chip enters the ***SAFE*** mode.

62.4.3.4 Processor Low-Power Mode Entry

If, on completion of the Peripheral Clocks Disable step, the mode transition is to the ***HALT0*** or ***STOP0*** mode, the MC_ME requests each processor to enter its stopped state. Each processor acknowledges its stop state request bit after completing all outstanding bus transactions.

If, on completion of the Peripheral Clocks Disable step, the mode transition is from a non-low-power mode to another non-low-power mode, the MC_ME requests each processor which is configured in the **ME_CCTL*n*** registers to be disabled to enter its stopped state. The processor acknowledges its stop state request bit after completing all outstanding bus transactions.

62.4.3.5 Processor Clocks Disable

If, on completion of the Processor Low-Power Mode Entry step, and all processors which are configured in the **ME_CCTL*n*** registers to be disabled are in their appropriate stopped state, the MC_ME disables the applicable processor clocks bits to achieve further power saving.

The clocks to processors which are configured to be running in both the current and target modes or disabled in both the current and target modes are unaffected while transitioning between non-low-power modes in that those bits of core_clk_en keep their previous value.

Note

Clocks to the whole chip including the processor and system memory can be disabled in ***TEST*** mode.

62.4.3.6 System Clock Disable

If, on completion of the Processor Clocks Disable step, all processor clocks have been disabled, the system clock used for non-processor system functions and memory is disabled.

62.4.3.7 Clock Sources Switch-On

On completion of the Processor Low-Power Mode Entry step, the MC_ME switches on all clock sources based on the **<clock source>ON** bits of the **ME_<current mode>_MC** and **ME_<target mode>_MC** registers. The following clock sources are switched on at this step:

- 16 MHz internal RC oscillator
- 4-40 MHz crystal oscillator

- PLL0
- PLL1

The clock sources that are required by the target mode are switched on. The duration required for the output clocks to be stable depends on the type of source, and all further steps of mode transition depending on one or more of these clocks waits for the stable status of the respective clocks. The availability status of these clocks is updated in the **S_<clock source>** bits of **ME_GS** register.

The clock sources which need to be switched off are unaffected during this process in order to not disturb the system clock which might require one of these clocks before switching to a different target clock.

62.4.3.8 Pad Outputs-On

On completion of the step, if the **PDO** bit of the **ME_<target mode>_MC** register is cleared, then

- all pad outputs are enabled to return to their previous state
- the I/O pads power sequence driver is switched on

62.4.3.9 Peripheral Clocks Enable

Based on the current and target chip modes, the peripheral configuration registers **ME_RUN_PC0…7**, **ME_LP_PC0…7**, and the peripheral control registers **ME_PCTLn,** the MC_ME enables the clocks for selected modules as required. This step is executed only after the process is completed, and if the value of the **PWRLVL** field of the **ME_<target mode>_MC** register is different from that of the **ME_<current mode>_MC** register's, the peripheral clock enable/disable isn't updated until after the system clock frequency ramp-down step of the System Clock Switching process is completed.

62.4.3.10 System and Processor Clocks Enable

The MC_ME enables the non-processor system clock and the clocks of all processors which are configured in the **ME_CCTL*n*** registers to be disabled in the current mode and to be running in the target mode.

The MC_ME initiates the resetting of all processors which are configured in the **ME_CCTL*n*** registers to be running in the target mode and which have their **RMC** bits in the **ME_CADDR*n*** registers set to '1'. The core_rst_b outputs remain asserted until the end of the mode transition as indicated by the **S_MTRANS** bit of the **MW_GS** register at which time they are deasserted.

If the value of the **PWRLVL** field of the **ME_<target mode>_MC** register is different from that of the **ME_<current mode>_MC** register's, the mode change handshake will not be initiated until after the system clock frequency ramp-down step of the System Clock Switching process is completed.

62.4.3.11 Processor Low-Power Mode Exit

On completion of the System and Processor Clocks Enable step, the MC_ME requests all processors which are configured in the **ME_CCTL*n*** registers to be disabled in the current mode and to be running in the target mode.

62.4.3.12 System Clock Switching

Based on the **SYSCLK** bit field of the **ME_<current mode>_MC** and **ME_<target mode>_MC** registers, if the target and current system clock configurations differ, the following method is implemented for clock switching.

- The target clock configuration for the 16 MHz internal RC oscillator takes effect only after the **S_IRC** bit of the **ME_GS** register is set by hardware (i.e., the 16 MHz internal RC oscillator has stabilized).
- The target clock configuration for the 4-40 MHz crystal oscillator takes effect only after the **S_XOSC** bit of the **ME_GS** register is set by hardware (i.e., the 4-40 MHz crystal oscillator has stabilized).
- The target clock configuration for the system PLL takes effect only after the **S_PLL0** bit of the **ME_GS** register is set by hardware (i.e., the system PLL has stabilized).
- If the clock is to be disabled, the **SYSCLK** bit field should be programmed with "1111". This is possible only in the ***TEST*** mode.

If the value of the **PWRLVL** field of the **ME_<target mode>_MC** register not equal to that of the **ME_<current mode>_MC** register's, a progressive clock switching is performed in the MC_CGM in order to reduce the impact of a sudden increase or drop in the power consumption of the device.

The current system clock configuration can be observed by reading the **S_SYSCLK** bit field of the **ME_GS** register, which is updated after every system clock switching. Until the target clock is available, the system uses the previous clock configuration.

System clock switching starts only after

- the Clock Sources Switch-On process has completed if the target system clock source is one of the following:
  - the 16 MHz internal RC oscillator
  - the 4-40 MHz crystal oscillator
- the Peripheral Clocks Disable process has completed in order not to change the system clock frequency before peripherals close their internal activitiesthe Peripheral Clocks Enable process has completed

An overview of system clock source selection possibilities for each mode is shown in the following table. A '√' indicates that a given clock source is selectable for a given mode.

Table 62-102. MC_ME System Clock Selection Overview

System Clock Source	Mode						
	RESET	***TEST***	***SAFE***	***DRUN***	***RUN0...3***	***HALT0***	***STOP0***
16 MHz int. RC osc.	√ (default)	√ (default)	√ (default)	√ (default)	√ (default)	√ (default)	√ (default)
4-40 MHz crystal osc.		√		√	√	√	√
system PLL		√		√	√	√	
system clock is disabled		√[1]					

1. disabling the system clock during ***TEST*** mode will require a reset in order to exit ***TEST*** mode

62.4.3.13 Pad Switch-Off

If the **PDO** bit of the **ME_<target mode>_MC** register is '1' then

- the outputs of the pads are forced to the high impedance state if the target mode is ***SAFE*** or ***TEST***

This step is executed only after the Peripheral Clocks Disable process has completed.

62.4.3.14 Clock Sources (with no Dependencies) Switch-Off

Based on the chip mode and the **<clock source>ON** bits of the **ME_<mode>_MC** registers, if a given clock source is to be switched off and no other clock source needs it to be on, the MC_ME requests the clock source to power down and updates its availability status bit **S_<clock source>** of the **ME_GS** register to '0'. The following clock sources are switched off at this step:

- the secondary PLL

For the clock sources related to the system clock, this step is executed only after the System Clock Switching process has completed.

62.4.3.15 Clock Sources (with Dependencies) Switch-Off

Based on the chip mode and the **<clock source>ON** bits of the **ME_<mode>_MC** registers, if a given clock source is to be switched off and all clock sources which need this clock source to be on have been switched off, the MC_ME requests the clock source to power down and updates its availability status bit **S_<clock source>** of the **ME_GS** register to '0'. The following clock sources are switched off at this step:

- the 4-40 MHz crystal oscillator
- the system PLL

This step is executed only after

- the System Clock Switching process has completed in order not to lose the current system clock during mode transition
- the Clock Sources (with no Dependencies) Switch-Off process has completed in order to, for example, prevent unwanted lock transitions

62.4.3.16 Current Mode Update

The current mode status bit field **S_CURRENT_MODE** of the **ME_GS** register is updated with the target mode bit field **TARGET_MODE** of the **ME_MCTL** register when :

- all the updated status bits in the **ME_GS** register match the configuration specified in the **ME_<target mode>_MC** register
- power sequences are done
- clock disable/enable process is finished
- processor low-power mode (stop) entry and exit processes are finished

Note

SAFE mode entry does not wait for the clock disable/enable process to finish. It only waits for the **ME_GS.S_RC** bit to be set. This is to ensure that the ***SAFE*** mode is entered as quickly as possible.

Software can monitor the mode transition status by reading the **S_MTRANS** bit of the **ME_GS** register. The mode transition latency can differ from one mode to another depending on the resources' availability before the new mode request and the target mode's requirements.

If a mode transition is taking longer to complete than is expected, the **ME_DMTS** register can indicate which process is still in progress.

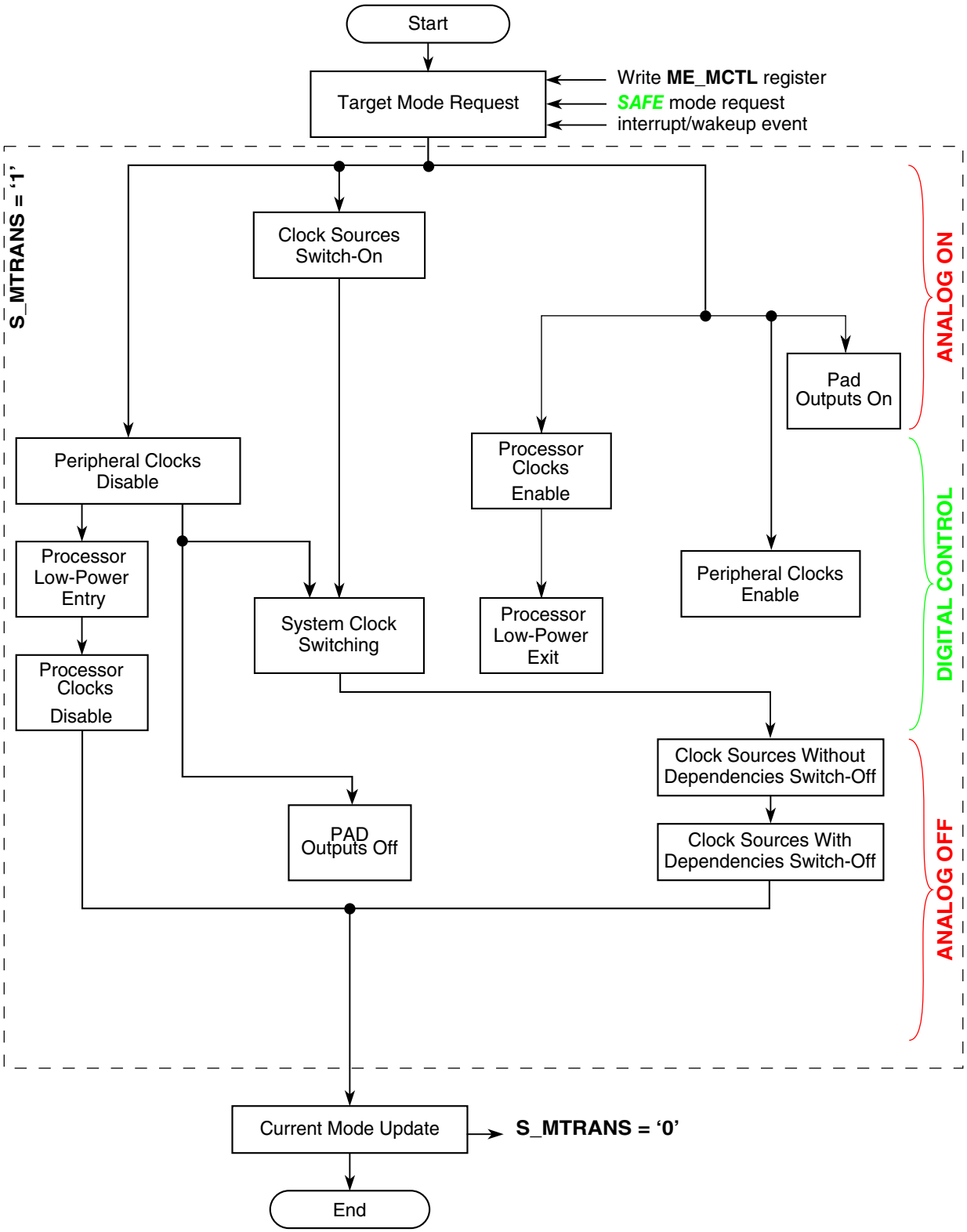


Figure 62-100. MC_ME Transition Diagram

62.4.4 Software Considerations for Preventing the Blocking of Mode Transitions

Mode transitions will not complete (i.e., mode transition is blocked) in certain conditions where

- resources are required in the target mode but are configured to be disabled in that target mode
- a low-power mode transition is attempted and there is a pending wakeup event
- a low-power mode transition is attempted and there is a pending interrupt

In these cases, **ME_STATUS[S_MTRANS]** never clears, **ME_IS[I_MTC]** never sets, and the invalid mode transition flags may not set.

While waiting for the mode transition to complete (e.g., polling **ME_GS[S_MTRANS]**), a timer can be used to detect if a mode transition takes too long. If the timer expires an error recovery function can be called.

To prevent mode transitions from being blocked, software must handle the following mode transition cases:

- **Enable required clock sources for the target mode's system clock selection.**

  Mode transitions can be blocked when a clock source to generate the system clock is not turned on in the target mode.

  Example for when XOSC is the reference clock for PLL0:

  > If the target mode is ***RUN1*** and **ME_RUN1_MC[SYSCLK]** is PLL0, then both **ME_RUN1_MC[XOSCON]** and **ME_RUN1_MC[PLL0ON]** must be set to '1'. If **ME_RUN1_MC[XOSCON]** is '0', the mode transition is blocked.

  Software must ensure that all required clock sources for generating the target mode's system clock are turned on for that target mode configuration.

- **Enable required clock sources for peripherals in the target mode.**

  Mode transitions can be blocked when an attempt is being made to stop a clock while one or more peripherals require this clock in the target mode.

  A given peripheral's clock is configured to be stopped in a target mode via peripheral configuration registers **ME_RUN_PC*x*** and **ME_LP_PC*x*** and the corresponding peripheral control register **ME_PCTL*x***. In addition, some peripherals may require local disabling before stopping their clocks via the MC_ME. Local disabling is accomplished by setting the peripheral's **MDIS** bit to '1'.

Modules requiring clocks being disabled before stopping them are:

- CAN modules

Example:

Assume a CAN module is using XOSC. A mode transition to ***RUN1*** is requested where **MC_RUN1_MC[XOSCON]** is '0'. The mode transition will not complete.

Software must disable the module whose clock source is to be turned off in the target mode.

- **Disable or service any pending interrupts when transitioning to low-power modes.**

  For transition into low power modes (***STOP0*** and ***HALT0***), mode transitions can be blocked by a peripheral's pending interrupt.

  Pending interrupts are detected by the MC_ME when the peripheral's interrupt flag is set, and the eripheral's interrupt enable is set.

  Note

  The **MSR[EE]** and **INTC_PSR*x*[PRI]** registers have no effect on the MC_ME detecting a pending interrupt.

  Software must either

  - disable interrupt enable flags in peripherals that may have their flag set, or,
  - detect and process the interrupt by either
    - polling the peripheral flag and servicing it in software, or,
    - taking the interrupt vector (requires **MSR[EE]** = 1 and **INTC_PSRx[PRI]** >0 for that peripheral.

- **Service any pending wakeup events when transitioning to low power modes.**

  For transition into low-power modes (***STOP0*** and ***HALT0***), mode transitions can be blocked by enabled wakeup events from the API or a wakeup pin transition.

  Example:

  The API is set to a very short value and enabled as a wakeup event. After the first of two writes to the **ME_MCTL** register to request a mode transition, the API expires. The mode transition will not complete because the wakeup event requires servicing.

Software must either

- ensure wakeup events cannot occur before low-power mode entry completion, or,
- detect and process the wakeup events during mode transition by either
  - polling the flags of enabled wakeup events in the **WISR** register and servicing it in software, or,
  - enabling wakeup events to cause an interrupt and execute the interrupt service routine (ISR).

Note

If the ISR also invokes a mode transition on it's own, then any prior writes to the **ME_MCTRL** register for the low-power mode transition will not be valid.

62.4.5 Protection of Mode Configuration Registers

While programming the mode configuration registers **ME_<mode>_MC**, the following rules must be respected. Otherwise, the write operation is ignored and an invalid mode configuration interrupt may be generated.

- If the 16 MHz internal RC oscillator is selected as the system clock, IRC must be on.
- If the 4-40 MHz crystal oscillator clock is selected as the system clock, XOSC must be on.
- If the system PLL clock is selected as the system clock, PLL0 must be on.
- If *CSRC_DEP[(12 × i) + j]* & csrc_dep_curr[(12 × i) + j] = '1' and **CSRC*j*ON** is '1', **CSRC*i*ON** must also be '1' (or in the case of *i* or *j* = 0, **RCON**).
- If PLL1 is on, XOSC must also be on.
- If PLL1 is on, PLL0 must also be on.

Note

Software must ensure that clock sources with dependencies other than those mentioned above are switched on as needed. There is no automatic protection mechanism to check this in the MC_ME.

- Configuration "00" for the **FLAON** bit field is reserved.
- System clock configurations marked as 'reserved' may not be selected.
- Configuration "1111" for the **SYSCLK** bit field is allowed only for the ***TEST*** mode, and only in this case may all system clock sources be turned off.

Note

If the system clock is stopped during ***TEST*** mode, the chip can exit only via a system reset.

62.4.6 Mode Transition Interrupts

The MC_ME provides interrupts for incorrectly configuring a mode, requesting an invalid mode transition, indicating a ***SAFE*** mode transition not due to a software request, and indicating when a mode transition has completed.

62.4.6.1 Invalid Mode Configuration Interrupt

Whenever a write operation is attempted to the **ME_<mode>_MC** registers violating the protection rules mentioned in Software Considerations for Preventing the Blocking of Mode Transitions, Mode transitions will not complete (i.e., mode transition is blocked) in certain conditions where the interrupt pending bit **I_ICONF** of the **ME_IS** register is set and an interrupt request is generated if the mask bit **M_ICONF** of the **ME_IM** register is '1'.

In addition, during a mode transition, if a clock source has been configured in the **ME_<target mode>_MC** register to be off and a peripheral requiring this clock source to be on has been enabled via the **ME_RUN_PC0…7/ME_LP_PC0…7** and **ME_PCTLn** registers, the interrupt pending bit **I_ICONF_CU** of the **ME_IS** register is set and an interrupt request is generated if the mask bit **M_ICONF_CU** of the **ME_IM** register is '1'.

If an attempt to write to one of the **ME_CCTLn** or **ME_CADDRn** registers is made after a mode change request has been made via the second write to the **ME_CTL** register and before the completion of that mode transition as indicated by the **S_MTRANS** bit in the **ME_GS** register deasserting, the interrupt pending bit **I_ICONF_CC** of the **ME_IS** register is set and an interrupt request is generated if the mask bit **M_ICONF_CC** of the **ME_IM** register is '1'.

62.4.6.2 Invalid Mode Transition Interrupt

The mode transition request is considered invalid under the following conditions:

- If the system is in the ***SAFE*** mode and the ***SAFE*** mode request from MC_RGM is active, and if the target mode requested is other than ***RESET*** or ***SAFE***, then this new mode request is considered to be invalid, and the **S_SEA** bit of the **ME_IMTS** register is set.
- If the **TARGET_MODE** bit field of the **ME_MCTL** register is written with a value different from the specified mode values (i.e., a non-existing mode), an invalid mode transition event is generated. When such a non existing mode is requested, the **S_NMA** bit of the **ME_IMTS** register is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the **ME_MCTL** register. The following modes are defined as non-existing:
  - "1001"
  - "1011"
  - "1100"
  - "1101"
  - "1110"
- If some of the chip modes are disabled as programmed in the **ME_ME** register, their respective configurations are considered reserved, and any access to the **ME_MCTL** register with those values results in an invalid mode transition request. When such a disabled mode is requested, the **S_DMA** bit of the **ME_IMTS** register is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the **ME_MCTL** register.
- If the target mode is not a valid mode with respect to the current mode, the mode request illegal status bit **S_MRI** of the **ME_IMTS** register is set. This condition is detected only when the proper key mechanism is followed while writing the **ME_MCTL** register. Otherwise, the write operation is ignored.

- If further new mode requests occur while a mode transition to a mode other than ***SAFE*** mode is in progress (the **S_MTRANS** bit of the **ME_GS** register is '1'), the mode transition illegal status bit **S_MTI** of the **ME_IMTS** register is set. This condition is detected only when the proper key mechanism is followed while writing the **ME_MCTL** register. Otherwise, the write operation is ignored.
- If a new mode request occurs while a mode transition to the ***SAFE*** mode is in progress (the **S_MTRANS** bit of the **ME_GS** register is '1'), the mode request ignored status bit **S_MRIG** of the **ME_IMTS** register is set. This condition is detected only when the proper key mechanism is followed while writing the **ME_MCTL** register. Otherwise, the write operation is ignored.

Note

As the causes of invalid mode transitions may overlap at the same time, the priority implemented for invalid mode transition status bits of the **ME_IMTS** register in the order from highest to lowest is **S_SEA**, **S_NMA**, **S_DMA**, **S_MRI**, **S_MRI**, and **S_MTI**.

As an exception, the mode transition request is not considered as invalid under the following conditions:

- A new request is allowed to enter the ***RESET*** or ***SAFE*** mode irrespective of the mode transition status.
- As the exit of ***HALT0*** and ***STOP0*** modes depends on the interrupts of the system which can occur at any instant, these requests to return to ***RUN0…3*** modes are always valid.
- In order to avoid any unwanted lockup of the chip modes, software can abort a mode transition by requesting the parent mode if, for example, the mode transition has not completed after a software determined 'reasonable' amount of time for whatever reason. The parent mode is the chip mode before a valid mode request was made.

CAUTION

Depending on the cause of an incomplete mode transition, aborting by requesting the parent mode might not succeed. In this case, software must request a change to either SAFE or RESET mode, because the cause of the lockup is most likely a hardware fault.

- Self-transition requests (e.g., ***RUN0*** → ***RUN0***) are not considered as invalid even when the mode transition process is active (i.e., **S_MTRANS** is '1'). During the low-power mode exit process, if the system is not able to enter the respective ***RUN0…3***

mode properly (i.e., all status bits of the **ME_GS** register match with configuration bits in the **ME_<mode>_MC** register), then software can only request the ***SAFE*** or ***RESET*** mode. It is not possible to request any other mode or to go back to the low-power mode again.

Whenever an invalid mode request is detected, the interrupt pending bit **I_IMODE** of the **ME_IS** register is set, and an interrupt request is generated if the mask bit **M_IMODE** of the **ME_IM** register is '1'.

62.4.6.3 *SAFE* Mode Transition Interrupt

Whenever the system enters the ***SAFE*** mode as a result of a ***SAFE*** mode request from the MC_RGM due to a hardware failure, the interrupt pending bit **I_SAFE** of the **ME_IS** register is set, and an interrupt is generated if the mask bit **M_SAFE** of **ME_IM** register is '1'.

The ***SAFE*** mode interrupt pending bit can be cleared only when the ***SAFE*** mode request is deasserted by the MC_RGM (see the MC_RGM chapter for details on how to clear a ***SAFE*** mode request). If the system is already in ***SAFE*** mode, any new ***SAFE*** mode request by the MC_RGM also sets the interrupt pending bit **I_SAFE**. However, the ***SAFE*** mode interrupt pending bit is not set when the ***SAFE*** mode is entered by a software request (i.e., programming of **ME_MCTL** register).

62.4.6.4 Mode Transition Complete interrupt

Whenever the system fully completes a mode transition (i.e., the **S_MTRANS** bit of **ME_GS** register transits from '1' to '0'), the interrupt pending bit **I_MTC** of the **ME_IS** register is set, and an interrupt request is generated if the mask bit **M_MTC** of the **ME_IM** register is '1'. The interrupt bit **I_MTC** is not set when entering low-power modes ***HALT0*** and ***STOP0*** in order to avoid the same event requesting the immediate exit of these low-power modes.

62.4.7 Peripheral Clock Gating

During all chip modes, each peripheral can be associated with a particular clock gating policy determined by two groups of peripheral configuration registers.

The run peripheral configuration registers **ME_RUN_PC0…7** are chosen only during the software running modes ***DRUN***, ***TEST***, ***SAFE***, and ***RUN0…3***. All configurations are programmable by software according to the needs of the application. Each configuration

register contains a mode bit which determines whether or not a peripheral clock is to be gated. Run configuration selection for each peripheral is done by the **RUN_CFG** bit field of the **ME_PCTL*n*** registers.

The low-power peripheral configuration registers **ME_LP_PC0…7** are chosen only during the low-power modes ***HALT0*** and ***STOP0***. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Low-power configuration selection for each peripheral is done by the **LP_CFG** bit field of the **ME_PCTL*n*** registers.

Any modifications to the **ME_RUN_PC0…7**, **ME_LP_PC0…7**, and **ME_PCTL*n*** registers do not affect the clock gating behavior until a new mode transition request is generated.

Whenever the chip enters a debug session during any mode, the following occurs for each peripheral:

- The clock is gated if the **DBG_F** bit of the associated **ME_PCTL*n*** register is set. Otherwise, the peripheral clock gating status depends on the **RUN_CFG** and **LP_CFG** bits.

62.4.8 Initialization / Application Information

The following figure shows an example application flow for requesting a mode change and then waiting until the mode transition has completed.

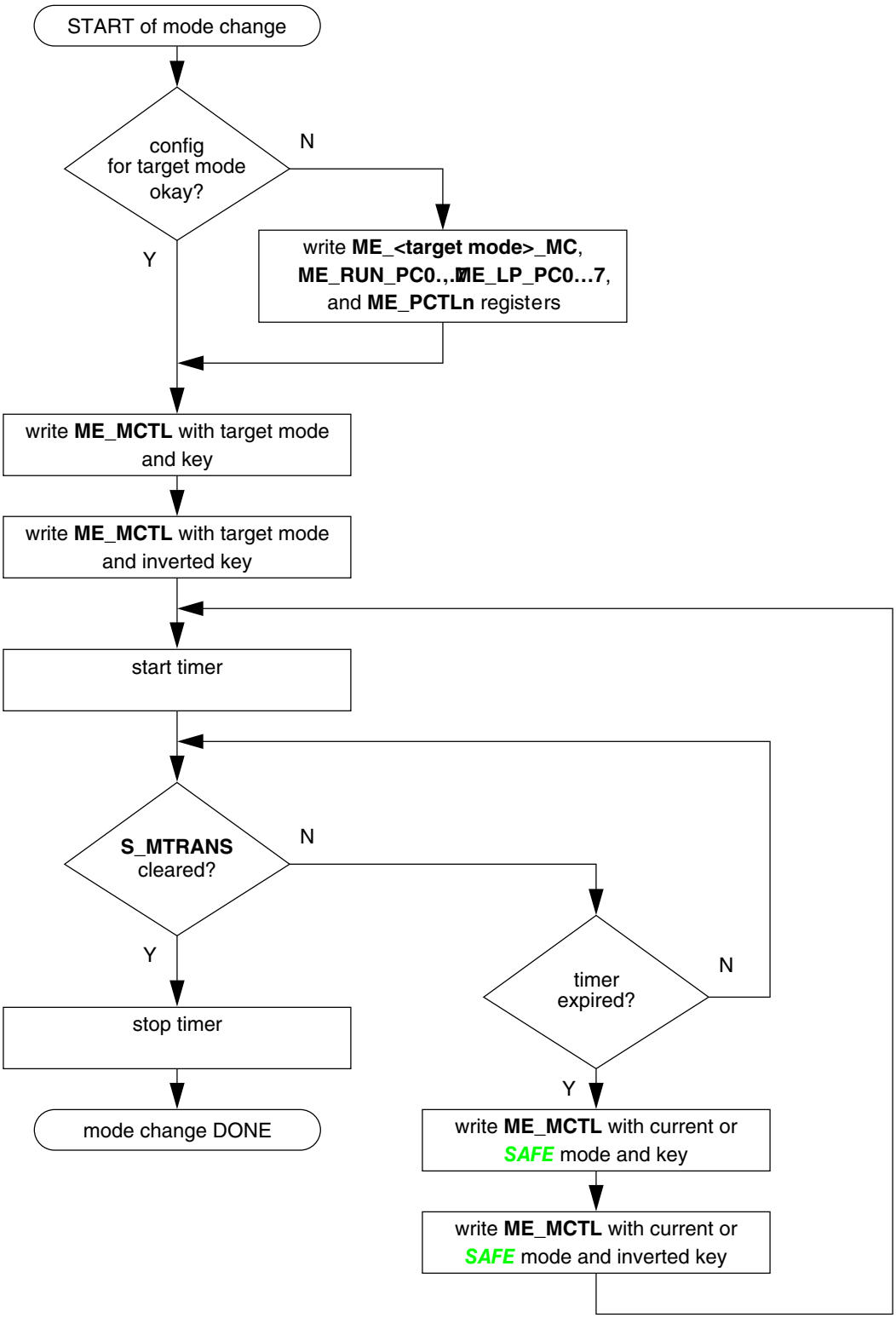


Figure 62-101. MC_ME Application Example Flow Diagram

Chapter 63
Calibration and Debug

63.1 Introduction

This chapter discusses the chip-specific information for the debug and calibration modules. All standards used have freely available specifications for tool developers.

The debug and calibration features for this device are provided in two separate dies: the Production die and the Buddy die, or Buddy Device (BD). The Production die is packaged alone in the Production Device (PD) package. The PD is also assembled into an Emulation and Debug Device (ED) package that consists of the Production die along with the Buddy die. The Buddy device supports additional debug capability.

The ED implements a high-speed serial Nexus trace auxiliary port. The Aurora interface allows the Nexus protocol information to be transmitted serially at high speed over four Aurora lanes. The Aurora protocol handles the encoding of the data and striping the information across the four Aurora lanes available on the device.

63.2 Core debug support

Internal debug support in the e200z425n cores allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes. The cores have the capability of exiting reset halted in debug mode, such that no code runs. This allows debuggers to perform operations such as configuration of calibration remap or edit of overlay memory prior to user code execution.

For details, see Core Debug Support.

63.3 Run control and memory access

Run control is any operation required to control device operation. Run control includes starting and stopping the processor's execution of code, as well as accessing memory while the device is stopped to download code. The Debug and Calibration Interface (DCI) provides the central run control including cross-triggering of breakpoint conditions. This DCI provides support for the IEEE 1149.1 JTAG standard with the JTAG Controller (JTAGC) as well as the IEEE 1149.7 standard with the Compact JTAG (CJTAG) module. The JTAG interface supports both Boundary Scan and Debug modes. It is based on the IEEE 1149.1 and IEEE 1149.7 standards. The JTAG Controller (JTAGC) operates in the standard IEEE 1149.1 5-wire interface. The JTAGC can either be fed directly from the JTAG pins or the CJTAG module, which supports the IEEE 1149.7 standard.

The Sequence Processing Unit (SPU) provides cross-triggering of events from the device. The SPU can access some of the DCI run control features since it can control the $\overline{\text{EVTO}}$ pins and monitor the $\overline{\text{EVTI}}$ pins. Furthermore, the SPU can use system triggers to cause many different types of actions, including triggering a breakpoint cross-trigger (managed by the DCI).

The following table shows the JTAG operating frequencies.

Table 63-1. JTAG operating frequencies

PD configuration	PD system clock frequency [MHz]	Max JTAG frequency [MHz]	Notes
Unpowered, in reset state, or powered, out of reset, PLL not engaged	16	3.88	Assuming 64 MHz PD IRC with 30% tolerance
Powered, out of reset, PLL engaged	190	See datasheet	JTAG speed is limited by TDO delay (16 ns as per PES) and setup time with respect to TCK (5 ns)

Table 63-2. Reference links to related information

Related module	Reference
Production Device Debug and Calibration Interface (DCI)	Debug and Calibration Interface (DCI)
JTAG Controller (JTAGC)	JTAG Controller (JTAGC)
IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)	IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)
JTAG Data Communication (JDC)	JTAG Data Communication (JDC)
Sequence Processing Unit (SPU)	Sequence Processing Unit (SPU)

63.3.1 Debug and Calibration Interface (DCI)

The Debug and Calibration Interface (DCI) module provides debug and calibration features. The DCI module includes the device JTAG Controller and the IEEE 1149.7 interface. The DCI provides the following features:

- Debug mode enable control for connected tools or software
- IEEE 1149.1 and 1149.7 controllers
- Debug break and cross-triggering control
- Synchronous restart control for all CPUs when exiting debug mode
- Tool hot plug capability
- Security access control
- Simultaneous operation for debug and calibration features

63.3.1.1 DCI signals

The DCI sends and receives the $\overline{\text{EVTI}}$[1:0] and $\overline{\text{EVTO}}$[1:0] signals to/from the cores and other modules. The tables below summarize the source and destination of $\overline{\text{EVTI}}$, $\overline{\text{EVTO}}$, and related signals for the PD and the BD. For example, for the PD, $\overline{\text{EVTI}}$[0] are outputs of the JTAGM and SPU and inputs to the cores and NXMC.

Table 63-3. PD DCI signals

Signal	Cores	NXMC	NAR	JTAGM	eTPU	SPU	DTS
$\overline{\text{EVTI}}$[0]	I	I	—	O	I	I/O	—
$\overline{\text{EVTI}}$[1]	—	—	—	O	—	I/O	—
$\overline{\text{EVTO}}$[0]	O	O	—	I[1, 1]	O	I/O	—
$\overline{\text{EVTO}}$[1]	—	—	—	I[1]	—	I/O	O
Core debug event	I	—	—	—	—	—	—
DCI cross triggering	O	—	—	—	O	—	—

1. The EVTO signal provided to the JTAGM is stretched by the DCI on the 16MHz IRC clock. Therefore, the incoming EVTO cannot be reliably detected by the JTAGM when the JTAGM clock is configured at less than 32MHz.

Table 63-4. BD DCI signals

Signal	RWA	NAR	NAL	JTAGM
$\overline{\text{EVTI}}$[0]	—	I	—	O

Table continues on the next page...

Table 63-4. BD DCI signals (continued)

Signal	RWA	NAR	NAL	JTAGM
EVTI[1]	—	—	—	O
EVTO[0]	—	—	—	I[1, 1]
EVTO[1]	—	—	—	I[1]

1. The EVTO signal provided to the JTAGM is stretched by the DCI on the 16MHz IRC clock. Therefore, the incoming EVTO cannot be reliably detected by the JTAGM when the JTAGM clock is configured at less than 32MHz.

The DCI sends the system debug signal, ipg_debug, to the modules below. For details regarding module functionality during system debug, refer to the individual module chapter.

- CAN
- LFAST (Interprocessor Communications instance only)
- DECFILTER
- DSPI
- eDMA
- eMIOS
- eTPU
- FCCU
- IGF
- MC_ME
- PIT
- REACM2
- SDADC
- SIPI
- SRX
- STM
- SWT

63.3.1.2 DCI EVTx Pin Multiplexing Control Register (DCI_PINCR)

The DCI EVTx Pin Multiplexing Control Register (DCI_PINCR) is described below. The EVTI and EVTO functions share the same set of possible pins. If this register is written to enable both EVTI and EVTO on the same pin, the EVTI function is selected, although

this type of programming should be avoided. Programming the $\overline{\text{EVTI}}$ fields to select $\overline{\text{EVTI}}$[0] or $\overline{\text{EVTI}}$[1] inputs to come from more than one pin is considered a programming error and no pin is enabled.

	31	30	29	28	27	26	25	24
R W	Reserved							
Reset	0	0	0	0	0	0	0	0
	23	22	21	20	19	18	17	16
R W	Reserved							
Reset	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8
R W	Reserved			EVTO1B	EVTO1A	EVTO0C	EVTO0B	EVTO0A
Reset	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
R W	Reserved			EVTI1B	EVTI1A	EVTI0C	EVTI0B	EVTI0A
Reset	0	0	0	0	0	0	0	0

Table 63-5. DCI_PINCR field descriptions

Field	Description
31-13	Reserved
12 EVTO1B	Enables $\overline{\text{EVTO}}$[1] function on pin PL[1]
11 EVTO1A	Enables $\overline{\text{EVTO}}$[1] function on pin PJ[1]
10 EVTO0C	Enables $\overline{\text{EVTO}}$[0] function on pin PL[0]
9 EVTO0B	Enables $\overline{\text{EVTO}}$[0] function on pin PJ[7]
8 EVTO0A	Enables $\overline{\text{EVTO}}$[0] function on pin PJ[0]
7-5	Reserved
4 EVTI1B	Enables $\overline{\text{EVTI}}$[1] function on pin PL[1]

Table continues on the next page...

Table 63-5. DCI_PINCR field descriptions (continued)

Field	Description
3 EVTI1A	Enables $\overline{\text{EVTI}}$[1] function on pin PJ[3]
2 EVTI0C	Enables $\overline{\text{EVTI}}$[0] function on pin PL[0]
1 EVTI0B	Enables $\overline{\text{EVTI}}$[0] function on pin PJ[5]
0 EVTI0A	Enables $\overline{\text{EVTI}}$[0] function on pin PJ[4]

63.3.1.3 JTAG Controller (JTAGC)

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format. All Nexus debug registers are accessed and configured via the JTAG Controller.

63.3.1.3.1 JTAGC ACCESS_AUX block instructions

The PD and BD ACCESS_AUX instructions are described in these tables.

Table 63-6. PD ACCESS_AUX 6-bit JTAG instructions

Instruction	Code[5:0]	Instruction Summary
ACCESS_AUX_x	100000-111110	Grants one of the auxiliary TAP controllers ownership of the TAP as shown in the cells below.
Reserved	100000	Reserved
ACCESS_AUX_NR_PD	100001	Enables access to the NAR TAP controller on PD
Reserved	100010	Reserved
Reserved	100011	Reserved
Reserved	100100	Reserved
Reserved	100101	Reserved
ACCESS_AUX_JDC	100110	Enables access to the JDC TAP controller
Reserved	100111	Reserved
ACCESS_AUX_CORE_0	101000	Enabled access to E200 Core 0
ACCESS_AUX_CORE_1	101001	Enables access to E200 Core 1
Reserved	101010	Reserved
Reserved	101011	Reserved

Table continues on the next page...

Table 63-6. PD ACCESS_AUX 6-bit JTAG instructions (continued)

Instruction	Code[5:0]	Instruction Summary
Reserved	101100	Reserved
Reserved	101101	Reserved
Reserved	101110	Reserved
Reserved	101111	Reserved
ACCESS_AUX_ETPU	110000	Enables access to eTPU controller
Reserved	110001	Reserved
Reserved	110010	Reserved
Reserved	110011	Reserved
ACCESS_AUX_BUS_MON_0	110100	Enables access to NXMC Client 0 TAP controller
ACCESS_AUX_BUS_MON_1	110101	Enables access to NXMC Client 1 TAP controller
Reserved	110110	Reserved
Reserved	110111	Reserved
Reserved	111000	Reserved
Reserved	111001	Reserved
ACCESS_AUX_SPU_A	111010	Enables access to SPU Client TAP controller
Reserved	111011	Reserved
ACCESS_AUX_PARALLEL[1]	111100	Enables access to all cores in parallel
Reserved	111101	Reserved
Reserved	111110	Reserved

1. The use of this instruction is limited to the synchronous exit of debug mode via the execution of the GO+EXIT OnCE command. The execution of other instructions or commands is not supported

Table 63-7. BD ACCESS_AUX 6-bit JTAG instructions

Instruction	Code[5:0]	Instruction Summary
ACCESS_AUX_x	100000-111110	Grants one of the auxiliary TAP controllers ownership of the TAP as shown in the cells below.
ACCESS_AUX_NR_BD	100000	Enables access to the NAR TAP controller on BD
Reserved	100001	Reserved
ACCESS_AUX_NAL	100010	Enables access to the NAL TAP controller
ACCESS_AUX_RWA_BD	100011	Enables access to the RWA TAP controller
Reserved	100100	Reserved
Factory test reserved	100101	Intended for factory test only
Reserved	100110	Reserved
Reserved	100111	Reserved
Reserved	101000	Reserved
Reserved	101001	Reserved
Reserved	101010	Reserved
Reserved	101011	Reserved

Table continues on the next page...

Table 63-7. BD ACCESS_AUX 6-bit JTAG instructions (continued)

Instruction	Code[5:0]	Instruction Summary
Reserved	101100	Reserved
Reserved	101101	Reserved
Reserved	101110	Reserved
Reserved	101111	Reserved
Reserved	110000	Reserved
Reserved	110001	Reserved
Reserved	110010	Reserved
Reserved	110011	Reserved
Reserved	110100	Reserved
Reserved	110101	Reserved
Reserved	110110	Reserved
Reserved	110111	Reserved
Reserved	111000	Reserved
Reserved	111001	Reserved
Reserved	111010	Reserved
Reserved	111011	Reserved
Reserved	111100	Reserved
Reserved	111101	Reserved
ACCESS_AUX_DCI_PD	111110	Enables access to pass control from BD to PD

63.3.1.4 Compact JTAG (CJTAG)

The CJTAG module implements parts of the IEEE 1149.7 standard for test and debug capabilities.

63.3.2 JTAG Data Communication (JDC)

The JTAG Data Communication (JDC) module allows access to two registers by JTAG and software running on one of the CPUs in the device. These registers can be used to exchange data between an internal CPU and an external debug tool.

63.3.3 Sequence Processing Unit (SPU)

The Sequence Processing Unit (SPU) provides on-device logic analyzer trigger functions, such as performance monitoring, by using internal device signals as trigger sources. Performance monitor functions are available at both the system and CPU level. Complex trigger and system performance monitor functions are implemented in the SPU module. System-level performance monitor functions are integrated into the SPU complex trigger logic, thus allowing the counting and timing of any debug trigger combinations supported by the SPU.

Various clients generate watchpoints and other triggers when operating in Debug mode. The SPU collects these triggers and uses them as conditions for programmable sequences of states and resultant actions. The SPU provides the capability to create complex debug triggers. By configuring each of the events to trigger specific actions, it achieves complex logic-analyzer-like behavior, providing vital real-time visibility and the ability to debug system activities.

63.3.3.1 SPU Level1 Input Mux Configuration registers

The SPU Level1 Mux inputs for the L1SELn registers (where n = 0–7) are described in the following tables. The L1SEL0 register fields are described in the following table.

Table 63-8. L1SEL0 register field descriptions

Field	Description	
MUX 0	000	CPU0 instruction address compare 1 (watchpoint 0)
	001	CPU1 instruction address compare 1 (watchpoint 0)
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	CPU0 debug event input 1 (watchpoint 10)
	111	CPU0 debug event output 2 (watchpoint 20)
MUX 1	000	CPU0 instruction address compare 2 (watchpoint 1)
	001	CPU1 instruction address compare 2 (watchpoint 1)
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	CPU1 debug event input 1 (watchpoint 10)
	111	CPU1 debug event output 2 (watchpoint 20)

Table continues on the next page...

Table 63-8. L1SEL0 register field descriptions (continued)

Field	Description	
MUX 2	000	CPU0 instruction address compare 3 (watchpoint 2)
	001	CPU1 instruction address compare 3 (watchpoint 2)
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	Reserved
MUX 3	000	CPU0 instruction address compare 4 (watchpoint 3)
	001	CPU1 instruction address compare 4 (watchpoint 3)
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	Reserved
MUX 4	000	CPU0 instruction address compare 5 (watchpoint 8)
	001	CPU1 instruction address compare 5 (watchpoint 8)
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	CPU0 debug event input 2 (watchpoint 11)
	111	CPU0 debug event output 3 (watchpoint 21)
MUX 5	000	CPU0 instruction address compare 6 (watchpoint 9)
	001	CPU1 instruction address compare 6 (watchpoint 9)
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	CPU1 debug event input 2 (watchpoint 11)
	111	CPU1 debug event output 3 (watchpoint 21)

Table continues on the next page...

Table 63-8. L1SEL0 register field descriptions (continued)

Field	Description	
MUX 6	000	CPU0 instruction address compare 7 (watchpoint 14)
	001	CPU1 instruction address compare 7 (watchpoint 14)
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	Reserved
MUX 7	000	CPU0 instruction address compare 8 (watchpoint 15)
	001	CPU1 instruction address compare 8 (watchpoint 15)
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	Reserved

The L1SEL1 register fields are described in the following table.

Table 63-9. L1SEL1 register field descriptions

Field	Description	
MUX 8	000	CPU0 instruction address compare 1 (watchpoint 0)
	001	CPU1 instruction address compare 1 (watchpoint 0)
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	CPU0 debug event output 0 (watchpoint 12)
	111	Reserved
MUX 9	000	CPU0 instruction address compare 2 (watchpoint 1)
	001	CPU1 instruction address compare 2 (watchpoint 1)
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	CPU1 debug event output 0 (watchpoint 12)
	111	Reserved

Table continues on the next page...

Table 63-9. L1SEL1 register field descriptions (continued)

Field	Description	
MUX 10	000	CPU0 instruction address compare 3 (watchpoint 2)
	001	CPU1 instruction address compare 3 (watchpoint 2)
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	Reserved
MUX 11	000	CPU0 instruction address compare 4 (watchpoint 3)
	001	CPU1 instruction address compare 4 (watchpoint 3)
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	Reserved
MUX 12	000	CPU0 instruction address compare 5 (watchpoint 8)
	001	CPU1 instruction address compare 5 (watchpoint 8)
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	CPU0 debug event output 1 (watchpoint 13)
	111	Reserved
MUX 13	000	CPU0 instruction address compare 6 (watchpoint 9)
	001	CPU1 instruction address compare 6 (watchpoint 9)
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	CPU1 debug event output 1 (watchpoint 13)
	111	Reserved

Table continues on the next page...

Table 63-9. L1SEL1 register field descriptions (continued)

Field	Description	
MUX 14	000	CPU0 instruction address compare 7 (watchpoint 14)
	001	CPU1 instruction address compare 7 (watchpoint 14)
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	Reserved
MUX 15	000	CPU0 instruction address compare 8 (watchpoint 15)
	001	CPU1 instruction address compare 8 (watchpoint 15)
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	Reserved

The L1SEL2 register fields are described in the following table.

Table 63-10. L1SEL2 register field descriptions

Field	Description	
MUX16	000	CPU0 data address compare 1 (watchpoint 4)
	001	CPU1 data address compare 1 (watchpoint 4)
	010	Reserved
	011	Reserved
	100	CPU0 write to L1CSR0
	101	Reserved
	110	Reserved
	111	Reserved
MUX 17	000	CPU0 data address compare 2 (watchpoint 5)
	001	CPU1 data address compare 2 (watchpoint 5)
	010	Reserved
	011	Reserved
	100	CPU0 write to L1CSR1
	101	Reserved
	110	Reserved
	111	Reserved

Table continues on the next page...

Table 63-10. L1SEL2 register field descriptions (continued)

Field	Description	
MUX 18	000	CPU0 data address compare 3 (watchpoint 6)
	001	CPU1 data address compare 3 (watchpoint 6)
	010	Reserved
	011	Reserved
	100	CPU0 write to E2ECTL0
	101	Reserved
	110	Reserved
	111	Reserved
MUX 19	000	CPU0 data address compare 4 (watchpoint 7)
	001	CPU1 data address compare 4 (watchpoint 7)
	010	Reserved
	011	Reserved
	100	CPU0 write to DMEMCTL0
	101	Reserved
	110	Reserved
	111	Reserved
MUX 20	000	CPU0 data address compare 1 (watchpoint 4)
	001	CPU1 data address compare 1 (watchpoint 4)
	010	Reserved
	011	Reserved
	100	CPU0 write to IMEMCTL0
	101	Reserved
	110	Reserved
	111	Reserved
MUX 21	000	CPU0 data address compare 2 (watchpoint 5)
	001	CPU1 data address compare 2 (watchpoint 5)
	010	Reserved
	011	Reserved
	100	CPU0 write to HID0
	101	CPU1 write to MSR
	110	Reserved
	111	Reserved

Table continues on the next page...

Table 63-10. L1SEL2 register field descriptions (continued)

Field	Description	
MUX 22	000	CPU0 data address compare 3 (watchpoint 6)
	001	CPU1 data address compare 3 (watchpoint 6)
	010	Reserved
	011	Reserved
	100	CPU0 write to HID1
	101	CPU1 write to IVPR
	110	Reserved
	111	Reserved
MUX 23	000	CPU0 data address compare 4 (watchpoint 7)
	001	CPU1 data address compare 4 (watchpoint 7)
	010	Reserved
	011	Reserved
	100	CPU0 write to BUCSR
	101	CPU1 write to SPEFSCR
	110	Reserved
	111	Reserved

The L1SEL3 register fields are described in the following table.

Table 63-11. L1SEL3 register field descriptions

Field	Description	
MUX 24	000	CPU0 data address compare 1 (watchpoint 4)
	001	CPU1 data address compare 1 (watchpoint 4)
	010	Reserved
	011	Reserved
	100	CPU0 write to SPEFSCR
	101	CPU1 write to BUCSR
	110	Reserved
	111	Reserved
MUX 25	000	CPU0 data address compare 2 (watchpoint 5)
	001	CPU1 data address compare 2 (watchpoint 5)
	010	Reserved
	011	Reserved
	100	CPU0 write to IVPR
	101	CPU1 write to HID1
	110	Reserved
	111	Reserved

Table continues on the next page...

Table 63-11. L1SEL3 register field descriptions (continued)

Field	Description	
MUX 26	000	CPU0 data address compare 3 (watchpoint 6)
	001	CPU1 data address compare 3 (watchpoint 6)
	010	Reserved
	011	Reserved
	100	CPU0 write to MSR
	101	CPU1 write to HID0
	110	Reserved
	111	Reserved
MUX 27	000	CPU0 data address compare 4 (watchpoint 7)
	001	CPU1 data address compare 4 (watchpoint 7)
	010	Reserved
	011	Reserved
	100	Reserved
	101	CPU1 write to IMEMCTL0
	110	Reserved
	111	Reserved
MUX 28	000	CPU0 data address compare 1 (watchpoint 4)
	001	CPU1 data address compare 1 (watchpoint 4)
	010	Reserved
	011	Reserved
	100	Reserved
	101	CPU1 write to DMEMCTL0
	110	Reserved
	111	Reserved
MUX 29	000	CPU0 data address compare 2 (watchpoint 5)
	001	CPU1 data address compare 2 (watchpoint 5)
	010	Reserved
	011	Reserved
	100	Reserved
	101	CPU1 write to E2ECTL0
	110	Reserved
	111	Reserved

Table continues on the next page...

Table 63-11. L1SEL3 register field descriptions (continued)

Field	Description	
MUX 30	000	CPU0 data address compare 3 (watchpoint 6)
	001	CPU1 data address compare 3 (watchpoint 6)
	010	Reserved
	011	Reserved
	100	Reserved
	101	CPU1 write to L1CSR1
	110	Reserved
	111	Reserved
MUX 31	000	CPU0 data address compare 4 (watchpoint 7)
	001	CPU1 data address compare 4 (watchpoint 7)
	010	Reserved
	011	Reserved
	100	Reserved
	101	CPU1 write to L1CSR0
	110	Reserved
	111	Reserved

The L1SEL4 register fields are described in the following table.

Table 63-12. L1SEL4 register field descriptions

Field	Description	
MUX 32	000	CPU0 performance monitor counter 0 (watchpoint 23)
	001	CPU1 performance monitor counter 0 (watchpoint 23)
	010	CPU0 memory protection unit (watchpoint 27)
	011	CPU0 TRAP (watchpoint 28)
	100	Reserved
	101	Reserved
	110	Production device NAR full
	111	Reserved
MUX 33	000	CPU0 performance monitor counter 1 (watchpoint 24)
	001	CPU1 performance monitor counter 1 (watchpoint 24)
	010	CPU1 memory protection unit (watchpoint 27)
	011	CPU1 TRAP (watchpoint 28)
	100	Reserved
	101	Reserved
	110	Production device NAR partial full
	111	Reserved

Table continues on the next page...

Table 63-12. L1SEL4 register field descriptions (continued)

Field	Description	
MUX 34	000	CPU0 performance monitor counter 2 (watchpoint 25)
	001	CPU1 performance monitor counter 2 (watchpoint 25)
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	Buddy device NAR partial full
	111	Reserved
MUX 35	000	CPU0 performance monitor counter 3 (watchpoint 26)
	001	CPU1 performance monitor counter 3 (watchpoint 26)
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	Buddy device NAR full
	111	Reserved
MUX 36	000	Reserved
	001	Reserved
	010	CPU0 performance monitor event input (watchpoint 22)
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	Reserved
MUX 37	000	Reserved
	001	Reserved
	010	CPU1 performance monitor event input (watchpoint 22)
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	Reserved

Table continues on the next page...

Table 63-12. L1SEL4 register field descriptions (continued)

Field	Description	
MUX 38	000	Reserved
	001	Reserved
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	Reserved
MUX 39	000	Reserved
	001	Reserved
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	Reserved

The L1SEL5 register fields are described in the following table.

Table 63-13. L1SEL5 register field descriptions

Field	Description	
MUX 40	000	CPU0 interrupt (watchpoint 16)
	001	CPU0 critical interrupt (watchpoint 18)
	010	CPU0 data trace control range 1 (watchpoint 29)
	011	Reserved
	100	Reserved
	101	Reserved
	110	SPU counter event 0
	111	Reserved
MUX 41	000	CPU1 interrupt (watchpoint 16)
	001	CPU1 critical interrupt (watchpoint 18)
	010	CPU1 data trace control range 1 (watchpoint 29)
	011	Reserved
	100	Reserved
	101	Reserved
	110	SPU counter event 1
	111	Reserved

Table continues on the next page...

Table 63-13. L1SEL5 register field descriptions (continued)

Field	Description	
MUX 42	000	Reserved
	001	Reserved
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	SPU counter event 2
	111	Reserved
MUX43	000	Reserved
	001	Reserved
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	SPU counter event 3
	111	Reserved
MUX 44	000	CPU0 return (watchpoint 17)
	001	CPU0 critical return (watchpoint 19)
	010	CPU0 data trace control range 2 (watchpoint 30)
	011	Reserved
	100	Reserved
	101	Reserved
	110	SPU counter event 4
	111	Reserved
MUX 45	000	CPU1 return (watchpoint 17)
	001	CPU1 critical return (watchpoint 19)
	010	CPU1 data trace control range 2 (watchpoint 30)
	011	Reserved
	100	Reserved
	101	Reserved
	110	SPU counter event 5
	111	Reserved

Table continues on the next page...

Table 63-13. L1SEL5 register field descriptions (continued)

Field	Description	
MUX 46	000	Reserved
	001	Reserved
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	SPU counter event 6
	111	Reserved
MUX 47	000	Reserved
	001	Reserved
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	SPU counter event 7
	111	Reserved

The L1SEL6 register fields are described in the following table.

Table 63-14. L1SEL6 register field descriptions

Field	Description	
MUX 48	000	CPU0 context load (using e_lmvsprw instruction)
	001	CPU0 context restore (using e_stmvsprw instruction)
	010	CPU0 data trace control range 3 (watchpoint 31)
	011	NXMC0 watchpoint 0
	100	Reserved
	101	Reserved
	110	SPU counter event 8
	111	Reserved
MUX 49	000	CPU1 context load (using e_lmvsprw instruction)
	001	CPU1 context restore (using e_stmvsprw instruction)
	010	CPU1 data trace control range 3 (watchpoint 31)
	011	NXMC0 watchpoint 1
	100	Reserved
	101	Reserved
	110	SPU counter event 9
	111	Reserved

Table continues on the next page...

Table 63-14. L1SEL6 register field descriptions (continued)

Field	Description	
MUX 50	000	Reserved
	001	Reserved
	010	Reserved
	011	NXMC0 watchpoint 2
	100	Reserved
	101	Reserved
	110	SPU counter event 10
	111	Reserved
MUX 51	000	Reserved
	001	Reserved
	010	Reserved
	011	NXMC0 watchpoint 3
	100	Reserved
	101	Reserved
	110	SPU counter event 11
	111	Reserved
MUX 52	000	CPU0 interrupt current priority match to C0PIS
	001	Reserved
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	SPU counter event 12
	111	Reserved
MUX 53	000	CPU1 interrupt current priority match to C1PIS
	001	Reserved
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	SPU counter event 13
	111	Reserved

Table continues on the next page...

Table 63-14. L1SEL6 register field descriptions (continued)

Field	Description	
MUX 54	000	Reserved
	001	Reserved
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	SPU counter event 14
	111	Reserved
MUX 55	000	Reserved
	001	Reserved
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	SPU counter event 15
	111	Reserved

The L1SEL7 register fields are described in the following table.

Table 63-15. L1SEL7 register field descriptions

Field	Description	
MUX 56	000	Reserved
	001	Reserved
	010	NXMC1 watchpoint 0
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	DCI EVTI0
MUX 57	000	Reserved
	001	Reserved
	010	NXMC1 watchpoint 1
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	DCI EVTI1

Table continues on the next page...

Table 63-15. L1SEL7 register field descriptions (continued)

Field	Description	
MUX 58	000	Reserved
	001	Reserved
	010	NXMC1 watchpoint2
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	DCI EVTO0
MUX 59	000	Reserved
	001	Reserved
	010	NXMC1 watchpoint 3
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	DCI EVTO1
MUX 60	000	CPU0 processor exception vector match to C0PEVP
	001	Reserved
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	DCI system halt
MUX 61	000	CPU1 processor exception vector match to C1PEVP
	001	Reserved
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	Reserved

Table continues on the next page...

Table 63-15. L1SEL7 register field descriptions (continued)

Field	Description	
MUX 62	000	Reserved
	001	Reserved
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	Reserved
MUX 63	000	Reserved
	001	Reserved
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	Reserved

63.4 Calibration interface

This device includes software enabled debugging and calibration mode. Apart from the JTAG sources, such as a JTAG-based tool, it is possible to generate JTAG sequences through software programming. To support a software debug session, the software can enable debug and calibration mode via an memory-mapped register in the JTAGM. Software based debug is not possible when a tool is connected. The presence of the tool is determined using the rising edge of the JCOMP pad to set an internal flag which, when set, disables the selection of JTAGM as a JTAG source.

Table 63-16. Reference links to related information

Related module	Reference
JTAG Master (JTAGM)	JTAG Master (JTAGM)
Development Tool Semaphore (DTS)	Development Tool Semaphore (DTS)
Flash memory controller	Flash memory controller

63.4.1 JTAG Master (JTAGM)

The JTAG Master (JTAGM) acts as JTAG master inside the device. The module has a parallel interface that can exchange data with another serial communication module or through software.

The data transferred to this module is transformed to produce TCK, TMS, TDI, and TRST outputs and to accept TDO inputs. The module allows these five signals to connect to the JTAGC as if the JTAG data were coming from an external tool. The JTAGM generates all required JTAG scan chains to allow software and high-speed serial communication access to all JTAG-mapped resources.

63.4.2 Development Trigger Semaphore (DTS)

The Development Trigger Semaphore (DTS) module enables software to signal an external tool by driving a persistent (affected only by reset or an external tool) signal on an external device pin. There are a variety of ways that this module can be used, including as a component of an external real-time data acquisition system.

Note

When used as a component of a triggered data-acquisition system, Nexus read/write access is via the JTAG interface of the Nexus debug port and is different than the data acquisition protocol defined in the IEEE-ISTO 5001-2003 or IEEE-ISTO 5001-2012 Nexus standards, which use the Nexus auxiliary port.

63.5 Flash calibration support

The flash memory controller supports calibration development by providing a remapping function to route flash accesses to overlay RAM. The calibration remap function supports the following overlay RAM targets:

- On-chip (PD) overlay RAM
- Extended off-chip (ED) overlay RAM
- Portion of the PD system RAM

63.6 Nexus Trace Aurora interface

The Aurora interface allows Nexus protocol information to be transmitted serially at high speed over multiple Aurora lanes to provide advanced trace information from the device. The Aurora protocol handles the encoding of the data and the striping of information across the lanes. The Aurora interface is only available on the Emulation and Debug Device. Trace information is transferred to the Buddy device for transmission via the Aurora interface. The PD only supports trace to the internal trace/calibration memory.

The Aurora information can be used to reconstruct events or operations that occurred inside the microcontroller. Nexus supports the transmission, through a single Aurora port, of information from multiple trace clients within the MCU. Each Nexus message is tagged with the source client identifier and the type of message. The actual trace information that is available depends on the Nexus client; generally, it can be individually enabled (both the type of message and the client). Table 63-17 shows a few examples of Nexus messages. All messages can be enabled to include a timestamp, with the exception of the messages from the eTPU.

Table 63-17. Nexus message types

Message Type	Description
Program trace	Nexus program trace messages transmit any discontinuities in the program flow, such as branches and interrupts. Multiple types of messages can be generated.
Ownership trace	Ownership trace provides information on process identification changes.
Watchpoint trace	Watchpoint messages are generated any time a watchpoint match occurs in the program or data flow, or as a result of SPU actions. These can be programmed for many types of events within the MCU.
Device identification	The device identification message allows information about the MCU to be transmitted upon startup to allow tools to identify the target system MCU type.
Debug status	Debug status messages provide additional information that may be needed to reconstruct software execution, such as whether the device has entered Debug or Low-power mode.
Data acquisition	Data acquisition messages are an optional feature that allows software control of information to be transmitted.
Error	Error messages are transmitted when an error condition occurs, such as internal buffer overflows.

The Nexus trace information can be output to internal memory, or the parallel information can be serialized for transmission over a serial interface through the Nexus Aurora interface.

The Nexus blocks do not have memory-mapped registers. The Nexus registers are accessed by the development tool via the JTAG port using a two step process. First, the specific block is selected by loading the corresponding ACCESS_AUX instruction as described in the JTAGC chapter. Next, after the block is selected, the Nexus registers are enabled by loading the enable instruction to the JTAGC Instruction Register (IR). The

enable instructions for the different Nexus clients are shown in the following table. After the enable instruction is received, the development tool has access to the Nexus registers of the selected client.

Table 63-18. Nexus client JTAG enable instructions

Module	Enable instruction	Opcode
Production Die		
Cores	CORE_ENABLE	0x7C (10 bits)
NXMC	NEXUS_ENABLE	0x0 (4 bits)
NAR		
SPU		
eTPU		
Buddy Die		
RWA	CORE_ENABLE	0x7C (10 bits)
NAR	NEXUS_ENABLE	0x0 (4 bits)
NAL		

All of the internal JTAG modules are accessed through a single JTAG connection to the device. In the case of the ED, all JTAG communication is performed through the JTAG interface located in the BD. In the production package configuration, access is performed through the JTAG interface located in the PD.

With the hierarchical JTAG configuration, the BD JTAGC is considered the master JTAG TAP controller. The PD JTAGC TAP controller is selected via an AUX_ACCESS instruction. Other TAP controllers residing on the BD are also selected via AUX_ACCESS instructions (NAR, RWA, BD_DCI). Once control is transferred from the BD JTAGC to the PD JTAGC via the AUX_ACCESS instruction, all other PD TAP controllers are then selected via AUX_ACCESS instructions executed by the PD JTAGC. This is also referenced as TAP-sharing and is shown in the following figure.

NOTE

On the Emulation device, when the PAUSE-DR/UPDATE-DR state is executed to pass control of the JTAG TAP controller, control is always passed to the top level JTAG TAP controller, in other words, the DCI of the BD. For this reason, it is recommended that all JTAG operations, except when the BD clients need to be accessed, that all PD JTAG access be preceded with ACCESS_PD_JTAGC command. If the device is not an ED, the access is ignored.

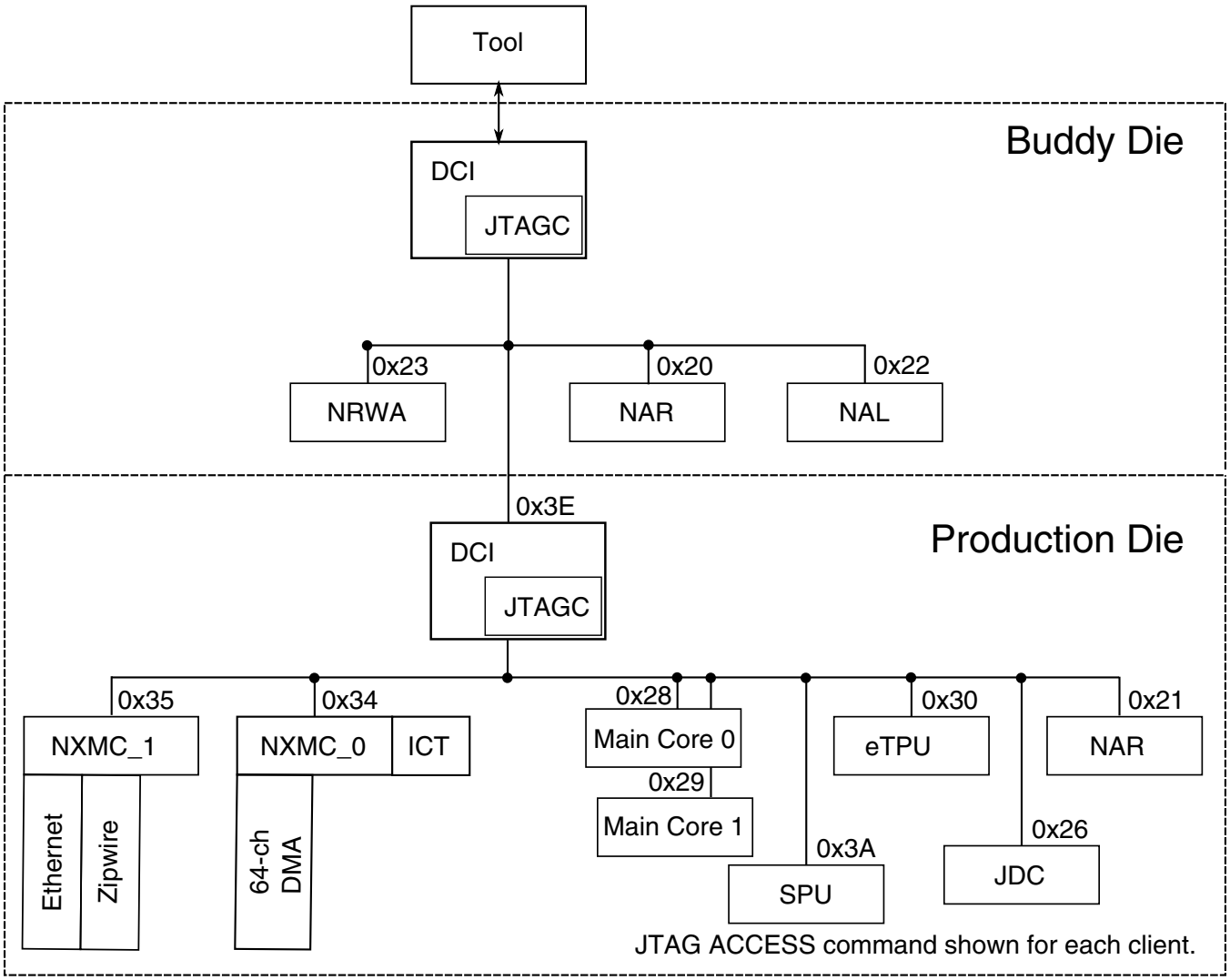


Figure 63-1. JTAG hierarchal view

Table 63-19. Reference links to related information

Related module	Reference
Production Device Nexus Aurora Router (NAR)	Production Device Nexus Aurora Router (NAR)

63.6.1 Nexus Aurora overview

The IEEE-ISTO 5001-2011 standard adds a new type of auxiliary trace port that supports a higher bandwidth. This new interface type uses the Aurora protocol to reformat the parallel Nexus messages into one or more output serial lanes of data, where a lane is defined as a 2-pin Low Voltage Differential Signals (LVDS) interface. The protocol handles striping the data onto the multiple lanes. Each lane uses an 8b/10b encoded LVDS driver. The encoding allows the receiver to recover clocking information.

The Aurora physical interface is based on the XAUI (10 Gigabit Attachment Unit Interface) standard used in communication protocols, which covers the jitter, differential skew, inter-lane skew, and bit error rate requirements.

The Aurora protocol covers lane initialization and channel bonding for both simplex and duplex implementations. The current implementation uses a simplex output-only connection from the MCU to the external tool.

The Aurora interface connects Nexus parallel trace output to an Aurora Protocol Engine (APE), the Aurora Lane Control (ALC), and the Nexus Aurora Physical interface (NAP). Figure 63-2 shows the Aurora interface block diagram, where the number of lanes, n, is 4.

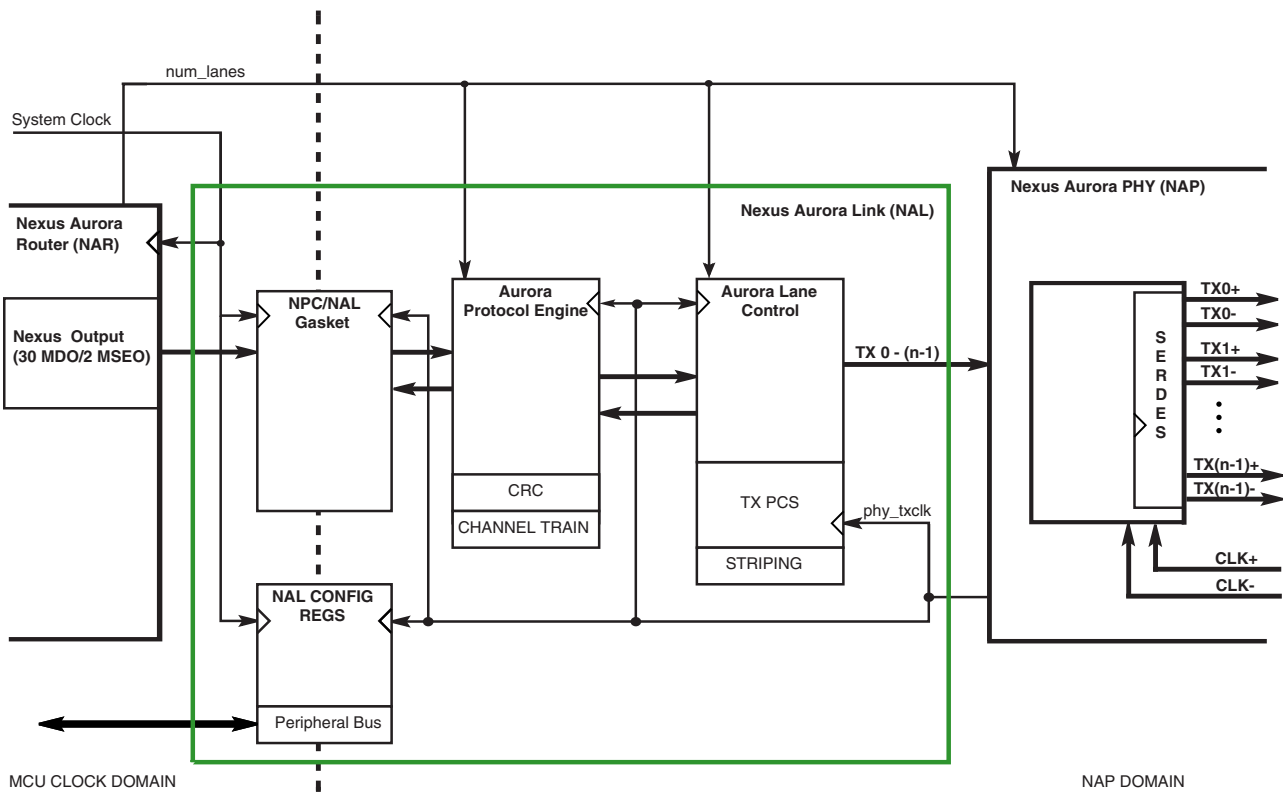


Figure 63-2. Aurora interface block diagram

63.6.2 Nexus Aurora Router (NAR)

The Nexus Aurora Router (NAR) manages the Nexus trace information from each of the Nexus clients. The NAR takes the parallel trace information from each of the connected clients, buffers it to avoid overruns, and routes it to either the internal overlay RAM or the Buddy Device, where it is stored in on-chip SRAM or transmitted out of the device through the Aurora interface.

16 KB of trace SRAM is available on the PD (located in the flash module), and 1 MB of trace SRAM is available in the ED. (Some of this SRAM may be used for calibration overlay content as well as trace content.) Similarly, a smaller NAR output queue is located on the PD (2 KB), and a much larger NAR output queue is located in the ED (8 KB). The physical Aurora pins are only available on the ED. Figure 63-3 shows the NAR block diagram.

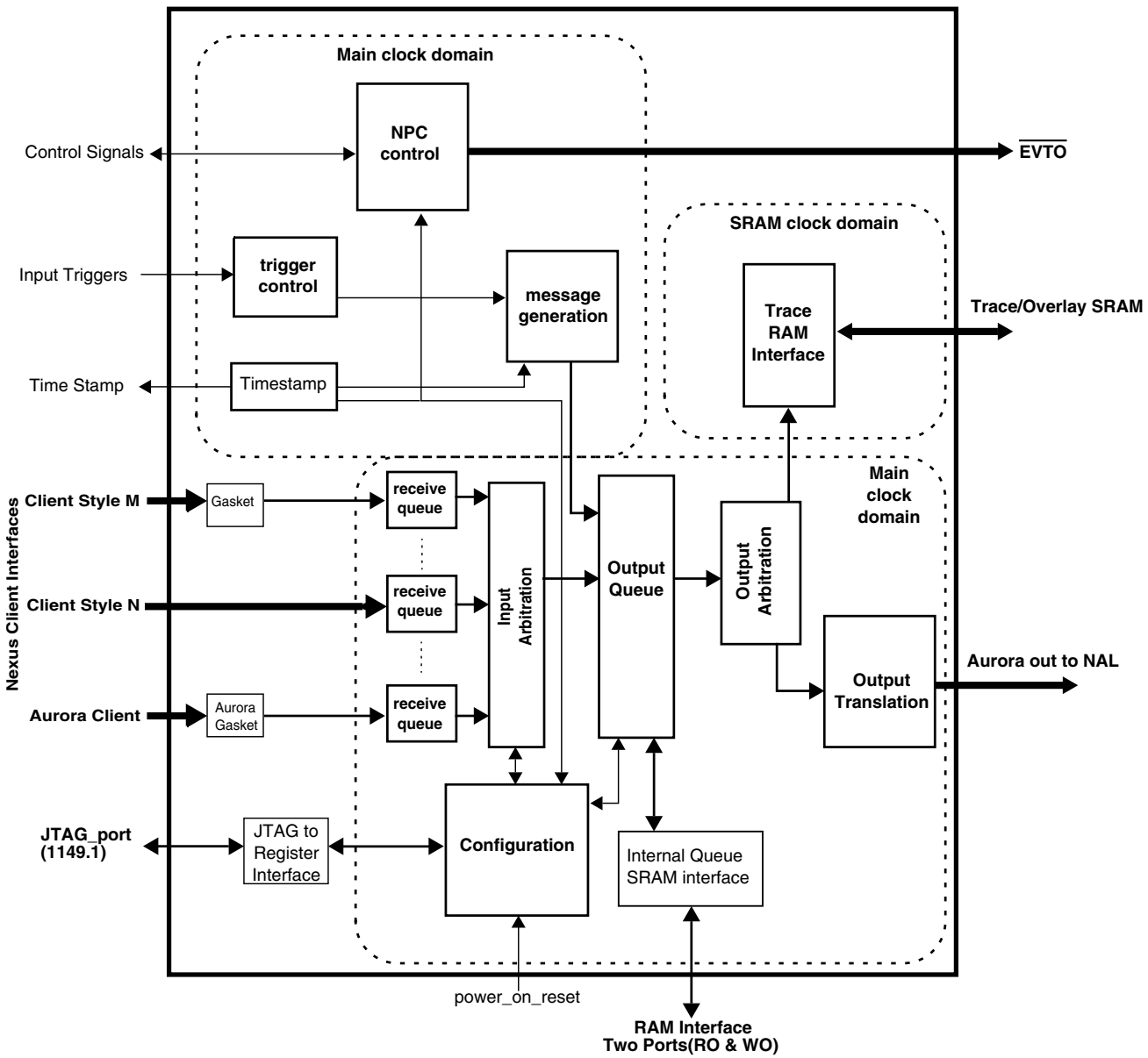


Figure 63-3. NAR block diagram

63.6.2.1 NAR clients

The PD NAR module has seven clients, with client index 1–7. Clients 8–16 are not connected. The client connections and the corresponding NAR client index are shown in the following table. The NAR client index is used in the NAR_STCR, NAR_CSSR, and NAR_CDR registers. The source trace ID is used in the NAR_SFR.

Table 63-20. PD NAR clients

Client index	Client	Nexus source trace ID (SRC)	Beats per client (Bn)
—	NAR	0b1111 (0xF)	—
1	—	—	—
2	Core 0	0b0000 (0x0)	1
3	Core 1	0b0001 (0x1)	1
4	NXMC Client 1 Concentrator	0b1101 (0xD)	1
5	NXMC Client 0 Concentrator	0b1100 (0xC)	1
6	eTPUA	0b0100 (0x4)	1
	eTPUB	0b0101 (0x5)	1
	eTPU_CDC_AB	0b0110 (0x6)	1
7	—	—	—
8	—	—	—
9	—	—	—
10	—	—	—
11	—	—	—
12	—	—	—
13	—	—	—
14	—	—	—
15	—	—	—
16	—	—	—

The BD NAR module has one client and the client connections and the corresponding NAR client index are shown in the following table.

Table 63-21. BD NAR clients

Client index	Client	Nexus source trace ID (SRC)	Beats per client (Bn)
1	PD NAR	0b1110 (0xE)	2

The NAR is limited writing trace information to the overlay RAM. The PD NAR only has access to the 16 KB internal overlay RAM. Therefore, the PD NAR trace memory bus base address registers (NAR_TBALO and NAR_TBAHI) must specify a base address in the range 0x0D000000–0x0D003FFF. The BD NAR also has access to 1 MB

of extended overlay RAM. Therefore, the BD NAR trace memory bus base address registers (NAR_TBALO and NAR_TBAHI) must specify a base address in the range 0x0C00_0000 to 0x0C0F_FFFF.

The NUM_LANES field of the NAR_NAPCR is described in the following table.

Table 63-22. NAR_NAPCR field descriptions

Field	Description	
19–16 NUM_LANE	TX Lane Configuration. Number of enabled TX lanes.	
	0000	Disable NAP and NAL
	0001	Reserved
	0010	Enable 2 lanes (Lane 0–1)
	0011	Reserved
	0100	Enable 4 lanes (Lanes 0–3)
	0101	Reserved
	0110	Reserved
	0111	Reserved
	1000	Reserved
	1001–1111	Reserved

63.6.3 Nexus Aurora Link (NAL)

The Nexus Aurora Link (NAL) is only included in the Emulation and Debug Device. The Nexus Aurora Link (NAL) consists of all the logic on the MCU needed to implement the connection up to the physical-layer serializer and transmitter (no receiver). It includes Aurora control and data multiplexing, data-encoding and striping, and the Physical Coding Sublayer (PCS) portion of the protocol.

Debug data in the Nexus format from multiple clients is packetized into a 32-bit format by the clients and then collected by the NAR. The NAL encodes the message data (8b/10b), stripes it across the lanes, and inserts Aurora control characters (such as start of message, end of message, idle, clock compensation characters, and so on) around the payload. The NAL PCS drives the Nexus Aurora Physical (NAP) interface.

63.6.4 Nexus Aurora PHY (NAP)

The Nexus Aurora PHY (NAP) is only included in the Emulation and Debug Device. The NAP, in conjunction with the Nexus Aurora Link (NAL), supports one-way simplex high-speed serial communication with an external debugger. The NAP receives 8b/10b encoded data from the ALC block in the NAL and transmits this data onto a given PHY

lane's LVDS pads. The NAL performs lane striping with settings in the NAL and NAR specifying lane configuration. Each NAP lane consists of a Data Symbol Character Accumulator that registers the 10-bit character for that lane from the ALC on the rising edge of the NAL clock. A serializer takes the 10-bit character and serializes it into the individual transmit data bits at the PHY clock rate to be transmitted via the LVDS output drivers to the external debugger receive channel.

Note

The NAL performs static link training for an attached debugger's PHY. The NAP is not directly aware of the link training operation, since it merely transmits the data presented to it by the NAL.

On the receive data side of the external debugger, it is the responsibility of the external debugger to perform elastic buffering, symbol detection, symbol locking (called "lane alignment" in Aurora terminology), 10b/8b decoding, decode and disparity error tracking, and CRC checking in the received data stream.

Figure 63-4 shows a block diagram of the Nexus Aurora Physical (NAP) interface and the LVDS output drivers, where the number of lanes, n, is 4.

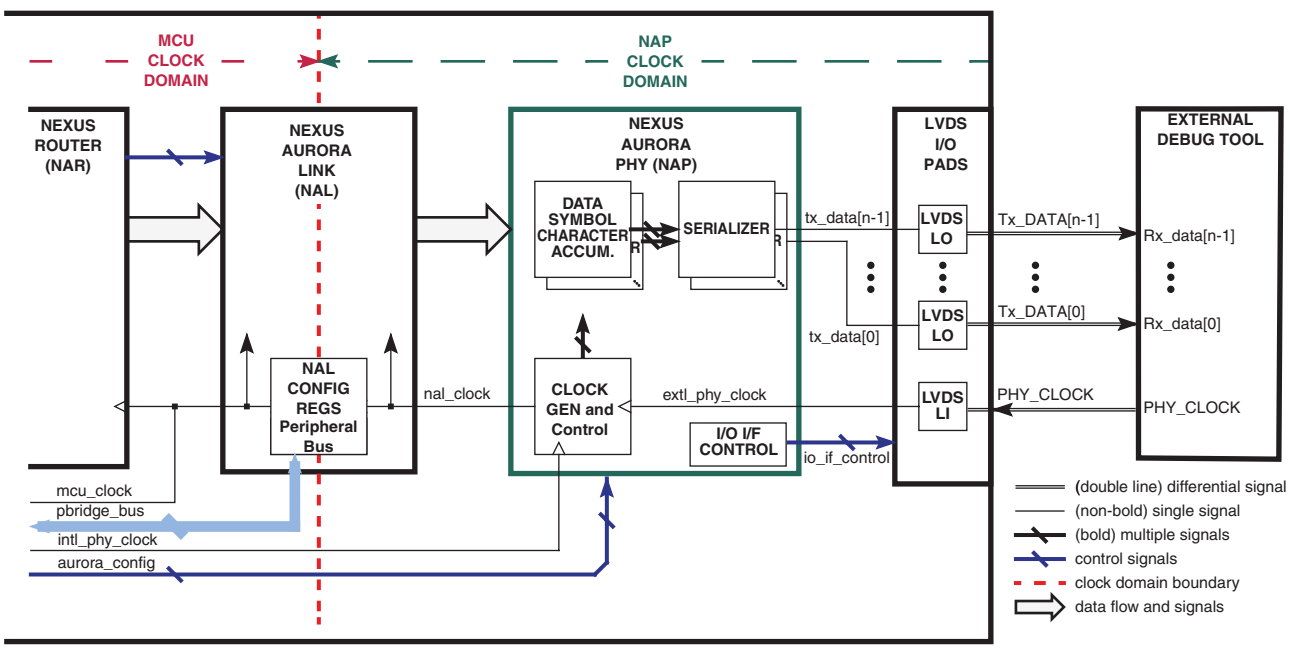


Figure 63-4. NAP block diagram

63.7 Nexus clients

Nexus supports the transmission, through a single Aurora port, of information from multiple trace clients within the device.

Table 63-23. Reference links to related information

Related module	Reference
Enhanced Time Processing Unit (eTPU)	Enhanced Time Processing Unit (eTPU)
e200z4 Nexus Support	Nexus Module
Nexus Crossbar Multi-Master Client (NXMC)	Nexus Crossbar Multi-Master Client (NXMC)

63.7.1 Enhanced Time Processing Unit (eTPU)

eTPU is an intelligent, semi-autonomous co-processor designed for timing control. Operating in parallel with the Host CPU, the eTPU processes instructions and real-time input events, performs output waveform generation, and accesses shared data without Host intervention. Consequently, for each timer event, the Host CPU setup and service times are minimized or eliminated.

The Nexus ETPU Development Interface (NDEDI) provides real-time development capabilities for the eTPU2 system in compliance with the IEEE-ISTO 5001-2002 standard. The NDEDI interfaces to the eTPU via dedicated trace busses. The main development features supported are:

- Breakpoint/watchpoint configuration
- Debug mode register access
- Enter debug mode at reset negation or during normal execution
- Single stepping of instructions
- Branch trace
- Data trace
- Ownership trace
- Watchpoint trace

Combined, these features make the interface compliant with Class 3 of the IEEE-ISTO 5001-2002 standard.

63.7.2 e200z4 Nexus 3

The e200z425n3 Nexus 3 modules provide real-time development capabilities for e200z4d processors in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 3 module also supports two Class 4 features: Watchpoint Triggering and Processor Overrun Control.

63.7.3 Nexus Crossbar Multi-master Client (NXMC)

The Nexus Crossbar Multi-master Client (NXMC) module provides real-time trace capabilities in compliance with the IEEE-ISTO Nexus 5001-2011 standard. This module provides development support capabilities for SoCs without requiring address and data pins for internal visibility. A portion of the pin interface is also compliant with the IEEE 1149.1 JTAG standard.

63.7.3.1 NXMC block connections

To provide Nexus data trace messaging from bus masters on the master ports of the crossbar (XBAR) that do not inherently have a trace client, a Nexus Crossbar Multi-Master Client (NXMC) monitors traffic into the master port of the XBAR. Two NXMCs are implemented: one for the fast masters and one for the slow masters. Trace messages transmitted over the Nexus interface (or stored in trace memory) include which NXMC generated the message as well as an identifier for the pre-concentrator source of the message. In addition to XBAR bus master trace capability, the NXMC also allows for in-circuit trace capability that can be used to trace the Sequence Processing Unit (SPU) counter values. The two external hardware watchpoint triggers of each NXMC are used to generate timestamps (four total—two from each NXMC) for events from the SPU.

The following figure shows a block diagram illustrating the NXMC integration.

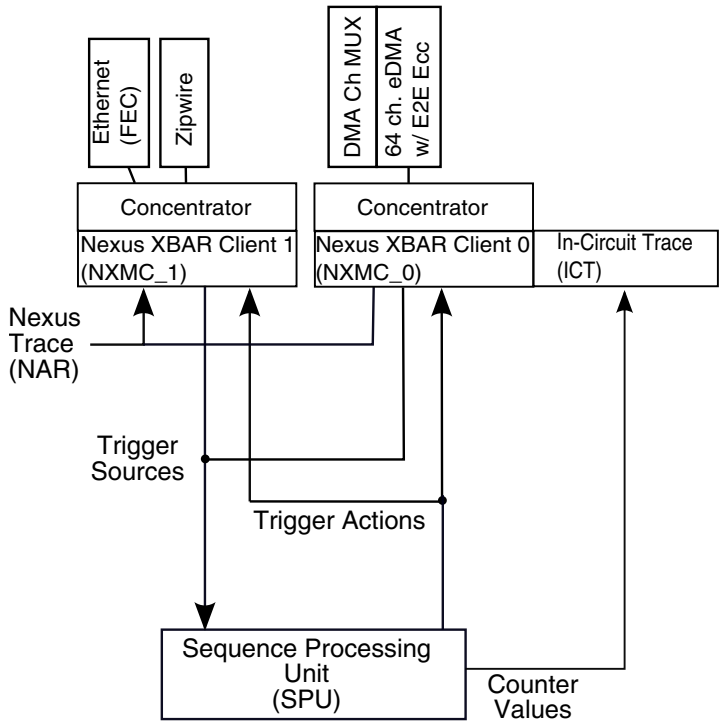


Figure 63-5. NXMC integration block diagram

The values of the trace master ID or in-circuit client ID, along with the Nexus message source identifier for the different clients are shown in the following table.

Table 63-24. NXMC source and master client IDs

Client		Nexus trace source ID (SRC)	Trace master ID or In-circuit trace client ID (MASTER)
In-Circuit Trace	SPU	0b1011	0b0000
XBAR Client 0 Concentrator	eDMA Controller	0b1100	0b0011
(fast masters)	Reserved		All other values
XBAR Client 1 Concentrator	Ethernet	0b1101	0b0100
(slow masters)	Zipwire		0b1000
	Reserved		All other values

Chapter 64
Core Debug Support

64.1 Overview

Internal debug support in the core allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes. For software based debugging, debug facilities consisting of a set of software accessible debug registers and interrupt mechanisms are provided. These facilities are also available to a hardware based debugger which communicates using a modified IEEE 1149.1 Test Access Port (TAP) controller and pin interface. When hardware debug is enabled, the debug facilities controlled by hardware are protected from software modification.

Software debug facilities are defined as part of *PowerISA 2.06*. supports a subset of these defined facilities. In addition to the facilities defined in *PowerISA 2.06*, provides additional flexibility and functionality in the form of additional instruction breakpoints, linked instruction and data breakpoints, and extended range and value masking. These features are also available to a hardware-based debugger.

When not being used for debugging purposes, a portion of the debug facilities may be configured to provide a stack limit checking mechanism.

64.1.1 Software Debug Facilities

provides debug facilities to enable hardware and software debug functions, such as instruction and data breakpoints and program single stepping. The debug facilities consist of a set of debug control registers (DBCR0–2,4–8, EDBRAC0), a set of address compare registers (IAC1–8, DAC1–4), a set of 64-bit data value compare registers (DVC1, DVC2), a Debug Status Register (DBSR) for enabling and recording various kinds of debug events, a Debug Data Effective Address Register (DDEAR), and a special Debug interrupt type built into the interrupt mechanism.

The debug facilities also provide a mechanism for software-controlled processor reset, and debug mode entry. In addition, the Performance Monitor may be configured to utilize the debug interrupt type. Also, the Memory Protection Unit (MPU) may be configured to generate debug events using region descriptors which are not utilized for performing a protection function.

Software debug facilities are enabled by setting the internal debug mode bit in Debug Control register 0 ($DBCR0_{IDM}$). When internal debug mode is enabled, debug events can occur, and can be enabled to record exceptions in the Debug Status register (DBSR). If enabled by MSR_{DE}, these recorded exceptions cause Debug interrupts to occur. When $DBCR0_{IDM}$ is cleared (and $EDBCR0_{EDM}$ is cleared as well), no debug events occur, and no status flags are set in DBSR unless already set. In addition, when $DBCR0_{IDM}$ is cleared (or is overridden by $EDBCR0_{EDM}$ being set and EDBRAC0 indicating no resource is "owned" by software) no Debug interrupts will occur, regardless of the contents of DBSR. A software Debug interrupt handler may access all system resources and perform necessary functions appropriate for system debug.

64.1.1.1 *PowerISA 2.06* Compatibility

The core implements a subset of the *PowerISA 2.06* internal debug features. The following restrictions on functionality are present:

- Instruction address compares do not support compare on physical (real) addresses.
- Data address compares do not support compare on physical (real) addresses.

64.1.2 Additional Debug Facilities

In addition to the debug functionality defined in *PowerISA 2.06*, provides capability to link instruction and data breakpoints, provides additional instruction and data breakpoints, extended range and value masking, multiple watchpoint outputs, provides capability for the Performance Monitor to generate debug events, and allows for sharing of debug resources between software and a hardware debugger. (See Sharing Debug Resources by Software/Hardware.) A data trace port is also provided.

64.1.2.1 Data Trace Port

The data trace port interface is provided to assist in implementing extended debug watchpoint/breakpoint/trace capture capability with logic external to the core. This port provides information corresponding to each read or write access completed without error

by the CPU. In order to report data accesses, the Nexus 3 DTC register DI control bit must be set to trace data accesses, not instruction accesses (i.e. the normal setting. See the "Data Trace Control Register (DTC)" section in the Nexus 3 Module chapter. The data trace port will report aligned accesses, and most misaligned accesses as one access, unless the two portions of a misaligned access are non-contiguous, such as when the second portion of the misaligned access is to address 0. Also, for accesses generated by a LSP load/store instruction using circular addressing mode which crosses a buffer boundary, the two portions of the misaligned access will be reported separately, since otherwise it is not possible to correctly correlate the data with the initial access address.

64.1.3 Hardware Debug Facilities

The core contains facilities that allow for external test and debugging. A modified IEEE 1149.1 control interface is used to communicate with the core resources. This interface is implemented through a standard 1149.1 TAP (test access port) controller.

By using public instructions, the external debugger can freeze or halt the core, read and write internal state and debug facilities, single-step instructions, and resume normal execution.

Hardware Debug is enabled by setting the External Debug Mode enable bit in External Debug Control register 0 ($EDBCR0_{EDM}$), which is also aliased to $DBCR0_{EDM}$. Setting $EDBCR0_{EDM}$ overrides the Internal Debug Mode enable bit $DBCR0_{IDM}$ unless resources are provided back to software via the settings in EDBRAC0. When the Hardware Debug facility is enabled, software is blocked from modifying the "hardware-owned" debug facilities. In addition, since resources are "owned" by the hardware debugger, inconsistent values may be present if software attempts to read "hardware-owned" debug-related resources.

When hardware debug is enabled by setting $EDBCR0_{EDM}$=1, the control registers and resources described in Debug registers are reserved for use by the external debugger. The same events described in Software Debug Events and Exceptions are also used for external debugging, but exceptions are not generated to running software. Hardware-owned debug events enabled in the respective DBCR0–8 registers are recorded in the EDBSR0 register (not the DBSR) regardless of MSR_{DE}, and no debug interrupts are generated unless a) the resource is granted back to software via EDBRAC0 settings, and b) debug mode entry is not masked by the corresponding event bit in EDBSRMSK0. Instead, the CPU will enter debug mode when an enabled event causes a EDBSR0 bit to become set. $EDBCR0_{EDM}$, EDBSR0, EDBSRMSK0, EDDEAR, and EDBRAC0 may only be written through the OnCE port.

Access to most debug resources (registers) requires that the CPU clock (**m_clk**) be running in order to perform write accesses from the external hardware debugger.

64.1.4 Sharing Debug Resources by Software/Hardware

Debug resources may be shared by a hardware debugger and software debug based on the settings of debug control register EDBRAC0. When $EDBCR0_{EDM}$ is set, EDBRAC0 settings determine which debug resources are allocated to software and which resources remain under exclusive hardware control. Software-owned resources which set DBSR bits when $DBCR0_{IDM}$=1 will cause a debug interrupt to occur when enabled with MSR_{DE}. Hardware-owned resources which set EDBSR0 bits when $EDBCR0_{EDM}$=1 will cause an entry into debug mode if the event is not masked in EDBSRMSK0. EDBRAC0 is read-only by software. When resource sharing is enabled, ($EDBCR0_{EDM}$=1 and $EDBRAC0_{IDM}$=1), only software-owned resources may be modified by software. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Hardware-owned resources will set status bits in the EDBSR0 register instead of in DBSR, and will report the effective address of data breakpoints in EDDEAR instead of DDEAR. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers are performed.

64.1.4.1 Simultaneous Hardware and Software Debug Event Handing

Since it is possible that a "hardware-owned" resource can produce a debug event in conjunction with a software-owned resource producing a different debug event simultaneously, a priority ordering mechanism is implemented which guarantees that the hardware event is handled as soon as possible, while preserving the recognition of the software event. The CPU will give highest priority to the software event initially in order to reach a recoverable boundary, and then will give highest priority to the hardware event in order to enter debug mode as near the point of event occurrence as possible. This is implemented by allowing software exception handing to begin internal to the CPU and to reach the point where the current program counter and MSR values have been saved into DSRR0/1, and the new PC pointing to the debug interrupt handler, along with the new MSR updates. At this point, hardware priority takes over, and the CPU enters debug mode.

The following figure shows the debug resources.

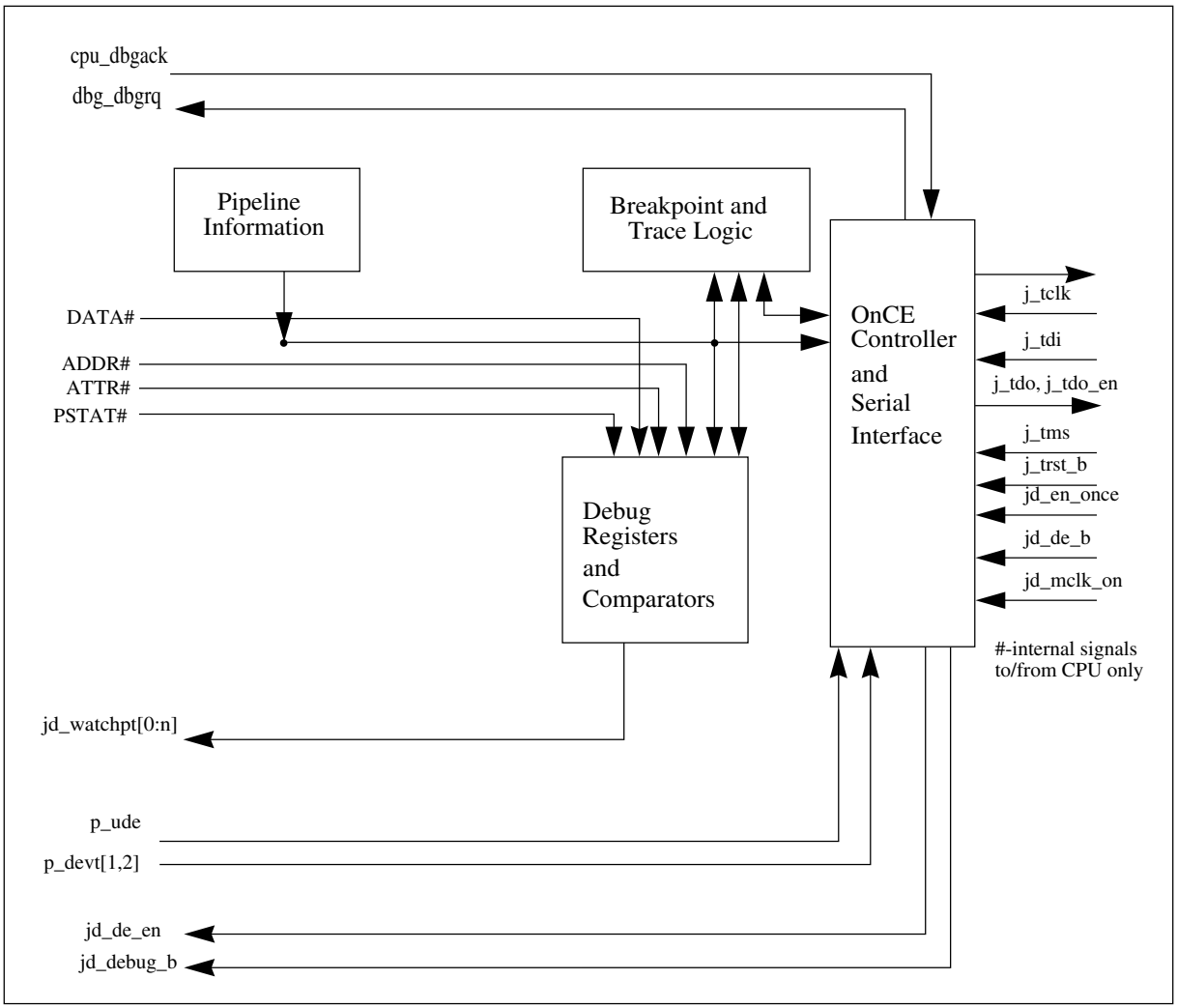


Figure 64-1. Debug Resources

64.2 Software Debug Events and Exceptions

Software debug events and exceptions are available when internal debug mode is enabled ($DBCR0_{IDM}$=1) and not overridden by external debug mode ($EDBCR0_{EDM}$ must either be cleared or corresponding resources must be allocated to software debug by the settings in EDBRAC0). When enabled, debug events cause debug exceptions to be recorded in the Debug Status Register. Specific event types are enabled by the Debug Control Registers (DBCR0–8). The Unconditional Debug Event (UDE) is an exception to this rule; it is always enabled. Once a Debug Status Register (DBSR) bit is set by a debug resource which is "owned" by software (other than MRR, DAC_OFST, or VLES), if Debug interrupts are enabled by MSR_{DE}, a Debug interrupt will be generated. The debug interrupt handler is responsible for ensuring that multiple repeated debug interrupts do not occur by clearing the DBSR as appropriate.

Certain debug events are not allowed to occur when MSR_{DE}=0 and $DBCR0_{IDM}$=1. In such situations, no debug exception occurs and thus no DBSR bit is set. Other debug events may cause debug exceptions and set DBSR bits regardless of the state of MSR_{DE}. A Debug interrupt will be delayed until MSR_{DE} is later set to '1'.

When a Debug Status Register bit is set while MSR_{DE}=0, an Imprecise Debug Event flag ($DBSR_{IDE}$) will also be set to indicate that an exception bit in the Debug Status Register was set while Debug interrupts were disabled. Debug interrupt handler software can use this bit to determine whether the address recorded in Debug Save/Restore Register 0 is an address associated with the instruction causing the debug exception, or the address of the instruction which enabled a delayed Debug interrupt by setting the MSR_{DE} bit. A **mtmsr** or **mtdbcr0** which causes both MSR_{DE} and $DBCR0_{IDM}$ to become set, enabling precise debug mode, may cause an Imprecise (Delayed) Debug exception to be generated due to an earlier recorded event in the Debug Status register.

There are eight types of debug events defined by *PowerISA 2.06*:

1. Instruction Address Compare debug events
2. Data Address Compare debug events
3. Trap debug events
4. Branch Taken debug events
5. Instruction Complete debug events
6. Interrupt/Critical Interrupt Taken debug events
7. Return/Critical Return debug events
8. Unconditional debug events

In addition, defines additional debug events:

- The External debug events DEVT1 and DEVT2 which are described in External debug event.
- The Performance Monitor Interrupt event PMI which is described in Performance Monitor Interrupt debug event.

The debug configuration supports most of these event types. Unsupported *PowerISA 2.06* defined functionality is as follows:

- Instruction Address Compare and Data Address Compare *Real address* mode is not supported

A brief description of each of the event types follows.

64.2.1 Instruction Address Compare Event

Instruction Address Compare debug events occur when enabled and execution is attempted of an instruction at an address that meets the criteria specified in the DBCR0, DBCR1, DBCR5, DBCR6, and IAC1–8 Registers. Instruction Address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. IAC events will not occur when an instruction would not have normally begun execution due to a higher priority exception at an instruction boundary.

IAC compares perform a 31-bit compare for VLE instruction storage accesses. Each halfword fetched by the instruction fetch unit will be marked with a set of bits indicating whether an Instruction Address Compare occurred on that halfword. Debug exceptions will occur if enabled and a 16-bit instruction, or the first halfword of a 32-bit instruction, is tagged with an IAC hit.

64.2.2 Data Address Compare Event

Data Address Compare debug events occur when enabled and execution of a load or store class instruction results in a data access that meets the criteria specified in the DBCR0, DBCR2, DBCR4, DBCR7–8, DAC1–4, DVC1, and DVC2 Registers. Data address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. Four address compare values (DAC1–4) are provided.

The effective address of the load or store operation is recorded in the DDEAR register when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set.

Note

In contrast to the *PowerISA 2.06* definition, Data Address Compare events on do not prevent the load or store class instruction from completing. If a load or store class instruction

completes successfully without a Data Storage interrupt, Data Address Compare exceptions are reported at the completion of the instruction. If the exception results in a precise Debug interrupt, the address value saved in DSRR0 is the address of the instruction following the load or store class instruction. For DVC DAC events, the exception can be imprecisely reported even further past the load or store class instruction generating the event (without necessarily affecting $DBSR_{IDE}$) and the saved address value can point to a subsequent instruction past the next instruction. This occurrence is indicated in the $DBSR_{DAC_OFST}$ field.

If a load or store class instruction does not complete successfully due to a Data Storage exception or a machine check condition for the load or store, and a Data Address Compare debug exception also occurs, the result is an imprecise Debug interrupt, the address value saved in DSRR0 is the address of the load or store class instruction, and the $DBSR_{IDE}$ bit will be set. In addition to occurring when $DBCR0_{IDM}$=1, this circumstance can also occur when $EDBCR0_{EDM}$=1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

Note

DAC events will not be recorded if a load multiple word or store multiple word instruction is interrupted prior to completion by a critical input or external input interrupt.

Note

DAC events are not signaled on the second portion of a misaligned load or store that is broken up into two separate accesses.

Note

DAC events are not signaled on the **mpure** or **mpuwe** MPU instructions.

Note

DAC[1,2] events are not signaled if DVC[1,2]M is non-zero and a DSI exception occurs on the load or store, since the load or store access is not performed. For a **lmw** or **stmw** transfer

however, if a DVC successfully occurs on a transfer and a later transfer encounters a DSI exception, the DAC event will be reported, since a successful data value compare took place.

Note

Due to internal pipeline status, for a reported DAC event on a **lmw** or **stmw** transfer, the value stored in the DDEAR/EDDEAR will be the effective address of the first transfer of the sequence, and not necessarily the precise transfer that caused the DAC event.

64.2.2.1 Data Address Compare Event Status Updates

Data Address Compare debug events with Data Value compares can be reported ambiguously in several circumstances involving issuing a sequence of load or store class instructions. Due to the CPU pipeline and the delay in performing the data value compare following completion of the access, if the first load or store class instruction generates a DVC DAC, a second and possibly third load or store class instruction may also generate a DAC or DVC DAC event, or may generate a DSI exception with or without a simultaneous DAC event.

Also, since non-load/store instructions may be dual-issued in combination with a load/store instruction, the actual number of additional instructions that are completed following a recognized DVC DAC on a load/store instruction may vary from 0 to 5. This value will be reported in the $DBSR_{DAC_OFST}$ field when the DVC DAC status is recorded.

Table 64-1 outlines the settings of the DBSR, DSRR0 saved value, and potential updating of the ESR register for various exception cases on sequences of load/store class instructions. Not all exception combinations are covered in the table, such as IAC, ISI, or Alignment exceptions on subsequent instructions. In general these exceptions will cause further instruction issue to be halted, execution of the excepting instruction to be aborted, and reporting of these exceptions will be masked. The saved DSRR0 value will point to this excepting instruction, and the exception(s) may be regenerated after returning from the debug interrupt handler and attempting to re-execute the instruction pointed to by DSRR0. In addition, in the examples in Table 64-1, the DAC_OFST and DSRR0 values assume no dual issue occurs. If dual-issue occurs with the first, second, or third column, then the DAC_OFST and DSRR0 values will point beyond the values shown.

Table 64-1. DAC events and Resultant Updates

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DSI, no DAC	—	—	Take DSI exception, no DBSR update. Update ESR.
DSI, with DACx	—	—	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST not set. DSRR0 points to 1st load/store class instruction. No ESR update.
DACx	—	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update.
DVC DACx	No exceptions, any instruction	No exceptions, Non-ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. No ESR update.
DVC DACx	No exceptions	No exceptions, Ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b010. DSRR0 points to instruction after 3rd instruction. No ESR update.
DVC DACx	DSI, no DAC	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DSI, with DACy	—	Take Debug exception, DBSR update setting DACx. DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DACy	—	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Normal Ldst	Non-Ldst instruction	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Normal Ldst	Ldst instruction,no exception	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Normal Ldst	DSI Error, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. No ESR update. DSRR0 points to 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.

Table continues on the next page...

Table 64-1. DAC events and Resultant Updates (continued)

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DVC DACx	DVC DACy,Normal Ldst	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Ldst multiple (lmw, stmw)	Any instruction including ld/st	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	Any instruction(no exception)	DSI, with or without DAC,Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. No ESR update. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	Any instruction(no exception)	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 64-2 through Table 64-5 show some example updates for specific code sequences of dual issuing of load/store class instructions with non-load/store class instructions and the results of DAC and DVC events on selected ones of the load/store instructions.

Table 64-2. DAC events and Resultant Updates, Dual-issue case 1

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6)
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table continues on the next page...

Table 64-2. DAC events and Resultant Updates, Dual-issue case 1 (continued)

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (5): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (5): DACy or DVC DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. No ESR update. DSRR0 points to instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 64-3. DAC events and Resultant Updates, Dual-issue case 2

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) load/store	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b101. DSRR0 points to instruction after instruction (6). No ESR update.
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b101. DSRR0 points to instruction (7). No ESR update. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table continues on the next page...

Table 64-3. DAC events and Resultant Updates, Dual-issue case 2 (continued)

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) load/store	Event(s)	Result
	Instruction (1): DVC DACx Instruction (3): DVC DACyInstruction (6): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instruction (4). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (6): DACy or DVC DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b101. No ESR update. DSRR0 points to instruction (7). No debug event updates for instruction (7). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 64-4. DAC events and Resultant Updates, Dual-issue case 3

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) alu (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6).
	Instruction (1): DVC DACx Instruction (5): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (5): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No debug event updates for instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (5): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6) **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 64-5. DAC events and Resultant Updates, Dual-issue case 4

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) alu	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6)
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

64.2.3 Linked Instruction Address Compare and Data Address Compare Events

Data Address Compare 1, 2, 3, and 4 debug events may be 'linked' with an Instruction Address Compare event by setting the DAC[1–4]LNK control bits in DBCR2 and DBCR8 to further refine when a Data Address Compare debug event is generated. DAC1

may be linked with IAC1, DAC2 (when not used as a mask or range bounds register) may be linked with IAC3. DAC3 may be linked with IAC5, DAC4 (when not used as a mask or range bounds register) may be linked with IAC7. When linked, a DAC debug event occurs when the same instruction that generates the DAC 'hit' also generates a corresponding linked IAC 'hit'. When linked, the IAC event is not recorded in the Debug Status register, regardless of whether a corresponding linked DAC event occurs, or whether the IAC event enable is set.

When enabled and execution of a load or store class instruction results in a data access with an address that meets the criteria specified in the DBCRx, DACx, and DVCx Registers, and the instruction also meets the criteria for generating an Instruction Address Compare event, a Linked Data Address Compare debug event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. The normal DAC1 status bit in the DBSR is used for recording these events. The IAC status bit is not set if the corresponding Instruction Address Compare register is linked.

Linking is enabled using control bits in DBCR2 and DBCR8. Note that linking is only available in EDM or IDM. Attempts to use linking otherwise are ignored.

Note

Linked DAC events will not be recorded if a load multiple word or store multiple word type instruction is interrupted prior to completion by a critical input or external input interrupt.

64.2.4 Trap Debug Event

A Trap debug event (TRAP) occurs if Trap debug events are enabled ($DBCR0_{TRAP}$=1), a Trap instruction (**tw**) is executed, and the conditions specified by the instruction for the trap are met. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Trap debug event occurs, the $DBSR_{TRAP}$ bit is set to 1 to record the debug exception.

64.2.5 Branch Taken Debug Event

A Branch Taken debug event (BRT) occurs if Branch Taken debug events are enabled ($DBCR0_{BRT}$=1) and execution is attempted of a branch instruction that will be taken (either an unconditional branch, or a conditional branch whose branch condition is true), and MSR_{DE}=1. Branch Taken debug events are not recognized if MSR_{DE}=0 at the time of execution of the branch instruction and thus $DBSR_{IDE}$ can not be set by a Branch Taken

debug event. When a Branch Taken debug event is recognized, the $DBSR_{BRT}$ bit is set to 1 to record the debug exception, and the address of the branch instruction will be recorded in DSRR0.

64.2.6 Instruction Complete Debug Event

An Instruction Complete debug event (ICMP) occurs if Instruction Complete debug events are enabled ($DBCR0_{ICMP}$=1), execution of any instruction is completed, and MSR_{DE}=1. If execution of an instruction is suppressed due to the instruction causing some other exception that is enabled to generate an interrupt, then the attempted execution of that instruction does not cause an Instruction Complete debug event. The ***se_sc*** instruction does not fall into the category of an instruction whose execution is suppressed, since the instruction actually executes and then generates a System Call interrupt. In this case, the Instruction Complete debug exception will also be set. When an Instruction Complete debug event is recognized, $DBSR_{ICMP}$ is set to 1 to record the debug exception and the address of the next instruction to be executed will be recorded in DSRR0.

Instruction Complete debug events are not recognized if MSR_{DE}=0 at the time of execution of the instruction, thus $DBSR_{IDE}$ is not generally set by an ICMP debug event.

One circumstance may cause the $DBSR_{ICMP}$ and $DBSR_{IDE}$ bits to be set. This occurs when a EFPU Round exception occurs. Since the instruction is by definition completed (SRR0 points to the following instruction), this interrupt takes higher priority than the Debug interrupt so as not to be lost, and $DBSR_{IDE}$ is set to indicate the imprecise recognition of a Debug interrupt. In this case, the Debug interrupt will be taken with SRR0 pointing to the instruction following the instruction that generated the EFPU Round exception, and DSRR0 will point to the Round exception handler. In addition to occurring when $DBCR0_{IDM}$=1, this circumstance can also occur when $EDBCR0_{EDM}$=1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

Note

Instruction complete debug events are not generated by the execution of an instruction that sets MSR_{DE} to '1' while $DBCR0_{ICMP}$=1, nor by the execution of an instruction that sets $DBCR0_{ICMP}$ to '1' while MSR_{DE}=1.

64.2.7 Interrupt Taken Debug Event

An Interrupt Taken debug event (IRPT) occurs if Interrupt Taken debug events are enabled ($DBCR0_{IRPT}$=1) and a base-class interrupt occurs. Only base-class interrupts (an interrupt using SRR0/1) cause an Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Interrupt Taken debug event occurs, the $DBSR_{IRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the non-critical interrupt handler.

64.2.8 Critical Interrupt Taken Debug Event

A Critical Interrupt Taken debug event (CIRPT) occurs if Critical Interrupt Taken debug events are enabled ($DBCR0_{CIRPT}$=1) and a critical interrupt occurs. Only critical class interrupts (an interrupt using CSRR0/1) cause a Critical Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Interrupt Taken debug event occurs, the $DBSR_{CIRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the critical interrupt handler.

64.2.9 Return Debug Event

A Return debug event (RET) occurs if Return debug events are enabled ($DBCR0_{RET}$=1) and an attempt is made to execute an **se_rfi** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Return debug event occurs, the $DBSR_{RET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfi** (i.e. before the MSR is updated by the **se_rfi**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfi**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfi** instruction.

64.2.10 Critical Return Debug Event

A Critical Return debug event (CRET) occurs if Critical Return debug events are enabled ($DBCR0_{CRET}$=1) and an attempt is made to execute an **se_rfci** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Return debug event occurs, the $DBSR_{CRET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfci** (i.e. before the MSR is updated by the **se_rfci**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfci**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfci** instruction.

64.2.11 External debug event

An External debug event (DEVT1, DEVT2) occurs if External debug events are enabled ($DBCR0_{DEVT1}$=1 or $DBCR0_{DEVT2}$=1), and the respective **p_devt1** or **p_devt2** input signal transitions to the asserted state while the CPU is not in the Stopped state. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an External debug event occurs, $DBSR_{DEVT\{1,2\}}$ is set to '1' to record the debug exception. This debug event is an asynchronous event, but is only sampled when the CPU **m_clk** is active.

64.2.12 Unconditional debug event

An Unconditional debug event (UDE) occurs when the Unconditional Debug Event (**p_ude**) input transitions to the asserted state. The Unconditional debug event is the only debug event that does not have a corresponding enable bit for the event in DBCR0. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Unconditional debug event occurs, the $DBSR_{UDE}$ bit is set to '1' to record the debug exception. This debug event is an asynchronous event.

64.2.13 Performance Monitor Interrupt debug event

A Performance Monitor Interrupt debug event (PMI) occurs if Performance Monitor Interrupt debug events are enabled ($PMGC0_{UDI}$=1), and a performance monitor interrupt event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Performance Monitor Interrupt debug event occurs, $DBSR_{PMI}$ is set to '1' to record the debug exception. This debug event is an asynchronous event.

64.3 Debug registers

This section describes debug-related registers that are software accessible. These registers are intended for use by special debug tools and debug software, not by general application code.

Access to these registers (other than DBSR) by software is conditioned by the External Debug mode control bit ($EDBCR0_{EDM}$) and the settings of debug control register EDBRAC0, which can be set by the hardware debug port. If $EDBCR0_{EDM}$ is set and if the bit in EDBRAC0 corresponding to the resource is cleared, software is prevented from modifying debug register values other than in DBSR, since the resource is not "owned" by software. Software always has ownership of DBSR. Execution of a **mtspr** instruction targeting a debug register or register field not "owned" by software will not cause modifications to occur, and no exception will be signaled. In addition, since the external debugger hardware may be manipulating debug register values, the state of these registers or register fields not "owned" by software is not guaranteed to be consistent if accessed (read) by software with a **mfspr** instruction, other than the $DBCR0_{EDM}$ bit itself and the EDBRAC0 register. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control registers as appropriate when hardware modifications to the debug registers is performed.

64.3.1 Debug Address and Value Registers

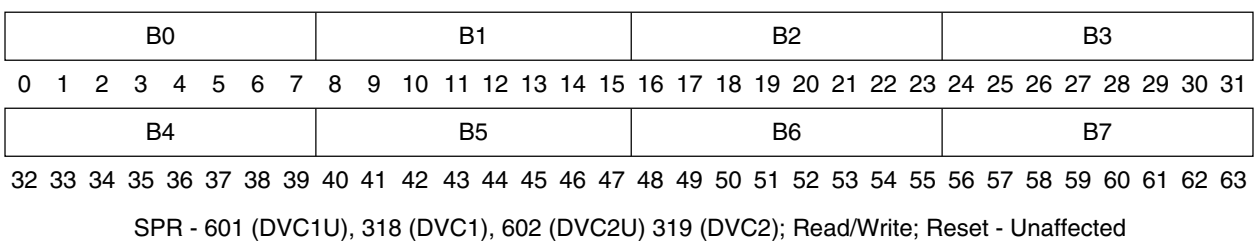
Instruction Address Compare registers IAC1–8 are used to hold instruction addresses for address comparison purposes. In addition, IAC2 and IAC4 may hold mask information for IAC1 and IAC3 respectively and IAC6 and IAC8 may hold mask information for IAC5 and IAC7 respectively, when *Address Bit Match* compare modes are selected. Note that when performing instruction address compares, the low order bit of the instruction address and the corresponding IAC register is ignored for VLE instructions.

Data Address Compare registers DAC1–4 are used to hold data access addresses for address comparison purposes. In addition, DAC2 and DAC4 may hold mask information for DAC1 and DAC3 respectively when *Address Bit Match* compare modes are selected.

Data Value Compare registers DVC1 and DVC2 are used to hold data values for data comparison purposes. DVC1 and DVC2 are 64-bit registers. Data value comparisons are used to qualify Data Address compare 1 and 2 debug events. Data value comparisons are

not available for Data address compare 3–4 events. DVC1 is associated with DAC1, and DVC2 is associated with DAC2. The most significant byte of the DVC1(2) register (labeled B0 in the following figure) corresponds to the byte data value transferred to/from memory byte offset 0, 8, …, and the least significant byte of the register (labeled B7 in the following figure) corresponds to byte offset 7, F, … . When enabled for performing data value comparisons, each enabled byte in DVC1(2) is compared with the memory value transferred on the corresponding active byte lane of the data memory interface to determine if a match occurs. Inactive byte lanes do not participate in the comparison, they are implicitly masked. The Byte Strobe Assertion for Transfers table in the Core Complex Overview chapter shows active byte lanes for data transfers. Software must also program the DVC1(2) register byte positions based on the alignment of the access. Misaligned accesses are not fully supported, since the data address and data value comparisons are only performed on the initial access in the case of a misaligned access; thus, accesses that cross a 64-bit boundary cannot be fully matched. For address and size combinations that involve two transfers, only the initial transfer is used for data address and value matching.

DVC1 and DVC2 may be read or written using **mtspr** and **mfspr** instructions. The DVC1U and DVC2U SPR numbers (601,602) are used for accessing the upper 32-bit portion of the DVC register. The DVC1 and DVC2 SPR numbers (318,319) are used for accessing the lower 32-bit portion of the DVC register.



SPR - 601 (DVC1U), 318 (DVC1), 602 (DVC2U) 319 (DVC2); Read/Write; Reset - Unaffected

Figure 64-2. DVC1, DVC2 registers

64.3.2 Debug Control and Status registers

Debug Control Registers (DBCR0–8 and EDBRAC0) are used to enable debug events, reset the processor, and set the debug mode of the processor. The Debug Status register (DBSR) records debug exceptions while Internal Debug mode is enabled. The Debug Data Effective Address register (DDEAR) records the effective address of a Data Address Compare event while Internal Debug mode is enabled.

requires that a context synchronizing instruction follow a **mtspr** DBCR0–8 or DBSR to ensure that any alterations enabling/disabling debug events are effective. The context synchronizing instruction may or may not be affected by the alteration. Typically, an **se_isync** instruction is used to create a synchronization boundary beyond which it can be guaranteed that the newly written control values are in effect.

For watchpoint generation, configuration settings contained in DBCR1–8 are used, even though the corresponding event(s) may be disabled (via DBCR0) from setting DBSR flags.

64.3.2.1 Debug Control Register 0 (DBCR0)

Debug Control Register 0 is used to enable debug modes and controls which debug events are allowed to set DBSR or EDBSR0 flags. adds some implementation specific bits to this register, as seen in the following figure.

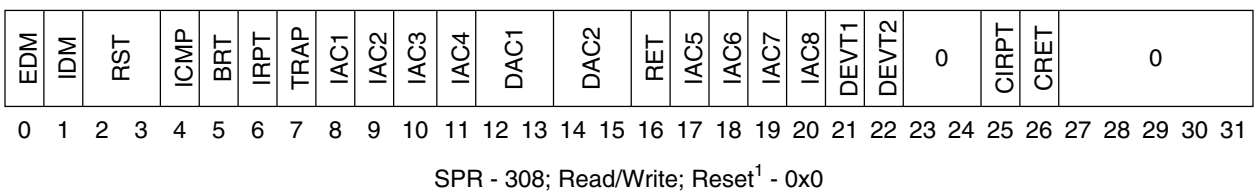


Figure 64-3. Debug Control Register 0 (DBCR0) register

Note

[1] $DBCR0_{EDM}$ is affected by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b**. All other bits are reset by processor reset **p_reset_b** if $DBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $DBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources (other than RST) from reset by **p_reset_b**, and only software-owned resources indicated by EDBRAC0 and the $DBCR0_{RST}$ field will be reset by **p_reset_b**. The $DBCR0_{RST}$ field will always be reset by **p_reset_b** regardless of the value of $DBCR0_{EDM}$.

The following provides bit definitions for Debug Control Register 0.

Table 64-6. DBCR0 field descriptions

Bit	Name	Description
0	EDM	External Debug mode. This bit is read-only by software. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset p_reset_b. This allows the debugger to set up hardware debug events that remain active across a processor reset.
		0 External debug mode disabled. Internal debug events not mapped into external debug events. 1 External debug mode enabled. Hardware-owned debug events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCR0–8, IAC1–8, DAC1–4, DVC1–2[U]} unless permitted by settings in EDBRAC0. Hardware-owned events will set status bits in EDBSR0. Programming Notes: It is recommended that debug status bits in the Debug Status Registers be cleared before disabling external debug mode to avoid any internal imprecise debug interrupts. Software may use this bit to determine if external debug has control over the debug registers. The hardware debugger must set the EDM bit to '1' before other bits in this register (and other debug registers) may be altered. On the initial setting of this bit to '1', all other bits are unchanged. This bit is only writable through the OnCE port.
1	IDM	Internal Debug mode 0 Debug exceptions are disabled. Debug events do not affect DBSR. 1 Debug exceptions are enabled. Enabled debug events owned by software will update the DBSR. If MSR_{DE}=1, the occurrence of a debug event, or the recording of an earlier debug event in the Debug Status Register when MSR_{DE} was cleared, will cause a Debug interrupt.
2:3	RST	Reset Control 00 No function 01 **p_dbrstc[1]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset 10 **p_dbrstc[0]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset. 11 Reserved
4	ICMP	Instruction Complete Debug Event Enable 0 ICMP debug events are disabled 1 ICMP debug events are enabled
5	BRT	Branch Taken Debug Event Enable 0 BRT debug events are disabled 1 BRT debug events are enabled
6	IRPT	Interrupt Taken Debug Event Enable 0 IRPT debug events are disabled 1 IRPT debug events are enabled
7	TRAP	Trap Taken Debug Event Enable 0 TRAP debug events are disabled 1 TRAP debug events are enabled

Table continues on the next page...

Table 64-6. DBCR0 field descriptions (continued)

Bit	Name	Description
8	IAC1	Instruction Address Compare 1 Debug Event Enable 0 IAC1 debug events are disabled 1 IAC1 debug events are enabled
9	IAC2	Instruction Address Compare 2 Debug Event Enable 0 IAC2 debug events are disabled 1 IAC2 debug events are enabled
10	IAC3	Instruction Address Compare 3 Debug Event Enable 0 IAC3 debug events are disabled 1 IAC3 debug events are enabled
11	IAC4	Instruction Address Compare 4 Debug Event Enable 0 IAC4 debug events are disabled 1 IAC4 debug events are enabled
12:13	DAC1	Data Address Compare 1 Debug Event Enable 00 DAC1 debug events are disabled 01 DAC1 debug events are enabled only for store-type data storage accesses 10 DAC1 debug events are enabled only for load-type data storage accesses 11 DAC1 debug events are enabled for load-type or store-type data storage accesses
14:15	DAC2	Data Address Compare 2 Debug Event Enable 00 DAC2 debug events are disabled 01 DAC2 debug events are enabled only for store-type data storage accesses 10 DAC2 debug events are enabled only for load-type data storage accesses 11 DAC2 debug events are enabled for load-type or store-type data storage accesses
16	RET	Return Debug Event Enable 0 RET debug events are disabled 1 RET debug events are enabled
17	IAC5	Instruction Address Compare 5 Debug Event Enable 0 IAC5 debug events are disabled 1 IAC5 debug events are enabled
18	IAC6	Instruction Address Compare 6 Debug Event Enable 0 IAC6 debug events are disabled 1 IAC6 debug events are enabled
19	IAC7	Instruction Address Compare 7 Debug Event Enable 0 IAC7 debug events are disabled 1 IAC7 debug events are enabled
20	IAC8	Instruction Address Compare 8 Debug Event Enable 0 IAC8 debug events are disabled 1- IAC8 debug events are enabled

Table continues on the next page...

Table 64-6. DBCR0 field descriptions (continued)

Bit	Name	Description
21	DEVT1	External Debug Event 1 Enable 0 DEVT1 debug events are disabled 1 DEVT1 debug events are enabled
22	DEVT2	External Debug Event 2 Enable 0 DEVT2 debug events are disabled 1 DEVT2 debug events are enabled
23:24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event Enable 0 CIRPT debug events are disabled 1 CIRPT debug events are enabled
26	CRET	Critical Return Debug Event Enable 0 CRET debug events are disabled 1 CRET debug events are enabled
27:31	—	Reserved

64.3.2.2 Debug Control Register 1 (DBCR1)

Debug Control Register 1 is used to configure Instruction Address Compare operation. The DBCR1 register is shown in the following figure.

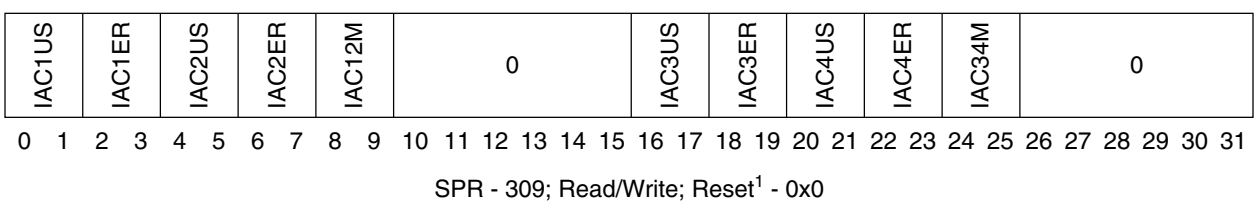


Figure 64-4. DBCR1 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 1.

Table 64-7. DBCR1 field descriptions

Bit	Name	Description
0:1	IAC1US	Instruction Address Compare 1 User/Supervisor Mode 00 IAC1 debug events not affected by MSR_{PR} 01 Reserved 10 IAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC1 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC1ER	Instruction Address Compare 1 Effective/Real Mode 00 IAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC2US	Instruction Address Compare 2 User/Supervisor Mode 00 IAC2 debug events not affected by MSR_{PR} 01 Reserved 10 IAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC2ER	Instruction Address Compare 2 Effective/Real Mode 00 IAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC12M	Instruction Address Compare 1/2 Mode 00 Exact address compare. IAC1 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC1. IAC2 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC2. 01 Address bit match. IAC1 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC2 are equal to the contents of IAC1, also ANDed with the contents of IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used. 10 Inclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC1 and less than the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used. 11 Exclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC1 or is greater than or equal to the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.
10:15	—	Reserved
16:17	IAC3US	Instruction Address Compare 3 User/Supervisor Mode 00 IAC3 debug events not affected by MSR_{PR} 01 Reserved 10 IAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC3 debug events can only occur if MSR_{PR}=1 (User mode)

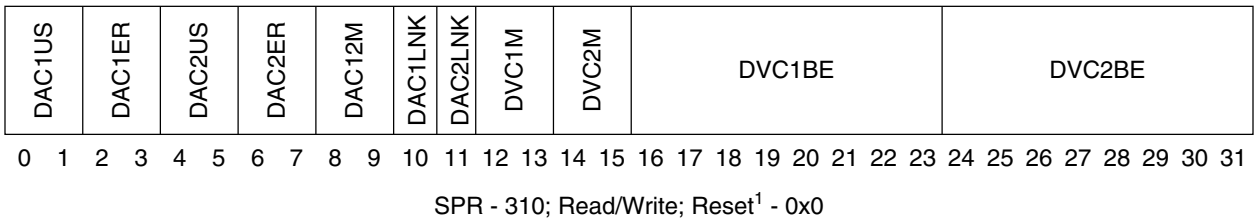
Table continues on the next page...

Table 64-7. DBCR1 field descriptions (continued)

Bit	Name	Description
18:19	IAC3ER	Instruction Address Compare 3 Effective/Real Mode 00 IAC3 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=1
20:21	IAC4US	Instruction Address Compare 4 User/Supervisor Mode 00 IAC4 debug events not affected by MSR_{PR} 01 Reserved 10 IAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC4 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC4ER	Instruction Address Compare 4 Effective/Real Mode 00 IAC4 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC34M	Instruction Address Compare 3/4 Mode 00 Exact address compare. IAC3 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC3. IAC4 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC4. 01 Address bit match. IAC3 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC4 are equal to the contents of IAC3, also ANDed with the contents of IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 10 Inclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC3 and less than the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 11 Exclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC3 or is greater than or equal to the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used.
26:31	—	Reserved

64.3.2.3 Debug Control Register 2 (DBCR2)

Debug Control Register 2 is used to configure Data Address Compare and Data Value Compare operation.The DBCR2 register is shown in the following figure.



SPR - 310; Read/Write; Reset[1] - 0x0

Figure 64-5. DBCR2 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 2.

Table 64-8. DBCR2 Bit Definitions

Bit	Name	Description
0:1	DAC1US	Data Address Compare 1 User/Supervisor Mode 00 DAC1 debug events not affected by MSR_{PR} 01 Reserved 10 DAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC1 debug events can only occur if MSR_{PR}=1 (User mode)
2:3	DAC1ER	Data Address Compare 1 Effective/Real Mode 00 DAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC2US	Data Address Compare 2 User/Supervisor Mode 00 DAC2 debug events not affected by MSR_{PR} 01 Reserved 10 DAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	DAC2ER	Data Address Compare 2 Effective/Real Mode 00 DAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=1

Table continues on the next page...

Table 64-8. DBCR2 Bit Definitions (continued)

Bit	Name	Description
8:9	DAC12M	Data Address Compare 1/2 Mode 00 Exact address compare. DAC1 debug events can only occur if the address of the data access is equal to the value specified in DAC1. DAC2 debug events can only occur if the address of the data access is equal to the value specified in DAC2. 01 Address bit match. DAC1 debug events can occur only if the address of the data access ANDed with the contents of DAC2, are equal to the contents of DAC1 also ANDed with the contents of DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 10 Inclusive address range compare. DAC1 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC1 and less than the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 11 Exclusive address range compare. DAC1 debug events can occur only if the address of the data access is less than the value specified in DAC1 or is greater than or equal to the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used.
10	DAC1LNK	Data Address Compare 1 Linked. When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC1 debug events are linked to IAC1 debug events. IAC1 debug events do not affect DBSR When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM.
11	DAC2LNK	Data Address Compare 2 Linked. When linked to IAC3, DAC2 debug events are conditioned based on whether the instruction also generated an IAC3 debug event. DAC2 can only be linked if DAC12M specifies Exact Address Compare since DAC2 debug events are not generated in the other compare modes. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC 2 debug events are linked to IAC3 debug events. IAC3 debug events do not affect DBSR

Table continues on the next page...

Table 64-8. DBCR2 Bit Definitions (continued)

Bit	Name	Description
12:13	DVC1M	Data Value Compare 1 Mode When $DBCR4_{DVC1C}$=0: 00 DAC1 debug events not affected by data value compares. 01 DAC1 debug events can only occur when all bytes specified in the DVC1BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC1 debug events can only occur when any byte specified in the DVC1BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC1 debug events can only occur when all bytes specified in the DVC1BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC1 value. **NOTE:** Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC1C}$=1: 00 Reserved 01 DAC1 debug events can only occur when any byte specified in the DVC1BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC1 debug events can only occur when all bytes specified in the DVC1BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **NOTE:** Note: Inactive byte lanes of the memory access are automatically masked.
14:15	DVC2M	Data Value Compare 2 Mode When $DBCR4_{DVC2C}$=0: 00 DAC2 debug events not affected by data value compares. 01 DAC2 debug events can only occur when all bytes specified in the DVC2BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC2 debug events can only occur when any byte specified in the DVC2BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC2 debug events can only occur when all bytes specified in the DVC2BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC2 value. **NOTE:** Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC2C}$=1: 00 Reserved 01 DAC2 debug events can only occur when any byte specified in the DVC2BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC2 debug events can only occur when all bytes specified in the DVC2BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **NOTE:** Inactive byte lanes of the memory access are automatically masked.

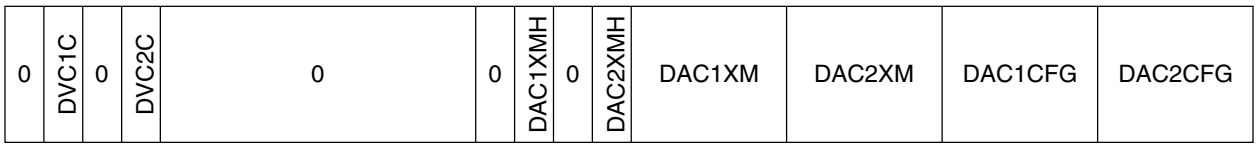
Table continues on the next page...

Table 64-8. DBCR2 Bit Definitions (continued)

Bit	Name	Description
16:23	DVC1BE	Data Value Compare 1 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC1. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1xxxxxxx Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC1. x1xxxxxx Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC1. xx1xxxxx Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC1. xxx1xxxx Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC1. xxxx1xxx Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC1. xxxxx1xx Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC1. xxxxxx1x Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC1. xxxxxxx1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC1.
24:31	DVC2BE	Data Value Compare2 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC2. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1xxxxxxx Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC2. x1xxxxxx Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC2. xx1xxxxx Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC2. xxx1xxxx Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC2. xxxx1xxx Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC2. xxxxx1xx Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC2. xxxxxx1x Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC2. xxxxxxx1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC2.

64.3.2.4 Debug Control Register 4 (DBCR4)

Debug Control Register 4 is used to extend data address and value compare matching functionality. DBCR4 is shown in the following figure.



SPR - 563; Read/Write; Reset[1] - 0x0

Figure 64-6. DBCR4 Register

Note

[1] DBCR4 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 4.

Table 64-9. DBCR4 field description

Bit	Name	Description
0	—	Reserved
1	DVC1C	Data Value Compare 1 Control. DVC1C controls whether DVC1 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition. 0 Normal DVC1 operation. Inverted polarity DVC1 operation
2	—	Reserved
3	DVC2C	Data Value Compare 2 Control. DVC2C controls whether DVC2 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition 0 Normal DVC2 operation. 1 Inverted polarity DVC2 operation
4:12	—	Reserved
13	DAC1XMH	Data Address Compare 1 Extended Mask Control High. DAC1XMH extends the range of the DAC1XM field. 0 - DAC1XM masks 0–15 low-order address bits 1 - DAC1XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC2XMH	Data Address Compare 2 Extended Mask Control High. . DAC2XMH extends the range of the DAC2XM field. 0 DAC2XM masks 0–15 low-order address bits 1 DAC2XM masks 16–31 low-order address bits

Table continues on the next page...

Table 64-9. DBCR4 field description (continued)

Bit	Name	Description
16:19	DAC1XM	Data Address Compare 1 Extended Mask Control. DAC1XM allows for binary power of 2 address range compares for DAC1 without requiring the use of DAC2. Value of DAC1XMH || DAC1XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC1 when comparing the storage address with the value in DAC1 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC2XM	Data Address Compare 2 Extended Mask Control Value of DAC2XMH || DAC2XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC2 when comparing the storage address with the value in DAC2 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported. DAC2XM allows for binary power of 2 address range compares for DAC2.
24:27	DAC1CFG	Data Address Compare 1 Configuration 0000 DAC1 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC1}$=00 0001 DAC1 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC1}$=00 0010 DAC1 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC1}$=00 0011 Reserved 01xx Reserved 1000 DAC1 address comparisons are used for stack limit checking. DAC1 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC1. When a qualified DAC1 match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC1 watchpoint is generated. $DBCR0_{DAC1}$ and $DBCR2_{DVC1M}$ settings are ignored. 1001 – 1111 Reserved DAC1CFG controls whether DAC1 data address comparisons utilize the normal PowerISA operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). **NOTE:** unlike the DAC3–4CFG fields, debug event enabling for DAC1 is controlled by the DAC1 field in DBCR0. The DAC1CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC1}$=0; when $DBCR0_{DAC1}$!=00, DAC1 watchpoints will fire whenever a DAC1 debug event occurs.

Table continues on the next page...

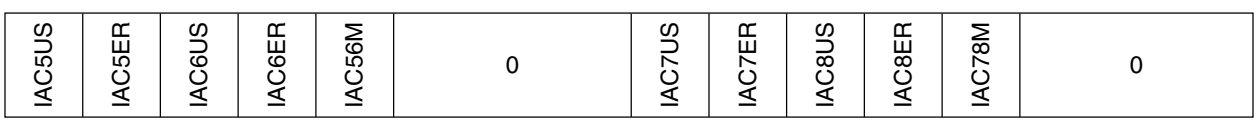
Table 64-9. DBCR4 field description (continued)

Bit	Name	Description
28:31	DAC2CFG	Data Address Compare 2 Configuration 0000 DAC2 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC2}$=00 0001 DAC2 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC2}$=00 0010 DAC2 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC2}$=00 0011 Reserved 01xx Reserved 1xxx Reserved DAC2CFG controls whether DAC2 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. **NOTE:** Unlike the DAC3–4CFG fields, debug event enabling for DAC2 is controlled by the DAC2 field in DBCR0. The DAC2CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC2}$=00. When $DBCR0_{DAC2}$!=00, DAC2 watchpoints will fire whenever a DAC2 debug event occurs.

64.3.2.5 Debug Control Register 5 (DBCR5)

Debug Control Register 5 is used to configure Instruction Address Compare operation for IAC5–8. The DBCR5 register is shown in the following figure.

.



SPR - 564; Read/Write; Reset[1] - 0x0

Figure 64-7. DBCR5 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 5.

Table 64-10. DBCR5 field descriptions

Bit(s)	Name	Description
0:1	IAC5US	Instruction Address Compare 5 User/Supervisor Mode 00 IAC5 debug events not affected by MSR_{PR} 01 Reserved 10 IAC5 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC5 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC5ER	Instruction Address Compare 5 Effective/Real Mode 00 IAC5 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC6US	Instruction Address Compare 6 User/Supervisor Mode 00 IAC6 debug events not affected by MSR_{PR} 01 Reserved 10 IAC6 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC6 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC6ER	Instruction Address Compare 6 Effective/Real Mode 00 IAC6 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC56M	Instruction Address Compare 5/6 Mode 00 Exact address compare. IAC5 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC5. IAC6 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC6. 01 Address bit match. IAC5 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC6 are equal to the contents of IAC5, also ANDed with the contents of IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 10 Inclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC5 and less than the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 11 Exclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC5 or is greater than or equal to the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used.
10:15	—	Reserved

Table continues on the next page...

Table 64-10. DBCR5 field descriptions (continued)

Bit(s)	Name	Description
16:17	IAC7US	Instruction Address Compare 7 User/Supervisor Mode 00 IAC7 debug events not affected by MSR_{PR} 01 Reserved 10 IAC7 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC7 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC7ER	Instruction Address Compare 7 Effective/Real Mode 00 IAC7 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=1
20:21	IAC8US	Instruction Address Compare 8 User/Supervisor Mode 00 IAC8 debug events not affected by MSR_{PR} 01 Reserved 10 IAC8 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC8 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC8ER	Instruction Address Compare 8 Effective/Real Mode 00 IAC8 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC78M	Instruction Address Compare 7/8 Mode 00 Exact address compare. IAC7 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC7. IAC8 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC8. 01 Address bit match. IAC7 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC8 are equal to the contents of IAC7, also ANDed with the contents of IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 10 Inclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC7 and less than the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 11 Exclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC7 or is greater than or equal to the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used.
26:31	—	Reserved

64.3.2.6 Debug Control Register 6 (DBCR6)

Debug Control Register 6 is used to extend instruction address compare matching functionality. DBCR6 is shown in the following figure.

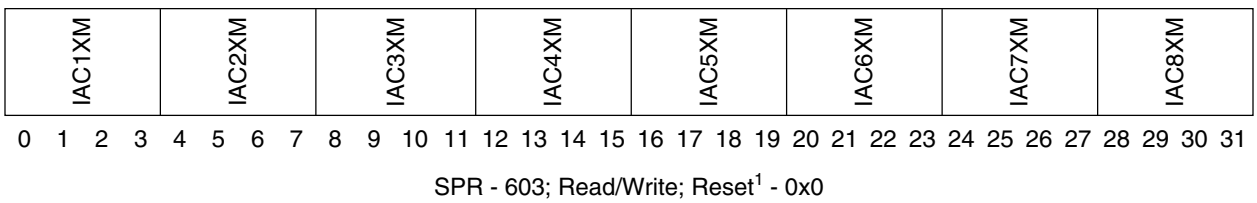


Figure 64-8. DBCR6 Register

Note

[1] DBCR6 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 6.

Table 64-11. DBCR6 field descriptions

Bit	Name	Description
0:3	IAC1XM	Instruction Address Compare 1 Extended Mask Control. IAC1XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC12M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC1 when comparing the storage address with the value in IAC1 for exact address compare (DBCR1[IAC12M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
4:7	IAC2XM	Instruction Address Compare 2 Extended Mask Control. IAC2XM allows for binary power of 2 address range compares for IAC2 without requiring the use of IAC1. 0000 No additional masking when DBCR1[IAC12M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC2 when comparing the storage address with the value in IAC2 for exact address compare (DBCR1[IAC12M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
8:11	IAC3XM	Instruction Address Compare 3 Extended Mask Control. IAC3XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC34M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC3 when comparing the storage address with the value in IAC3 for exact address compare (DBCR1[IAC34M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved

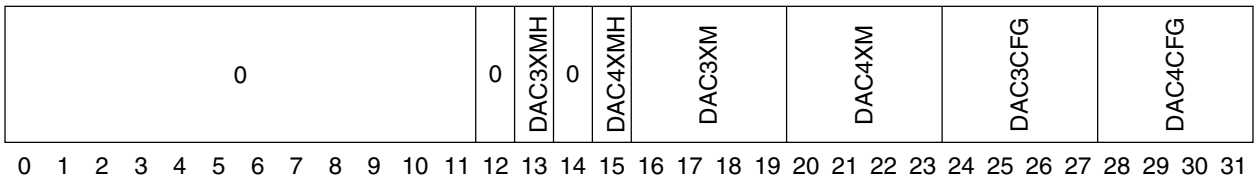
Table continues on the next page...

Table 64-11. DBCR6 field descriptions (continued)

Bit	Name	Description
12:15	IAC4XM	Instruction Address Compare 4 Extended Mask Controll. IAC4XM allows for binary power of 2 address range compares for IAC4 without requiring the use of IAC3. 0000 No additional masking when DBCR1[IAC34M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC4 when comparing the storage address with the value in IAC4 for exact address compare (DBCR1[IAC34M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
16:19	IAC5XM	Instruction Address Compare 5 Extended Mask Control. IAC5XM allows for binary power of 2 address range compares for IAC5 without requiring the use of IAC6. 0000 No additional masking when DBCR5[IAC56M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC5 when comparing the storage address with the value in IAC5 for exact address compare (DBCR5[IAC56M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
20:23	IAC6XM	Instruction Address Compare 6 Extended Mask Control. IAC6XM allows for binary power of 2 address range compares for IAC6 without requiring the use of IAC5. 0000 No additional masking when DBCR5[IAC56M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC6 when comparing the storage address with the value in IAC6 for exact address compare (DBCR5[IAC56M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
24:27	IAC7XM	Instruction Address Compare 7 Extended Mask Control. IAC7XM allows for binary power of 2 address range compares for IAC7 without requiring the use of IAC8. 0000 No additional masking when DBCR5[IAC78M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC7 when comparing the storage address with the value in IAC7 for exact address compare (DBCR5[IAC78M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
28:31	IAC8XM	Instruction Address Compare 8 Extended Mask Control. IAC8XM allows for binary power of 2 address range compares for IAC8 without requiring the use of IAC7. 0000 No additional masking when DBCR5[IAC78M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC8 when comparing the storage address with the value in IAC8 for exact address compare (DBCR5[IAC78M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved

64.3.2.7 Debug Control Register 7 (DBCR7)

Debug Control Register 7 is used to enable and configure Data Address Compare 3 and 4 functionality. DBCR7 is shown in the following figure.



SPR - 596; Read/Write; Reset[1] - 0x0

Figure 64-9. DBCR7 Register

Note

[1] DBCR7 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 7.

Table 64-12. DBCR7 Field Descriptions

Bit	Name	Description
0:12	—	Reserved
13	DAC3XMH	Data Address Compare 3 Extended Mask Control High .DAC3XMH extends the range of the DAC3XM field
		0 DAC3XM masks 0–15 low-order address bits 1 DAC3XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC4XMH	Data Address Compare 4 Extended Mask Control High. . DAC4XMH extends the range of the DAC4XM field.
		0 DAC4XM masks 0–15 low-order address bits 1 DAC4XM masks 16–31 low-order address bits
16:19	DAC3XM	Data Address Compare 3 Extended Mask Control. DAC3XM allows for binary power of 2 address range compares for DAC3 without requiring the use of DAC4.
		Value of DAC3XMH || DAC3XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC3 when comparing the storage address with the value in DAC3 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.

Table continues on the next page...

Table 64-12. DBCR7 Field Descriptions (continued)

Bit	Name	Description
20:23	DAC4XM	Data Address Compare 4 Extended Mask Control. DAC4XM allows for binary power of 2 address range compares for DAC4 without requiring the use of DAC3.
		Value of DAC4XMH || DAC4XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC4 when comparing the storage address with the value in DAC4 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
24:27	DAC3CFG	Data Address Compare 3 Configuration. DAC3CFG controls whether DAC3 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint).
		0000 DAC3 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC3 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC3 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100Reserved 0101 DAC3 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC3 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC3 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1000 DAC3 address comparisons are used for stack limit checking. DAC3 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC3. When a qualified match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC3 watchpoints is generated. 1001 - 1111 Reserved

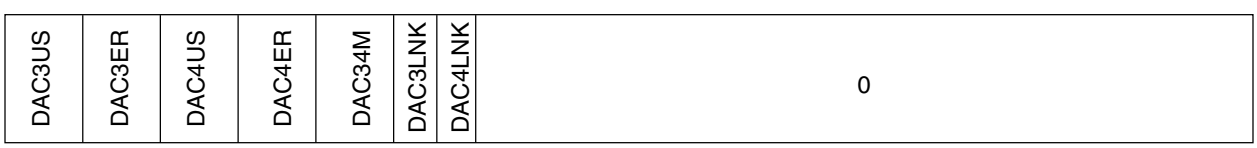
Table continues on the next page...

Table 64-12. DBCR7 Field Descriptions (continued)

Bit	Name	Description
28:31	DAC4CFG	Data Address Compare 4 Configuration. DAC4CFG controls whether DAC4 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function.
		0000 DAC4 debug watchpoints are enabled for load-type or store-type data storage accesses
		0001 DAC4 debug watchpoints are enabled only for store-type data storage accesses
		0010 DAC4 debug watchpoints are enabled only for load-type data storage accesses
		0011 Reserved
		0100 Reserved
		0101 DAC4 debug events and watchpoints are enabled only for store-type data storage accesses
		0110 DAC4 debug events and watchpoints are enabled only for load-type data storage accesses
		0111 DAC4 debug events and watchpoints are enabled for load-type or store-type data storage accesses
		1xxx Reserved

64.3.2.8 Debug Control Register 8 (DBCR8)

Debug Control Register 8 is used to configure Data Address Compare 3 and 4 operation. The DBCR8 register is shown in the following figure.



SPR - 597; Read/Write; Reset[1] - 0x0

Figure 64-10. DBCR8 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 8.

Table 64-13. DBCR8 Bit Definitions

Bit(s)	Name	Description
0:1	DAC3US	Data Address Compare 3 User/Supervisor Mode 00 - DAC3 debug events not affected by MSR_{PR} 01 - Reserved 10 - DAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 - DAC3 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	DAC3ER	Data Address Compare 3 Effective/Real Mode 00 - DAC3 debug events are based on effective address 01 - Unimplemented (Book E real address compare), no match can occur 10 - DAC3 debug events are based on effective address and can only occur if MSR_{DS}=0 11 - DAC3 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC4US	Data Address Compare 4 User/Supervisor Mode. 00 - DAC4 debug events not affected by MSR_{PR} 01 - Reserved 10 - DAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 - DAC4 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	DAC4ER	Data Address Compare 4 Effective/Real Mode 00 - DAC4 debug events are based on effective address 01 - Unimplemented (Book E real address compare), no match can occur 10 - DAC4 debug events are based on effective address and can only occur if MSR_{DS}=0 11 - DAC4 debug events are based on effective address and can only occur if MSR_{DS}=1
8:9	DAC34M	Data Address Compare 3/4 Mode 00 - Exact address compare. DAC3 debug events can only occur if the address of the data access is equal to the value specified in DAC3. DAC4 debug events can only occur if the address of the data access is equal to the value specified in DAC4. 01 - Address bit match. DAC3 debug events can occur only if the address of the data access ANDed with the contents of DAC4, are equal to the contents of DAC3 also ANDed with the contents of DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 10 - Inclusive address range compare. DAC3 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC3 and less than the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 11 - Exclusive address range compare. DAC3 debug events can occur only if the address of the data access is less than the value specified in DAC3 or is greater than or equal to the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used.

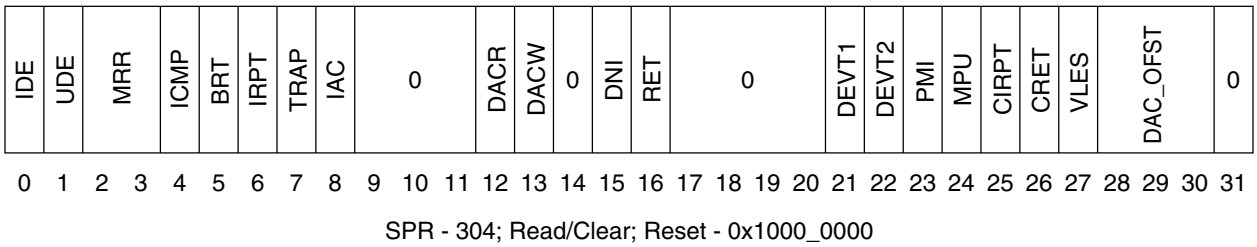
Table continues on the next page...

Table 64-13. DBCR8 Bit Definitions (continued)

Bit(s)	Name	Description
10	DAC3LNK	Data Address Compare 3 Linked 0 - No effect 1 - DAC3 debug events are linked to IAC5 debug events. IAC5 debug events do not affect DBSR When linked to IAC5, DAC3 debug events are conditioned based on whether the instruction also generated an IAC5 debug event. Note that linking is only available in EDM or IDM.
11	DAC4LNK	Data Address Compare 4 Linked 0 - No effect 1 - DAC4 debug events are linked to IAC7 debug events. IAC7 debug events do not affect DBSR When linked to IAC7, DAC4 debug events are conditioned based on whether the instruction also generated an IAC7 debug event. Note that linking is only available in EDM or IDM.
12:31	—	Reserved

64.3.2.9 Debug Status Register (DBSR)

The Debug Status Register (DBSR) contains status on debug events and the most recent processor reset. The Debug Status Register is set via hardware, and read and cleared via software. Bits in the Debug Status Register can be cleared using **mtspr***DBSR,RS*. Clearing is done by writing to the Debug Status Register with a 1 in any bit position that is to be cleared and 0 in all other bit positions. The write data to the Debug Status Register is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. Debug Status bits are set by Debug events only while Internal Debug Mode is enabled ($DBCR0_{IDM}$=1). When debug interrupts are enabled (MSR_{DE}=1 $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=0, or MSR_{DE}=1, $DBCR0_{IDM}$=1, $EDBCR0_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR other than MRR, DAC_OFST, or VLES will cause a debug interrupt to be generated. The debug interrupt handler is responsible for clearing DBSR bits prior to returning to normal execution. When resource sharing is enabled, ($EDBCR0_{EDM}$=1 and $EDBRAC0_{IDM}$=1), only software-owned resources may be modified by software, and status bits associated with hardware-owned resources will not be set by hardware in DBSR. The DBSR register is shown in the following figure.



SPR - 304; Read/Clear; Reset - 0x1000_0000

Figure 64-11. DBSR Register

The following table provides bit definitions for the Debug Status Register.

Table 64-14. DBSR Bit Definitions

Bit(s)	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if MSR_{DE}=0 and $DBCR0_{IDM}$=1 and a debug event causes its respective Debug Status Register bit to be set to 1. It may also be set to '1' if an imprecise debug event occurs due to a DAC event on a load or store which is terminated with error, or if an ICMP event occurs in conjunction with a EFPU FP round exception.
1	UDE	Unconditional Debug Event Set to 1 if an Unconditional debug event occurred.
2:3	MRR	Most Recent Reset. 00 - No reset occurred since these bits were last cleared by software 01 - A hard reset occurred since these bits were last cleared by software 10 - Reserved 11 - Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if an Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if an Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if an Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to 1 if a Trap Taken debug event occurred.
8	IAC	Instruction Address Compare Debug Event Set to 1 if an IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a write-type DAC debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 if a DNI debug event occurred

Table continues on the next page...

Table 64-14. DBSR Bit Definitions (continued)

Bit(s)	Name	Description
16	RET	Return Debug Event Set to 1 if a Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $PMGC0_{UDI}$=1
24	MPU	Memory Protection Unit Debug Event Set to 1 if a MPU debug event occurred. For MPU debug events, the IAC, DACR, or DACW status bit will also be set, depending on the type of access which matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. Note that software MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if an ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a PowerISA VLE Instruction. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction which took a DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3'b000 and $DBSR_{IDE}$ is set to 1. Normally set to 3'b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

64.3.2.10 Debug Data Effective Address Register (DDEAR)

The Debug Data Effective Address Register (DDEAR) contains address information for data address compare debug events (DAC or MPU DAC). DDEAR is update by hardware with the effective address of the load or store operation when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to

another DDEAR hardware update occurring. A subsequent DAC or MPU DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set, in order to capture the first event address.

The DDEAR register is shown in the following table.

Data Effective Address																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 600; Read/Write; Reset - unaffected

Figure 64-12. DDEAR Register

64.3.3 External Debug Resource Allocation Control Register (EDBRAC0)

The External Debug Resource Allocation Control Register (EDBRAC0) controls resource allocation when $EDBCR0_{EDM}$ is set to '1'. EDBRAC0 provides a mechanism for the hardware debugger to share certain debug resources with software. Individual resources are allocated based on the settings of EDBRAC0 when $EDBCR0_{EDM}$=1. EDBRAC0 settings are ignored when $EDBCR0_{EDM}$=0.

Hardware-owned resources which generate debug events update EDBSR0 instead of DBSR and cause entry into debug mode if the event is not masked in EDBSRMSK0, while software-owned resources which generate debug events if $DBCR0_{IDM}$=1 update DBSR, causing debug interrupts to occur if MSR_{DE}=1. EDBRAC0 is controlled via the OnCE port hardware, and is read-only to software.

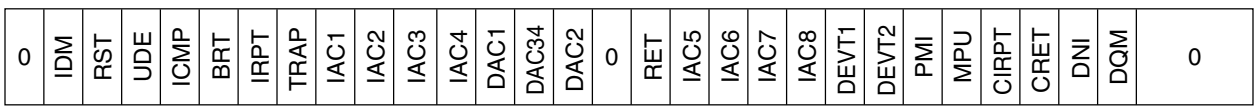
The DBSR status register is always owned by software. Debug status bits in DBSR are set by software-owned debug events only while Internal Debug Mode is enabled. When debug interrupts are enabled (MSR_{DE}=1 $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=0, or MSR_{DE}=1, $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR by an event which is software-owned (other than MRR, DAC_OFST, or VLES) will cause a debug interrupt to be generated.

Debug status bits in EDBSR0 are set by hardware-owned debug events only while External Debug Mode is enabled ($EDBCR0_{EDM}$=1). When $EDBCR0_{EDM}$=1, a set bit in EDBSR0 by an event which is hardware-owned (other than IDE, DAC_OFST, or VLES) will cause entry into debug mode unless entry is masked via EDBSRMSK0.

If $EDBCR0_{EDM}$=1, DBSR status bits corresponding to hardware-owned debug events are masked from being set by hardware.

Software-owned resources may be modified by software, but only the corresponding control bits in DBCR0–8 or MPU0CSR0 are affected by execution of a **mtspr**, thus only a portion of these registers may be affected, depending on the allocation settings in EDBRAC0. The debug interrupt handler is still responsible for clearing DBSR bits for software-owned resources prior to returning to normal execution. Hardware always has full access to all registers and register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers is performed.

The EDBRAC0 register is shown in the following figure.



SPR - 638; Read-only by Software; Reset - Unaffected by **p_reset_b,** reset to 0x00000180 by **m_por** or while in the test-logic-reset OnCE controller state

Figure 64-13. EDBRAC0 Register

Table 64-14 provides bit definitions for the Debug External Resource Control Register. Note that EDBRAC0 controls are disabled when EDBCR0EDM=0.

Table 64-15. EDBRAC0 Bit Definitions

Bit(s)	Name	Description
0	—	Reserved
1	IDM	Internal Debug Mode control 0 - Internal Debug mode may not be enabled by software. $DBCR0_{IDM}$ is owned exclusively by hardware. **mtspr** DBCR0–8 and other debug registers is always ignored. No resource sharing occurs, regardless of the settings of other fields in EDBRAC0. Hardware exclusively owns all resources. 1 - Internal Debug mode may be enabled by software. $DBCR0_{IDM}$ is owned by software. $DBCR0_{IDM}$ is software readable/writable. When $EDBRAC0_{IDM}$=1, software writes to hardware-owned bits in DBCR0–8 via **mtspr** are ignored.
2	RST	Reset Field Control 0 - $DBCR0_{RST}$ owned exclusively by hardware debug. No **mtspr** access by software to $DBCR0_{RST}$ field. 1 - $DBCR0_{RST}$ accessible by software debug. $DBCR0_{RST}$ is software readable/writable.
3	UDE	Unconditional Debug Event 0 - Event owned by hardware debug. 1 - Event owned by software debug.

Table continues on the next page...

Table 64-15. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
4	ICMP	Instruction Complete Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{ICMP}$ field. 1 - Event owned by software debug. $DBCR0_{ICMP}$ is software readable/writable.
5	BRT	Branch Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{BRT}$ field. 1 - Event owned by software debug. $DBCR0_{BRT}$ is software readable/writable.
6	IRPT	Interrupt Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{IRPT}$ field. 1 - Event owned by software debug. $DBCR0_{IRPT}$ is software readable/writable.
7	TRAP	Trap Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{TRAP}$ field. 1 - Event owned by software debug. $DBCR0_{TRA}$P is software readable/writable.
8	IAC1	Instruction Address Compare 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC1 control and status fields. 1 - Event owned by software debug. IAC1 control fields are software readable/writable.
9	IAC2	Instruction Address Compare 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC2 control and status fields. 1 - Event owned by software debug. IAC2 control fields are software readable/writable.
10	IAC3	Instruction Address Compare 3 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC3 control and status fields. 1 - Event owned by software debug. IAC3 control fields are software readable/writable.
11	IAC4	Instruction Address Compare 4 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC4 control and status fields. 1 - Event owned by software debug. IAC4 control fields are software readable/writable.
12	DAC1	Data Address Compare 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to DAC1 control and status fields. 1 - Event owned by software debug. DAC1 control fields are software readable/ writable.
13	DAC34	Data Address Compare 3 and 4 Debug Events 0 - Events owned by hardware debug. No **mtspr** access by software to DAC3 and DAC4 control and status fields. 1 - Event owned by software debug. DAC3 and DAC4 control fields are software readable/writable.

Table continues on the next page...

Table 64-15. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
14	DAC2	Data Address Compare 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to DAC2 control and status fields. 1 - Event owned by software debug. DAC2 control fields are software readable/ writable.
15	—	Reserved
16	RET	Return Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{RET}$ field. 1 - Event owned by software debug. $DBCR0_{RET}$ is software readable/writable.
17	IAC5	Instruction Address Compare 5 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC5 control and status fields. 1 - Event owned by software debug. IAC5 control fields are software readable/writable.
18	IAC6	Instruction Address Compare 6 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC6 control and status fields. 1 - Event owned by software debug. IAC6 control fields are software readable/writable.
19	IAC7	Instruction Address Compare 7 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC7 control and status fields. 1 - Event owned by software debug. IAC7 control fields are software readable/writable.
20	IAC8	Instruction Address Compare 8 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC8 control and status fields. 1 - Event owned by software debug. IAC8 control are software readable/writable.
21	DEVT1	External Debug Event Input 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT1}$ field. 1 - Event owned by software debug. $DBCR0_{DEVT1}$ is software readable/writable.
22	DEVT2	External Debug Event Input 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT2}$ field. 1 - Event owned by software debug. $DBCR0_{DEVT2}$ is software readable/writable.
23	PMI	Performance Monitor Interrupt Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to the PMRs. Performance monitor interrupts set $EDBSR0_{PMI}$ regardless of the setting of $PMGC0_{UDI}$ 1 - Event owned by software debug. PMRs are software readable/writable. **Note:** this bit is reset to '1'.

Table continues on the next page...

Table 64-15. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
24	MPU	Memory Protection Unit Debug Event 0 - Event owned by hardware debug. No **mpuwe** access by software to region descriptors which have the DEBUG control bit set to '1' or to the $MPU0CSR0_{DRDEN,DWDEN,IDEN}$ control bits, unless the CPU is in a debug session (**jd_debug_b** is asserted). MPU debug events set $EDBSR0_{MPU}$, and if not masked by $EDBSRMSK0_{MPU}$, one of $EDBSR0_{IAC}$, $EDBSR0_{DACR}$, or $EDBSR0_{DACW}$. MPU flash invalidates do not affect DEBUG=1 entries unless the CPU is in a debug session (**jd_debug_b** is asserted). 1 - Event owned by software debug. All region descriptors and the $MPU0CSR0_{DRDEN,DWDEN,IDEN}$ control bits are software readable/writable. **Note:** this bit is reset to '1'.
25	CIRPT	Critical Interrupt Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CIRPT}$ field. 1 - Event owned by software debug. $DBCR0_{CIRPT}$ is software readable/writable.
26	CRET	Critical Return Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CRET}$ field. 1 - Event owned by software debug. $DBCR0_{CRET}$ is software readable/writable.
27	DNI	DNI Instruction Debug Control 0 - DNI resource owned by hardware debug. Execution of an **e_dni** or **se_dni** instruction results in entry into debug mode. 1 - DNI resource owned by software debug. Execution of an **e_dni** or **se_dni** instruction results in either a debug interrupt ($DBCR0_{IDM}$=1 and MSR_{DE}=1) or a nop ($DBCR0_{IDM}$=0 or MSR_{DE}=0). **Note:** DNI events are not blocked during a debug session
28	DQM	Data Acquisition Messaging Registers 0 - $DEVENT_{DQTAG}$ and DDAM register are exclusively owned by hardware debug. No **mtspr** access by software to $DEVENTD_{QTAG}$ field or DDAM register. Attempted access by software is ignored. 1 - $DEVENT_{DQTAG}$ and DDAM register are owned by software. Software has read/write access to $DEVENT_{DQTAG}$ field and DDAM register.
29:31	—	Reserved

Table 64-16 shows which resources are controlled by EDBRAC0 settings.

Table 64-16. EDBRAC0 Resource Control

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr,affected by p_reset_b
0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	All debug registers
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{IDM}$
1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{RST}$
1	1	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{UDE}$
1	1	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{ICMP}$
1	1	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{BRT}$
1	1	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{IRPT}$
1	1	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{TRAP}$
1	1	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC1, $DBCR0_{IAC1}$, $DBCR1_{IAC1US,IAC1ER}$, $DBCR6_{IAC1XM}$
1	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC2, $DBCR0_{IAC2}$, $DBCR1_{IAC2US,IAC2ER}$, $DBCR6_{IAC2XM}$
1	1	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR1_{IAC12M}$
1	1	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC3, $DBCR0_{IAC3}$, $DBCR1_{IAC3US,IAC3ER}$, $DBCR6_{IAC3XM}$
1	1	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC4, $DBCR0_{IAC4}$, $DBCR1_{IAC4US,IAC4ER}$, $DBCR6_{IAC4XM}$
1	1	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR1_{IAC34M}$
1	1	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC5, $DBCR0_{IAC5}$, $DBCR5_{IAC5US,IAC5ER}$, $DBCR6_{IAC5XM}$

Table continues on the next page...

Table 64-16. EDBRAC0 Resource Control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr,affected by p_reset_b
1	1	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC6, $DBCR0_{IAC6}$, $DBCR5_{IAC6US,IAC6ER}$, $DBCR6_{IAC6XM}$
1	1	-	-	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR5_{IAC56M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC7, $DBCR0_{IAC7}$, $DBCR5_{IAC7US,IAC7ER}$, $DBCR6_{IAC7XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	IAC8, $DBCR0_{IAC8}$, $DBCR5_{IAC8US,IAC8ER}$, $DBCR6_{IAC8XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR5_{IAC78M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	DAC1, DVC1, DVC1U $DBCR0_{DAC1}$, $DBCR2_{DAC1US,DAC1ER}$, $DBCR2_{DVC1M,DVC1BE}$ $DBCR4_{DVC1C,DAC1XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	DAC3, DAC4 $DBCR7_{DAC\{3,4\},DAC\{3,4\}CFG, DAC\{3,4\}XM, DAC\{3,4\}XMH}$ $DBCR8_{DAC\{3,4\}US, DAC\{3,4\}ER, DAC\{3,4\}M}$

Table continues on the next page...

Table 64-16. EDBRAC0 Resource Control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$	$EDBRAC0_{MPU}$	$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr,affected by p_reset_b
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	DAC2, DVC2, DVC2U $DBCR0_{DAC2}$, $DBCR2_{DAC2US,DAC2ER}$, $DBCR2_{DVC2M,DVC2BE}$ $DBCR4_{DVC2C,DAC2XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	1	-	-	-	-	-	-	-	-	-	$DBCR2_{DAC12M}$
1	1	-	-	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	$DBCR2_{DAC1LNK}$
1	1	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	$DBCR2_{DAC2LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	$DBCR8_{DAC3LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	1	-	-	-	-	-	-	-	-	-	-	$DBCR8_{DAC4LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	$DBCR0_{RET}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	$DBCR0_{DEVT1}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	$DBCR0_{DEVT2}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	All PMRs
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	MPU entries with DEBUG bit set, $MPU0CSR0_{DRDEN,DWDEN,IDEN}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	$DBCR0_{CIRPT}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	$DBCR0_{CRET}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-	-	-	1	-	
1	-[1]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	$DEVENT_{DQTAG}$, DDAM

1. Note: IDM not required to be set to enable software access.

EDBRAC0 also controls which bits or fields in DBCR0–8, MPU0CSR0, and the PMRs are reset by assertion of **p_reset_b** when $EDBCR0_{EDM}$=1. Only software-owned bits or fields as shown in Table 64-16 are affected in this case, except that $DBCR0_{RST}$ and $DBSR_{MRR}$ are updated by assertion of **p_reset_b** regardless of the value of $EDBCR0_{EDM}$ or EDBRAC0.

64.3.4 Debug Event Select Register (DEVENT)

The Debug Event Select Register allows instrumented software to internally generate signals when a **mtspr** instruction is executed and this register is accessed. The values written to this register determine which of the **p_devnt_out[0:7]** processor output signals are asserted upon access. Writing a '1' to any of these bit positions will cause a one clock pulse to be generated on the corresponding output. For **p_devnt_out[0:3]**, a corresponding **jd_watchpt[x]** output is asserted as well to indicate a watchpoint has occurred. These signals may be used for internal core debug resources as well as for SoC level cross-triggering. See the SoC User's Manual for more information on SoC use cases.

The $DEVENT_{DEVNT}$ register field value is undefined on a read; it may or may not remain set to the last value written. Since it is unconditionally shared by hardware debug and software, software should not rely on any value remaining.

The upper 8 bits of the DEVENT register also provide the DQTAG used to identify channels within Data Acquisition Messages. See the "Data Acquisition ID Tag Field" section in the Nexus 3 Module chapter for more detail on the DQTAG.

The DEVENT register is shown in the following figure.

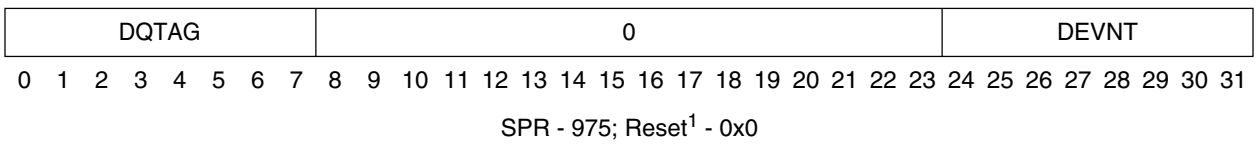


Figure 64-14. DEVENT Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**. Note that DEVNT field is shared by hardware and software but is always reset by **p_reset_b**.

The following table provides bit definitions for the Debug Event Register.

Table 64-17. DEVENT Bit Definitions

Bit(s)	Name	Description
0:7	DQTAG	Data Acquisition Message IDTAG channel identifier (supplied to Nexus 3)
8:23	—	Reserved, should be cleared.

Table continues on the next page...

Table 64-17. DEVENT Bit Definitions (continued)

Bit(s)	Name	Description	
24:31	DEVNT	Debug Event Signals	
			00000000 - No signal is asserted
			xxxxxxx1 - **p_devnt_out[0]** and **jd_watchpt[12]** are asserted for one clock
			xxxxxx1x - **p_devnt_out[1]** and **jd_watchpt[13]** are asserted for one clock
			xxxxx1xx - **p_devnt_out[2]** and **jd_watchpt[20]** are asserted for one clock
			xxxx1xxx - **p_devnt_out[3]** and **jd_watchpt[21]** are asserted for one clock
			xxx1xxxx - **p_devnt_out[4]** is asserted for one clock
			xx1xxxxx - **p_devnt_out[5]** is asserted for one clock
			x1xxxxxx - **p_devnt_out[6]** is asserted for one clock
			1xxxxxxx - **p_devnt_out[7]** is asserted for one clock

64.3.5 Debug Data Acquisition Message Register (DDAM)

The Debug Data Acquisition Message Register allows instrumented software to generate real-time Data Acquisition Messages (as defined by Nexus 3) via a **mtspr** instruction to this register. See the "Data Acquisition Messaging" section in the Nexus 3 Module chapter for details.

The DDAM register is shown in the following figure.

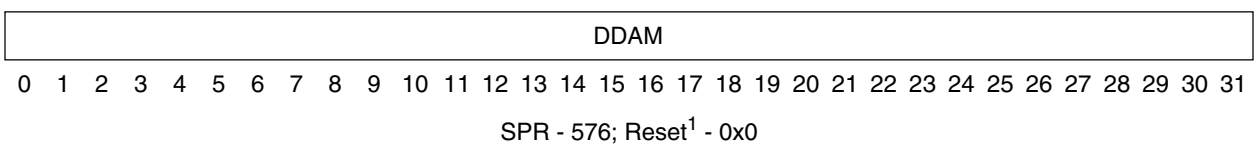


Figure 64-15. DDAM Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for the Debug Data Acquisition Message Register.

Table 64-18. DDAM Bit Definitions

Bit(s)	Name	Description
0:31	DDAM	Value to be transmitted in a Data Acquisition Message (DQM) (supplied to Nexus 3 with strobe)

64.4 Using Debug Resources for Stack Limit Checking

The DAC1,2 and DAC3,4 resources can be used for stack overflow/underflow detection when not being used as a hardware or software debug resource. Stack limit checking is available regardless of EDM or IDM mode, and when resources used for stack limit checking are owned by software, will utilize a DSI or machine check exception. Software-owned stack limit checking does not require IDM to be set. Hardware owned stack limit checking requires EDM to be set.

When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by software, DAC events are not generated for resources configured to perform stack limit checking, and no DBSR DAC status flag will be set due to a detected stack limit violation. Instead, depending on the processor mode, a data storage interrupt or a machine check exception is signaled. When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by hardware, DAC events will be generated for resources configured to perform stack limit checking, and the EDBSR0 DAC status flag will be set due to a detected stack limit violation, causing entry into debug halted mode in the same way as a DAC exception normally does. The only difference is that qualification of the access address is performed as discussed in the next paragraph.

Stack limit checking is implemented in the same way as range compares using DAC1,2 or 3,4, or extended masking using DAC1 or DAC3, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions which may calculate an EA using GPR R1 such as cache, MMU/ MPU management instructions, or instructions which indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions) or a modify specifier (for LSP load/store instructions w/modify). When DAC resources configured to perform stack limit checking are not owned by hardware, if a stack limit violation occurs when performing the load or store, the access is aborted, and an error report machine check is generated, with MCSRR0 pointing to the address of the load or store access which generated the stack overflow/underflow. If DAC resources configured to perform stack

limit checking are owned by hardware, then a normal DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event.

Enabling of this functionality is described in more detail in Table 64-9. Note that IDM is not required to be set for software-owned stack limit checking.

In contrast to the normal case of DAC address matching resulting in a debug event, the stack limit check address compare logic operates differently for stack limit checking. When using resources for stack limit checking, a DACn hit to a resource enabled for performing stack limit checking on a stack access indicates a stack limit violation.

When stack limit checking is enabled by the setting of the $DBCR4_{DAC1CFG}$ field to '1000' or the $DBCR8_{DAC3CFG}$ field to '1000' (or both), and the corresponding DACn resources are owned by software, DACn events are not generated and the DBSR DAC status flag will not be set due to a detected stack limit hit. Instead, if stack limit checking is enabled for supervisor mode stack accesses in DAC1 or DAC3, and a compare hit occurs for a supervisor mode stack access (A load or store using GPR R1 in the <EA> calculation), a machine check exception is signaled. If stack limit checking is enabled for user mode accesses, a DSI exception is signaled when a stack limit checking enabled DAC1 or DAC3 compare hit occurs for a user mode access. A watchpoint for DAC1 or DAC3 will be generated when the stack limit violation is detected, even though the instruction does not complete. Stack limit checking for supervisor mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to '1000' with $DBCR2_{DAC1US}$ set to '00' or '10', or the $DBCR7_{DAC3CFG}$ field is set to '1000' with $DBCR8_{DAC3US}$ set to '00' or '10', or both. Stack limit checking for user mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to '1000' with $DBCR2_{DAC1US}$ set to '00' or '01', or the $DBCR7_{DAC3CFG}$ field is set to '1000' with $DBCR8_{DAC3US}$ set to '00' or '01', or both. Note that unlike regular debug DAC events, both halves of a misaligned access are checked for limit violations.

When stack limit checking is enabled for a stack access, and DACn resources are owned by hardware, the EDBSR0 DAC status flag will be set due to a detected stack limit violation, to cause entry into debug halted mode or to generate a watchpoint, or both, in the same way as a DAC event normally does, i.e. after the access has completed. The only difference is that qualification of the access address is performed as discussed in the next paragraph. If the access is aborted due to a DSI or other exception such as machine check condition, the $EDBSR0_{IDE}$ status bit will also be set to indicate that the data access instruction was not completed.

Stack limit checking is implemented in the same way as address compares using DACn, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions which may calculate an EA

using GPR R1 such as cache, MMU/MPU management instructions, or instructions which indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions) or a modify specifier (for LSP load/store instructions w/modify). When DACn resources are not owned by hardware, if a stack limit violation occurs (a hit to a stack limit enabled DAC is detected) when performing the load or store, the access request is aborted and the access is not performed. Instead, for supervisor mode accesses, an error report machine check exception is generated, with MCSRR0 pointing to the address of the load or store instruction which generated the stack overflow/underflow, and for user mode accesses, a DSI exception is generated, with SRR0 pointing to the address of the load or store instruction which generated the stack overflow/underflow. If all stack limit check enabled DACn resources are owned by hardware, then a DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event. In this case, the instruction and access will normally have completed, in the same manner as a normal DAC event.

Independent limit checks for supervisor and user accesses may be implemented by allocating independent DACn resources to each, or a single limit may be applied using a single DACn resource. Typically, a DAC1,2 pair operating in exclusive address range mode is utilized for stack limit checking for a single shared limit. For separate limits, DAC3,4 may also be utilized. If more than one DACn resource is utilized, a DAC hit on any resource utilized for stack limit checking will cause the corresponding stack limit exception action to occur. If both a hardware-owned and a software-owned resource generate a stack limit exception for a given load or store, the software resource will have priority, since it is detected prior to completion of the access, and the access is aborted, thus the hardware event will not occur.

Enabling of this functionality is described in more detail in the description of the DACnCFG fields in Table 64-9 and Table 64-12 in Debug Control and Status registers. Note that for DAC1 and DAC2, access type (read, write) control is part of DBCR0.

64.5 External Debug Support

External debug support is supplied through the OnCE controller serial interface which allows access to internal CPU registers and other system state while the CPU is halted in debug mode. All debug resources including DBCR0–8, DBSR, IAC1–8, DAC1–4, and DVC1–2 are accessible through the serial OnCE interface in external debug mode. Setting the $EDBCR0_{EDM}$/$DBCR0_{EDM}$ bit to '1' through the OnCE interface enables external debug mode, and unless otherwise permitted by the settings in EDBRAC0, disables software updates to the debug control registers. When $EDBCR0_{EDM}$ is set, debug events enabled to set respective status bits will also cause the CPU to enter Debug Mode if the event is not masked in EDBSRMSK0, as opposed to generating Debug Interrupts,

unless the specific events are allocated to software via the settings in EDBRAC0. In Debug Mode, the CPU is halted at a recoverable boundary, and an external Debug Control Module may control CPU operation through the On-Chip Emulation logic (OnCE).

Note that the descriptions of events in the subsections of Software Debug Events and Exceptions refer to setting DBSR status bits, however, when resources are owned by hardware, the events for those resources set the respective status bits in EDBSR0 instead of DBSR.

Note

On the initial setting of $EDBCR0_{EDM}$ to '1', other bits in DBCR0 will remain unchanged. After $EDBCR0_{EDM}$ has been set, all debug register resources may be subsequently controlled through the OnCE interface. The CPU should be placed into debug mode via the OCR_{DR} control bit prior to writing EDM to '1'. This gives the debugger the opportunity to cleanly write to the DBCRx registers and the DBSR to clear out any residual state / control information which could cause unintended operation.

Note

It is intended for the CPU to remain in external debug mode ($EDBCR0_{EDM}$=1) in order to single step or perform other debug mode entry/ reentry via the OCR_{DR}, by performing go+noexit commands, or by assertion of the **jd_de_b** signal.

Note

$EDBCR0_{EDM}$ operation will be blocked if OnCE operation is disabled (**jd_en_once** negated) regardless of whether it is set or cleared. This means that if $EDBCR0_{EDM}$ was previously set, and then **jd_en_once** is negated (this should not occur), entry into debug mode will be blocked, and all *hardware* debug events are blocked. Watchpoints are not blocked.

Due to clock domain design, the CPU clock (**m_clk**) must be active in order to perform writes to debug registers other than the OnCE Command register (OCMD), the OnCE Control register (OCR), External Debug Control Register 0 (EDBCR0), External Debug Status register 0 (EDBSR0), External Debug Status Register Mask 0 (EDBSRMSK0), or the $EDBCR0_{EDM}$ bit. Register read data is synchronized back to the **j_tclk** clock domain. The OnCE Control register provides the capability of signaling the system level clock controller that the CPU clock should be activated if not already active.

Updates to the DBCRx, DBSR, and most other debug registers via the OnCE interface should be performed with the CPU in debug mode to guarantee proper operation. Due to the various points in the CPU pipeline where control is sampled and event handshaking is performed, it is possible that modifications to these registers while the CPU is running may result in early or late entry into debug mode, and may have incorrect status posted in the DBSR register.

If resource sharing is enabled via EDBRAC0, updates to the EDBRAC0, DBCRx, and DBSR registers must be performed with the CPU in debug mode, since simultaneous updates of register portions could otherwise be attempted, and such updates are not guaranteed to properly occur. The results of such an attempt are undefined.

64.5.1 External Debug Registers

The external debug registers are used for controlling several debug aspects of the core and reporting status while in External Debug Mode.

64.5.1.1 External Debug Control Register 0 (EDBCR0)

EDBCR0 is a control register accessible to an external debugger through the OnCE/Jtag port. An external development tool can write to this register in order to enable external debug mode or to enable Debugger Notify Halt instructions (**e_dnh**, **se_dnh**), as well as to control aspects of Debugger Notify Interrupt instructions (**e_dni**, **se_dni**).

EDBCR0 is not accessible by software, However, the state of $EDBCR0_{EDM}$ is reflected as a read-only bit in $DBCR0_{EDM}$ to software. There is only one physical EDM bit implemented; it is reflected in both the DBCR0 and EDBCR0 registers, and may be written and read using either register by the hardware debugger. For future compatibility, EDBCR0 updates are preferred.

EDBCR0 is shown in the following figure.

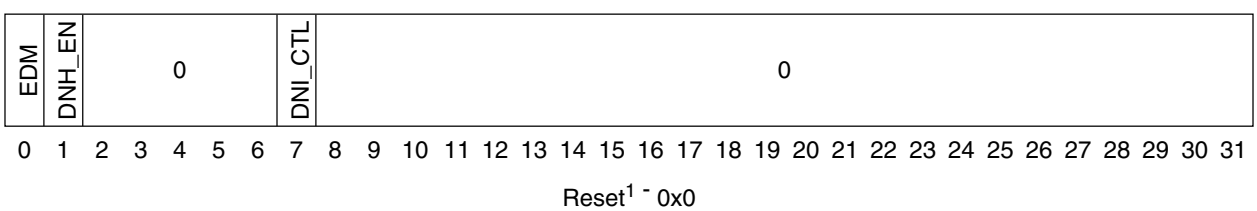


Figure 64-16. EDBCR0 Register

Note

[1] EDBCRO is affected (reset) by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b.**

The following table provides bit definitions for External Debug Control Register 0.

Table 64-19. EDBCR0 Bit Definitions

Bit(s)	Name	Description
0	EDM	External Debug Mode. This bit is also reflected in DBCR0 0 - External debug mode disabled. Internal debug events not mapped into external debug events. 1 - External debug mode enabled. Hardware-owned events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCRx, IAC1-8, DAC1-4, DVC1-2[U]} unless permitted by settings in EDBRAC0. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset **p_reset_b**. This allows the debugger to set up hardware debug events which remain active across a processor reset.
1	DNH_EN	**dnh** Instruction Enable 0 - execution of **e_dnh** and **se_dnh** instructions cause illegal instruction exceptions to occur. 1 - execution of **e_dnh** and **se_dnh** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of EDM.
2:6	---	Reserved
7	DNI_CTL	**dni** Instruction Control 0 - When the **dni** resource is owned by hardware, the MSR_{DE} bit is cared, and when MSRDE=0, execution of **e_dni** and **se_dni** instructions are nop'ed and no entry into debug mode occurs. 1 - When the **dni** resource is owned by hardware, the MSR_{DE} bit is don't-cared, and execution of **e_dni** and **se_dni** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of MSR_{DE}. Note that this control bit is only used when the dni resource is owned by hardware via control in $EDBRAC0_{DNI}$, and thus also only when EDM=1
8:31	---	Reserved

64.5.1.2 External Debug Status Register 0 (EDBSR0)

The External Debug Status Register 0 (EDBSR0) contains status on debug events owned by hardware. The External Debug Status Register 0 is set via hardware, and read and cleared via OnCE access by the debugger. Clearing is done by writing to the External Debug Status Register via the OnCE port, with a '1' in any bit position that is to be

cleared and '0' in all other bit positions. The write data to EDBSR0 is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. The EDBSR0 register is shown in the following figure.

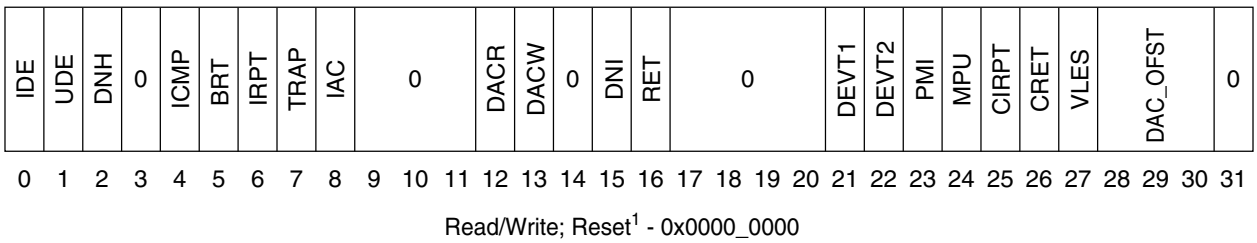


Figure 64-17. EDBSR0 Register

Note

[1] Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCRO_{EDM}$=O.

The following table provides bit definitions for External Debug Status Register 0.

Table 64-20. EDBSR0 Bit Definitions

Bit(s)	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if $EDBCR0_{EDM}$=1 and an imprecise debug event occurs for a hardware-owned DAC event due to a load or store which is terminated with error, or if a hardware-owned ICMP event occurs in conjunction with a EFPU round exception. This bit will not be set for imprecise debug events which are masked via settings in EDBSRMSK0.
1	UDE	Unconditional Debug Event Set to 1 if a hardware-owned Unconditional debug event occurred.
2	DNH	Debugger Notify Halt Event Set to 1 if a debugger notify halt instruction was executed and caused a debug halt.
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if a hardware-owned Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if a hardware-owned Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if a hardware-owned Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to 1 if a hardware-owned Trap Taken debug event occurred.
8	IAC	Instruction Address Compare 1 Debug Event Set to 1 if a hardware-owned IAC debug event occurred.
9:11	—	Reserved

Table continues on the next page...

Table 64-20. EDBSR0 Bit Definitions (continued)

Bit(s)	Name	Description
12	DACR	Data Address Compare Read Debug Event Set to 1 if a hardware-owned read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a hardware-owned write-type DAC1 debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 if a hardware-owned DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a hardware-owned Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a hardware-owned DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a hardware-owned DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $EDBRAC0_{PMI}$=0 regardless of the setting of $PMGC0_{UDI}$
24	MPU	Memory Protection Unit Debug Event Set to 1 if a hardware-owned MPU debug event occurred. For MPU debug events, if $EDBSRMSK0_{MPU}$ is cleared, the IAC, DACR, or DACW status bit will also be set, depending on the type of access which matched a region descriptor enabled to generate debug events, in order to further define the type of MPU debug event. If $EDBSRMSK0_{MPU}$ is set at the time a MPU debug event occurs, the IAC, DACR, and DACW status bits will not be set by MPU debug events, in order to properly mask the MPU debug event. Note that hardware MPU debug events on data accesses normally occur after the data access is performed and the instruction completes, thus act like a normal DAC debug event. The instruction may not complete if a DSI occurs. In this case, IDE will be set to indicate this occurrence.
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a hardware-owned Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a hardware-owned Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if a hardware-owned ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a PowerISA VLE Instruction. Also set for execution of an e_dnh or se_dnh instruction when enabled by $EDBCR0_{DNH_EN}$. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction which took a hardware-owned DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3'b000 and $EDBSR0_{IDE}$ is set to 1. Normally set to 3'b000 by a non-DVC DAC. A DVC DAC may set this field to any value.

Table continues on the next page...

Table 64-20. EDBSR0 Bit Definitions (continued)

Bit(s)	Name	Description
31	—	Reserved

64.5.1.3 External Debug Status Register Mask 0 (EDBSRMSK0)

The External Debug Status Register Mask 0 (EDBSRMSK0) is used to mask debug events set in EDBSR0 from causing entry into debug halted mode. A '1' stored in any mask bit prevents debug mode entry caused by the corresponding bit being set in EDBSR0. The mask has no effect on DBSR actions or on the setting of EDBSR0 status bits by hardware-owned events, except that the IDE bit will not be set by imprecise hardware-owned debug events which are masked. EDBSRMSK0 may be used to allow debug events owned by hardware to be configured for watchpoint generation purposes without causing debug mode entry when the watchpoint occurs. EDBSRMSK0 is read and written via OnCE access by the debugger. No software access is provided. The EDBSRMSK0 register is shown in the following figure.

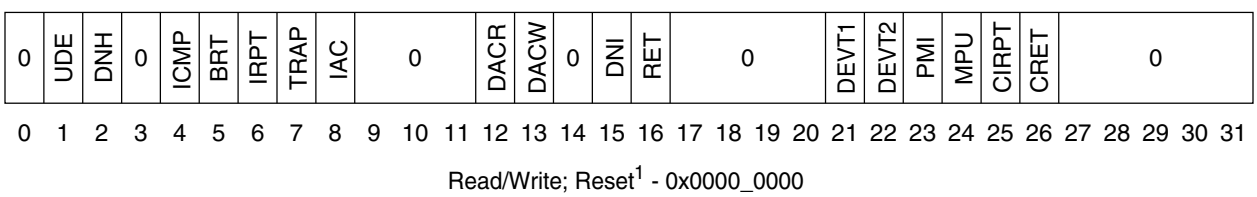


Figure 64-18. EDBSRMSK0 Register

Note

[1] Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCRO_{EDM}$=O.

The following table provides bit definitions for External Debug Status Register Mask 0.

Table 64-21. EDBSRMSK0 Bit Definitions

Bit(s)	Name	Description
0	—	Reserved
1	UDE	Unconditional Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{UDE}$
2	DNH	Debugger Notify Halt Event Set to 1 to mask debug mode entry by $EDBSR0_{DNH}$
3	—	Reserved

Table continues on the next page...

Table 64-21. EDBSRMSK0 Bit Definitions (continued)

Bit(s)	Name	Description
4	ICMP	Instruction Complete Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{ICMP}$
5	BRT	Branch Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{BRT}$
6	IRPT	Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IRPT}$
7	TRAP	Trap Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{TRAP}$
8	IAC	Instruction Address Compare Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IAC}$
9:11	—	Reserved
12	DACR	Data Address Compare 1 Read Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACR}$
13	DACW	Data Address Compare 1 Write Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACW}$
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DNI}$
16	RET	Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{RET}$
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT1}$
22	DEVT2	External Debug Event 2 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT2}$
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{PMI}$
24	MPU	Memory Protection Unit Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{MPU}$
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CIRPT}$
26	CRET	Critical Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CRET}$
22:31	—	Reserved

64.5.1.4 External Debug Data Effective Address Register (EDDEAR)

The External Debug Data Effective Address Register (EDDEAR) contains address information for hardware-owned data address compare debug events, including MPU DAC events. EDDEAR is update by hardware with the effective address of the load or store operation when an unmasked data address compare event is recorded in EDBSR0 if the previous values of $EDBSR0_{DAC\{R,W\}}$ bits which are unmasked in EDBSRMSK0 are zero. Once an EDDEAR update is performed, these unmasked bits must be cleared by the hardware debugger prior to another EDDEAR hardware update occurring. A subsequent hardware-owned unmasked DAC event will not update the EDDEAR register if either of the $EDBSR0_{DAC\{R,W\}}$ bits are set and are not masked by EDBSRMSK0, in order to capture the first unmasked event address. EDDEAR is read and written via OnCE access by the debugger. No software access is provided.

The EDDEAR register is shown in the following figure.

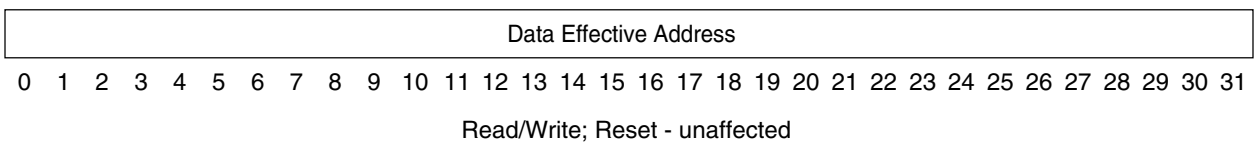


Figure 64-19. EDDEAR Register

64.5.2 OnCE Introduction

The on-chip emulation circuitry (OnCE™/Nexus Class 1 interface) provides a means of interacting with the core and integrated system so that a user may examine registers, memory, or on-chip peripherals facilitating hardware/software development. OnCE operation is controlled via an industry standard IEEE 1149.1 TAP controller. By using public instructions, the external hardware debugger can freeze or halt the CPU, read and write internal state, and resume normal execution. The core does not contain IEEE 1149.1 standard boundary cells on its interface, as it is a building block for further integration. It does not support the JTAG related boundary scan instruction functionality, although JTAG public instructions may be decoded and signaled to external logic.

The OnCE logic provides for Nexus Class 1 static debug capability (utilizing the same set of resources available to software while in internal debug mode).

In order to enable full OnCE operation, the **jd_enable_once** input signal must be asserted. In some system integrations, this is automatic, since the input will be tied asserted. Other integrations may require the execution of the Enable OnCE command via the TAP and appropriate entry of serial data. Exact requirements will be documented by the integrated product specification. The **jd_enable_once** input signal should not change state during a debug session, or undefined activity may occur.

The following figures show the TAP controller state model and the TAP registers implemented by the OnCE logic.

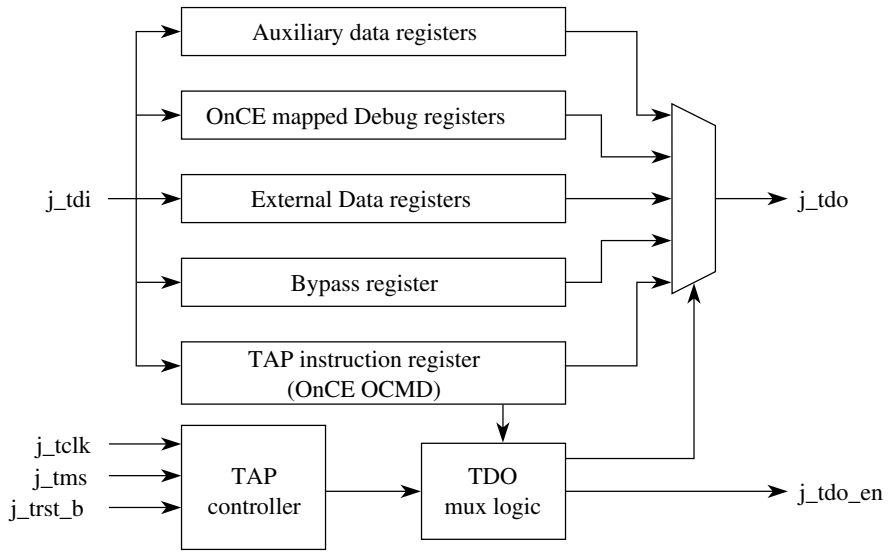
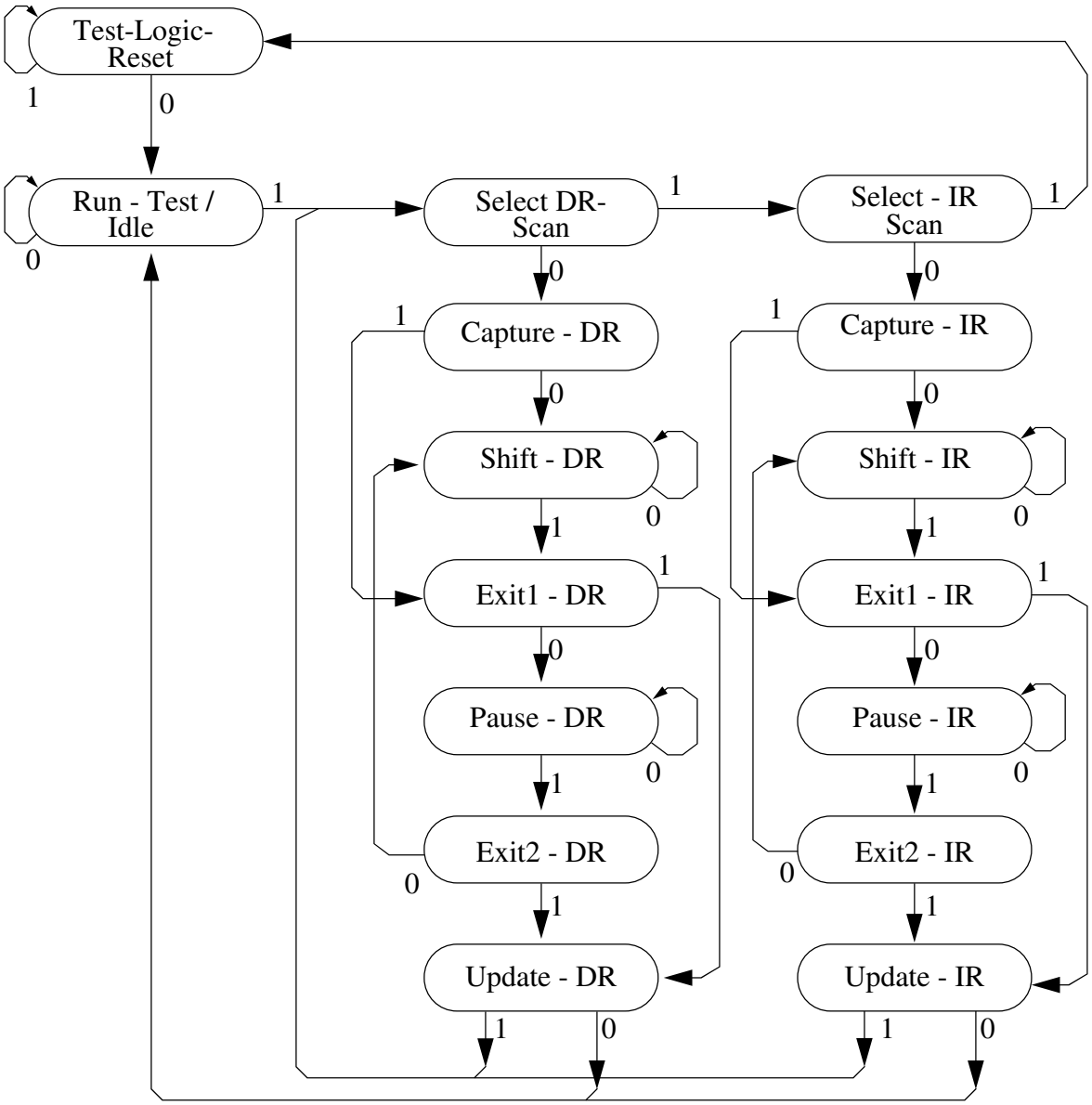


Figure 64-20. OnCE TAP Controller and Registers

The OnCE controller is implemented as a 16-state FSM, with a one-to-one correspondence to the states defined for the JTAG TAP controller.



Access to processor registers and the contents of memory locations are performed by enabling external debug mode (setting $EDBCR0_{EDM}$ to '1'), placing the processor into debug mode, followed by scanning instructions and data into and out of the CPU Scan Chain (CPUSCR); execution of scanned instructions by the CPU is used as the method to access required data. Memory locations may be read by scanning a load instruction into the CPU core which will reference the desired memory location, executing the load instruction, and then scanning out the result of the load. Other resources are accessed in a similar manner.

The initial entry by the CPU into the debug state (or mode) from normal, Waiting, Stopped, or Halted states (all indicated via the OnCE Status Register (OSR), OnCE Status Register) by assertion of one or more debug requests, begins a *debug session*. The **jd_debug_b** output signal indicates that a debug session is in progress, and the OSR will

indicate the CPU is in the debug state. Instructions may the be single-stepped by scanning new values into the CPUSCR, and performing a OnCE go+noexit command (See OnCE Command Register (OCMD)). The CPU will then temporarily exit the debug state (but not the debug session) to execute the instruction, and will then return to the debug state (again indicated via the OnCE Status Register (OSR)). The debug session remains in force until the final OnCE go+exit command is executed, at which time the CPU will return to the previous state it was in (unless a new debug request is pending). A scan into the CPUSCR is required prior to executing each go+exit or go+noexit OnCE command.

64.5.3 JTAG/OnCE Pins

The JTAG/OnCE pin interface is used to transfer OnCE instructions and data to the OnCE control block. Depending on the particular resource being accessed, the CPU may need to be placed in the Debug mode. For resources outside of the CPU block and contained in the OnCE block, the processor is not disturbed, and may continue execution. If a processor resource is required, an internal debug request (**dbg_dbgrq**) may be asserted to the CPU by the OnCE controller, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter Debug Mode, and wait for further commands. Asserting **dbg_dbgrq** will cause the chip to temporarily exit the Waiting, Stopped or Halted power management states.

The following table details the primary JTAG/OnCE interface signals.

Table 64-22. JTAG/OnCE Primary Interface Signals

Signal name	Type	Description
j_trst_b	I	JTAG test reset
j_tclk	I	JTAG test clock
j_tms	I	JTAG test mode select
j_tdi	I	JTAG test data input
j_tdo	O	Test data out to master controller or pad
j_tdo_en[1]	O	Enables TDO output buffer

1. j_tdo_en is asserted when the TAP controller is in the shift_DR or shift_IR state.

A full description of JTAG pins is provided in the JTAG Support Signals section of the Core Complex Overview.

64.5.4 OnCE Internal Interface Signals

The following paragraphs describe the OnCE interface signals to other internal blocks associated with the OnCE controller.

64.5.4.1 CPU Debug Request (dbg_dbgrq)

The **dbg_dbgrq** signal is asserted by the OnCE control logic to request the CPU to enter the debug state. It may be asserted for a number of different conditions, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter the debug mode, and wait for further commands.

64.5.4.2 CPU Debug Acknowledge (cpu_dbgack)

The **cpu_dbgack** signal is asserted by the CPU upon entering the debug state. This signal is used as part of the handshake mechanism between the OnCE control logic and the rest of the CPU. The CPU core may enter debug mode either through a software or hardware event.

64.5.4.3 CPU Address, Attributes

The CPU address and attribute information are used by the Nexus3 unit with information for real-time address trace information.

64.5.4.4 CPU Data

The CPU data buses are used to supply the Nexus3 debug unit with information for real-time data trace capability.

64.5.5 OnCE Interface Signals

The following paragraphs describe additional OnCE interface signals to other external blocks such as a Nexus controller and external blocks which may need information pertaining to debug operation.

64.5.5.1 OnCE Enable (jd_en_once)

The OnCE enable signal **jd_en_once** is used to enable the OnCE controller to allow certain instructions and operations to be executed. Assertion of this signal will enable the full OnCE command set, as well as operation of control signals and OnCE Control register functions. When this signal is disabled, only the Bypass, ID and Enable_OnCE

commands are executed by the OnCE unit, and all other commands default to a "Bypass" command. The OnCE Status register (OSR) is not visible when OnCE operation is disabled. In addition, OnCE Control register (OCR) functions are disabled, as is the operation of the **jd_de_b** input. Secure systems may choose to leave the **jd_en_once** signal negated until a security check has been performed. Other systems should tie this signal asserted to enable full OnCE operation. The **j_en_once_regsel** output signal is provided to assist external logic performing security checks.

The **jd_en_once** input must only change state during the Test-Logic-Reset, Run-Test/ Idle, or Update_DR TAP states. A new value will take affect after one additional **j_tclk** cycle of synchronization. In addition, **jd_enable_once** input signal must not change state during a debug session, or undefined activity may occur.

64.5.5.2 OnCE Debug Request/Event (jd_de_b, jd_de_en)

If implemented at the SoC level, a system level bidirectional open drain debug event pin **DE_b** (not part of the interface) provides a fast means of entering the Debug Mode of operation from an external command controller (when input) as well as a fast means of acknowledging the entering of the Debug Mode of operation to an external command controller (when output). The assertion of this pin by a command controller causes the CPU core to finish the current instruction being executed, save the instruction pipeline information, enter Debug Mode, and wait for commands to be entered. If **DE_b** was used to enter the Debug Mode then **DE_b** must be negated after the OnCE controller responds with an acknowledge and before sending the first OnCE command. The assertion of this pin by the CPU Core acknowledges that it has entered the Debug Mode and is waiting for commands to be entered.

To support operation of this system pin, the OnCE logic supplies the **jd_de_en** output and samples the **jd_de_b** input when OnCE is enabled (**jd_en_once** asserted). Assertion of **jd_de_b** will cause the OnCE logic to place the CPU into Debug Mode. Once Debug Mode has been entered, the **jd_de_en** output will be asserted for three **j_tclk** periods to signal an acknowledge. **jd_de_en** can be used to enable the open-drain pulldown of the system level **DE_b** pin.

For systems which do not implement a system level bidirectional open drain debug event pin **DE_b**, the **jd_de_en** and **jd_de_b** signals may still be used to handshake debug entry.

64.5.5.3 OnCE Debug Output (jd_debug_b)

The OnCE Debug output **jd_debug_b** is used to indicate to on-chip resources that a debug session is in progress. Peripherals and other units may use this signal to modify normal operation for the duration of a debug session, which may involve the CPU executing a sequence of instructions solely for the purpose of visibility/system control which are not part of the normal instruction stream the CPU would have executed had it not been placed in debug mode. This signal is asserted the first time the CPU enters the debug state, and remains asserted until the CPU is released by a write to the OnCE Command Register with the GO and EX bits set, and a register specified as either "No Register Selected" or the CPUSCR. This signal will remain asserted even though the CPU may enter and exit the debug state for each instruction executed under control of the OnCE controller. See OnCE Command Register (OCMD) for more information on the function of the GO and EX bits. This signal is not normally used by the CPU.

64.5.5.4 CPU Clock On Input (jd_mclk_on)

The CPU Clock On input **jd_mclk_on** is used to indicate that the CPU's **m_clk** input is active. This input signal is expected to be driven by system logic external to the core, is synchronized to the **j_tclk** (scan clock) clock domain, and is presented as a status flag on the **j_tdo** output during the Shift_IR state. External firmware may use this signal to ensure proper scan sequences will occur to access debug resources in the **m_clk** clock domain.

64.5.5.5 Watchpoint Events (jd_watchpt[0:31])

The **jd_watchpt[0:31]** signals may be asserted by the OnCE control logic to signal that a watchpoint condition has occurred. Watchpoints do not directly cause the CPU to be affected. They are provided to allow external visibility only or for triggering purposes. Watchpoint events are conditioned by the settings in the DBCRxx registers, as well as by the DEVENT register, the DTC/DTSA/DTEA registers, the Performance Monitor control register settings, and generation of MPU debug events. Refer to Table 64-28 for details of the signal assignments. Note that assertion of most watchpoint outputs is conditioned on being in EDM or IDM. The Performance monitor, DEVENT, and DTC watchpoints are not conditioned however, and may assert regardless of the state of EDM or IDM. In addition, DAC1 or DAC3 watchpoints may be generated for stack limit violation occurrences when those resources are configured to perform stack limit checking, regardless of the state of EDM or IDM.

64.5.5.6 Update DR w/go+exit (j_ocmd_go_exit)

This signal indicates the TAP controller is in the Update_DR state and that the go and exit bits in the OnCE Command register are high, and RS indicates "no register selected". This signal will assert regardless of whether the CPU is currently in debug mode. It may be monitored by external logic to cause a synchronous exit from debug mode of other modules (but not other CPUs) in the system. A debugger can place all of the CPU tap controllers in parallel, scan in a go+exit command with "no register selected", and sequence through the Update_DR state to cause any CPU currently in debug mode to exit. CPUs which are not it debug mode will ignore the command.

64.5.6 OnCE Controller and serial interface

The OnCE Controller contains the OnCE command register, the OnCE decoder, and the status/control register. The following figure is a block diagram of the OnCE controller. In operation, the OnCE Command register acts as the IR for the TAP controller, and all other OnCE resources are treated as data registers (DR) by the TAP controller. The Command register is loaded by serially shifting in or parallel loading commands during the TAP controller Shift-IR state, and is loaded during the Update-IR state. The Command register selects a resource to be accessed as a data register (DR) during the TAP controller Capture-DR, Shift-DR and Update-DR states.

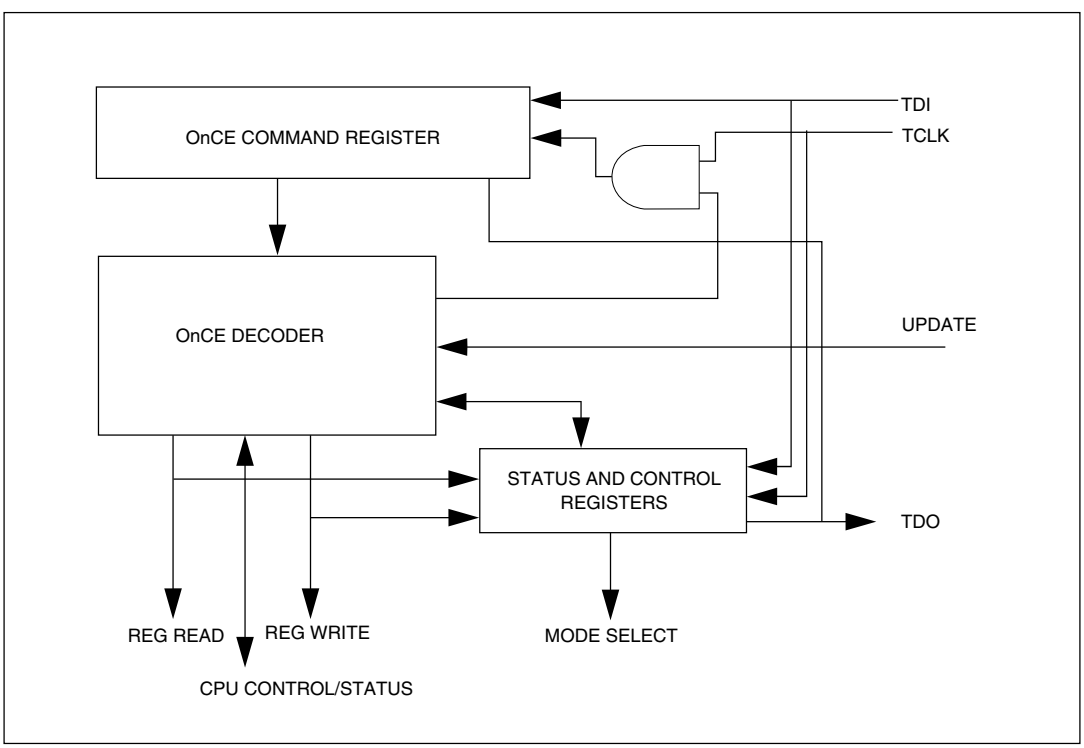


Figure 64-21. OnCE Controller and Serial Interface

64.5.6.1 OnCE Status Register

Status information regarding the state of the CPU is latched into the OnCE Status register when the OnCE controller state machine enters the Capture-IR state. When OnCE operation is enabled, this information is provided on the **j_tdo** output in serial fashion when the Shift_IR state is entered following a Capture-IR. Information is shifted out least significant bit first.

MCLK	ERR	0	RESET	HALT	STOP	DEBUG	WAIT	0	1
0	1	2	3	4	5	6	7	8	9

Figure 64-22. OnCE Status Register

The following table provides bit definitions for the Once Status Register.

Table 64-23. OnCE Status Register Bit Definitions

Bit(s)	Name	Description
0	MCLK	MCLK **m_clk** Status Bit 0 - Inactive state 1 - Active state This status bit reflects the logic level on the **jd_mclk_on** input signal after capture by **j_tclk**.
1	ERR	ERROR This bit is used to indicate that an error condition occurred during attempted execution of the last single-stepped instruction (GO+NoExit with CPUSCR or No Register Selected in OCMD), and that the instruction may not have been properly executed. This could occur if an Interrupt (all classes including External, Critical, machine check, Storage, Alignment, Program, etc.) occurred while attempting to perform the instruction single step. In this case, the CPUSCR will contain information related to the first instruction of the Interrupt handler, and no portion of the handler will have been executed.
3	RESET	RESET Mode This bit reflects the inverted logic level on the CPU **p_reset_b** input after capture by **j_tclk**.
4	HALT	HALT Mode This bit reflects the logic level on the CPU **p_halted** output after capture by **j_tclk**.
5	STOP	STOP Mode This bit reflects the logic level on the CPU **p_stopped** output after capture by **j_tclk**.
6	DEBUG	Debug Mode This bit is asserted once the CPU is in debug mode. It is negated once the CPU exits debug mode (even during a debug session)
7	WAIT	Waiting Mode This bit reflects the logic level on the CPU **p_waiting** output after capture by **j_tclk**.

Table continues on the next page...

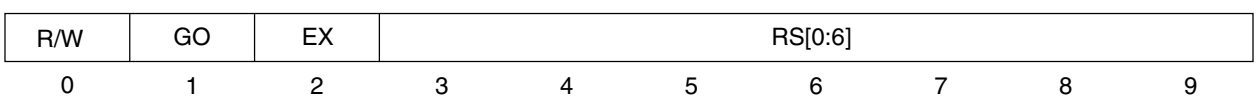
Table 64-23. OnCE Status Register Bit Definitions (continued)

Bit(s)	Name	Description
8	0	Reserved, set to 0 for 1149.1 compliance
9	1	Reserved, set to 1 for 1149.1 compliance

64.5.6.2 OnCE Command Register (OCMD)

The OnCE Command Register (OCMD) is a 10-bit shift register that receives its serial data from the TDI pin and serves as the instruction register (IR). It holds the 10-bit commands to be used as input for the OnCE Decoder. The Command Register is shown in Figure 64-23. The OCMD is updated when the TAP controller enters the Update-IR state. It contains fields for controlling access to a resource, as well as controlling single-step operation and exit from OnCE mode.

Although the OCMD is updated during the Update-IR TAP controller state, the corresponding resource is accessed in the DR scan sequence of the TAP controller, and as such, the Update-DR state must be transitioned through in order for an access to occur. In addition, the Update-DR state must also be transitioned through in order for the single-step and/or exit functionality to be performed, even though the command appears to have no data resource requirement associated with it.



Reset - 10'b1000000010 on assertion of **j_trst_b** or **m_por** , or while in the Test_Logic_Reset state

Figure 64-23. OnCE Command Register

Table 64-24 provides bit definitions for the Once Command Register.

Table 64-24. OnCE Command Register Bit Definitions

Bit(s)	Name	Description
0	R/W	Read/Write Command Bit The R/W bit specifies the direction of data transfer. The table below describes the options defined by the R/W bit. **Note:** The R/W bit generally ignored for read-only or write-only registers, although the PC FIFO pointer is only guaranteed to be update when R/W=1. In addition, it is ignored for all bypass operations. When performing writes, most registers are sampled in the Capture-DR state into a 32-bit shift register, and subsequently shifted out on **j_tdo** during the first 32 clocks of Shift-DR. 0 - Write the data associated with the command into the register specified by RS[0:6] 1 - Read the data contained in the register specified by RS[0:6]
1	GO	Go Go Command Bit If the GO bit is set, and the CPU is currently in debug mode, the CPU will execute the instruction which resides in the IR register in the CPUSCR. To execute the instruction, the processor leaves the debug mode, executes the instruction, and if the EX bit is cleared, returns to the debug mode immediately after executing the instruction. The processor goes on to normal operation if the EX bit is set, and no other debug request source is asserted. The GO command is executed only if the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the GO bit is ignored.The processor will leave the debug mode after the TAP controller Update-DR state is entered. On a GO+NoExit operation, returning to debug mode is treated as a debug event, thus exceptions such as machine checks and interrupts may take priority and prevent execution of the intended instruction. Debug firmware should mask these exceptions as appropriate. The OSR_{ERR} bit indicates such an occurrence. If the CPU is not currently in debug mode, then the GO command will be ignored. Note that asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See Exiting Debug Mode and Interrupt Blocking. 0 - Inactive (no action taken) 1 - Execute instruction in IR
2	EX	Exit Command Bit 0 - Remain in debug mode 1 - Leave debug mode If the EX bit is set, the processor will leave the debug mode and resume normal operation until another debug request is generated. The Exit command is executed only if the Go command is issued, and the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the EX bit is ignored. The processor will leave the debug mode after the TAP controller Update-DR state is entered. Note that if the DR bit in the OnCE control register is set or remains set, or if a bit in EDBSR0 is set and $EDBCR0_{EDM}$=1 (external debug mode is enabled), or if another debug request source is asserted, then the processor may return to the debug mode *without* execution of an instruction, even though the EX bit was set. Note that asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See Exiting Debug Mode and Interrupt Blocking.

Table continues on the next page...

Table 64-24. OnCE Command Register Bit Definitions (continued)

Bit(s)	Name	Description
3:9	RS	Register Select The Register Select bits define which register is source (destination) for the read (write) operation. Table 64-25 indicates the OnCE register addresses. Attempted writes to read-only registers are ignored.

Table 64-25 indicates the OnCE register addresses.

Table 64-25. OnCE Register Addressing

RS[0:6]	Register Selected
000 0000	Reserved
000 0001	Reserved
000 0010	JTAG ID (read-only)
000 0011–000 1111	Reserved
001 0000	CPU Scan Register (CPUSCR)
001 0001	No Register Selected (Bypass)
001 0010	OnCE Control Register (OCR)
001 0011	Reserved
001 0100–001 0111	Reserved
001 1000	Data Address Compare 3 (DAC3)
001 1001	Data Address Compare 4 (DAC4)
001 1010–001 1111	Reserved
010 0000	Instruction Address Compare 1 (IAC1)
010 0001	Instruction Address Compare 2 (IAC2)
010 0010	Instruction Address Compare 3 (IAC3)
010 0011	Instruction Address Compare 4 (IAC4)
010 0100	Data Address Compare 1 (DAC1)
010 0101	Data Address Compare 2 (DAC2)
010 0110	Data Value Compare 1 (DVC1) - *all 64 bits of the DVC register are accessed
010 0111	Data Value Compare 2 (DVC2) - *all 64 bits of the DVC register are accessed
010 1000	Instruction Address Compare 5 (IAC5)
010 1001	Instruction Address Compare 6 (IAC6)
010 1010	Instruction Address Compare 7 (IAC7)
010 1011	Instruction Address Compare 8 (IAC8)
010 1100	Debug Data Effective Address (DDEAR)
010 1101	External Debug Data Effective Address (EDDEAR)
010 1110	External Debug Control Register 0 (EDBCR0)
010 1111	External Debug Status Register 0 (EDBSR0)
011 0000	Debug Status Register (DBSR)
011 0001	Debug Control Register 0 (DBCR0)

Table continues on the next page...

Table 64-25. OnCE Register Addressing (continued)

RS[0:6]	Register Selected
011 0010	Debug Control Register 1 (DBCR1)
011 0011	Debug Control Register 2 (DBCR2)
011 0100	Reserved (do not access)
011 0101	Debug Control Register 4 (DBCR4)
011 0110	Debug Control Register 5 (DBCR5)
011 0111	Debug Control Register 6 (DBCR6)
011 1000	Debug Control Register 7 (DBCR7)
011 1001	Debug Control Register 8 (DBCR8)
011 1010	Reserved (do not access)
011 1011	Reserved (do not access)
011 1100	External Debug Status Register MASK 0 (EDBSRMSK0)
011 1101	Debug Data Acquisition Message Register (DDAM)
011 1110	Debug Event Control (DEVENT)
011 1111	External Debug Resource Allocation Control 0 (EDBRAC0)
100 0000	Reserved for Debug L1 Cache Control/Status 0 (DBL1CCSR0)
100 0001–110 1100	Reserved (do not access)
110 1101	MPU0 Control/Status 0 (MPU0CSR0)
110 1110	Performance Monitor Register Access
110 1111	Reserved for Shared Nexus Control Register Select
111 0000–111 1001	General Purpose register selects [0:9]
111 1010	Cache Debug Access Control Register (CDACNTL)
111 1011	Cache Debug Access Data Register (CDADATA)
111 1100	Nexus3-Access (<<<put in cross reference to Nexus 3 Module>>>)
111 1101	LSRL Select (see Test Specification)
111 1110	Enable_OnCE[1]
111 1111	Bypass

1. Causes assertion of the j_en_once_regsel output.

The Once Decoder receives as input the 10-bit command from the OCMD, and status signals from the processor, and generates all the strobes required for reading and writing the selected OnCE registers.

Single stepping of instructions is performed by placing the CPU in debug mode, scanning in appropriate information into the CPUSCR, and setting the Go bit (with the EX bit cleared) with the RS field indicating either the CPUSCR or No Register Selected. After executing a single instruction, the CPU will re-enter debug mode and await further commands. During single-stepping, exception conditions may occur if not properly masked by debug firmware (interrupts, machine checks, bus error conditions, etc.) and

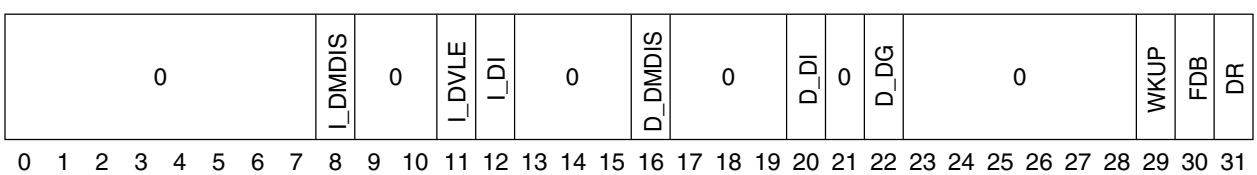
may prevent the desired instruction from being successfully executed. The OSR_{ERR} bit is set to indicate this condition. In these cases, values in the CPUSCR will correspond to the first instruction of the exception handler.

Additionally, the $EDBCR0_{EDM}$ bit is forced to '1' internally while single-stepping to prevent Debug events from generating Debug interrupts. Also, during a debug session, the DBSR register is frozen from updates due to debug events other than execution of a DNI-type instruction, regardless of EDBCR0EDM. DBSR may still be modified during a debug session via a single-stepped **mtspr** instruction, or via OnCE access.

If the CPU is not currently in debug mode, go+exit and go+noexit commands are ignored, although for a go+exit command with "No Register Selected", the j_ocmd_go_exit output will be asserted.

64.5.6.3 OnCE Control Register (OCR)

The OnCE Control Register is a 32-bit register used to force the core into debug mode and to enable / disable sections of the OnCE control logic. It also provides control over the MPU during a debug session (see MPU Operation During Debug). The control bits are read/write. These bits are only effective while OnCE is enabled (**jd_en_once** asserted). The OCR is shown in the following figure.



Reset - 0xo000_0000 on **m_por, j_trst_b,** or entering Test_logic_Reset state

Figure 64-24. OnCE Control Register

The following table provides bit definitions for the OnCE Control Register.

Table 64-26. OnCE Control Register Bit Definitions

Bit(s)	Name	Description
0:7	—	Reserved

Table continues on the next page...

Table 64-26. OnCE Control Register Bit Definitions (continued)

Bit(s)	Name	Description
8	I_DMDIS	Instruction Side Debug MPU Disable Control Bit (I_DMDIS) 0 - MPU not disabled for debug sessions 1 - MPU disabled for debug sessions This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Instruction Accesses. When enabled, the MPU functions normally. When disabled, for Instruction Accesses, the I bit is taken from the OCR bit I_DI. The SX and UX access permission control bits are set to'1' to allow full access. When disabled, no MPU exceptions are generated for Instruction accesses. External access errors can still occur. MPU entries with DEBUG=1 are not affected by this control bit.
9:10	—	Reserved
11	I_DVLE	Instruction Side Debug 'VLE' Attribute Bit (I_DVLE) This bit is used to provide the 'VLE' attribute bit to be used during a debug session. Note: this bit is ignored, since VLE is always enabled.
12	I_DI	Instruction Side Debug 'I' Attribute Bit (I_DI) This bit is used to provide the 'I' attribute bit to be used for Instruction accesses for Instruction accesses during a debug session.
13:15	—	Reserved
16	D_DMDIS	Data Side Debug MPU Disable Control Bit (D_DMDIS) 0 - MPU not disabled for debug sessions 1 - MPU disabled for debug sessions This bit may be used to control whether the MPU is enabled normally, or whether the MPU is disabled during a debug session for Data Accesses. When enabled, the MPU functions normally. When disabled, for Data Accesses, the MPU I and G bits are taken from the OCR bits D_DI and D_DG. The SR, SW, UR, and UW access permission control bits are set to'1' to allow full access. When disabled, no MPU exceptions are generated for Data accesses. External access errors can still occur. MPU entries with DEBUG=1 are not affected by this control bit.
17:19	—	Reserved
20	D_DI	Data Side Debug 'I' Attribute Bit (D_DI) This bit is used to provide the 'I' attribute bit to be used for Data accesses during a debug session.
21	—	Reserved
22	D_DG	Data Side Debug 'G' Attribute Bit (D_DG) This bit is used to provide the 'G' attribute bit to be used for Data accesses during a debug session.
23:28	—	Reserved
29	WKUP	Wakeup Request Bit (WKUP) This control bit may be used to force the **p_wakeup** output signal to be asserted. This control function may be used by debug firmware to request that the chip-level clock controller restore the **m_clk** input to normal operation regardless of whether the CPU is in a low power state to ensure that debug resources may be properly accessed by external hardware through scan sequences.

Table continues on the next page...

Table 64-26. OnCE Control Register Bit Definitions (continued)

Bit(s)	Name	Description
30	FDB	Force Breakpoint Debug Mode Bit (FDB) This control bit is used to determine whether the processor is operating in breakpoint debug enable mode or not. The processor may be placed in breakpoint debug enable mode by setting this bit. In breakpoint debug enable mode, execution of the '**bkpt**' pseudo- instruction will cause the processor to enter debug mode, as if the **jd_de_b** input had been asserted. This bit is qualified with $EDBCR0_{EDM}$, which must be set for FDB to take effect. Note that this bit has no effect on **e_dnh** or **se_dnh** instruction operation.
31	DR	CPU Debug Request Control Bit This control bit is used to unconditionally request the CPU to enter the Debug Mode. The CPU will indicate that Debug Mode has been entered via the data scanned out in the shift-IR state. 0 - No Debug Mode request 1 - Unconditional Debug Mode request When the DR bit is set the processor will enter Debug mode at the next instruction boundary.

64.5.7 Access to Debug Resources

Resources contained in the OnCE Module which do not require the processor core to be halted for access may be accessed while the core is running, and will not interfere with processor execution. Accesses to other resources such as the CPUSCR require the core to be placed in debug mode to avoid synchronization hazards. Debug firmware may ensure that it is safe to access these resources by determining the state of the core prior to access. Note that a scan operation to update the CPUSCR is required prior to exiting debug mode if debug mode has been entered.

Some cases of write accesses other than accesses to the OnCE Command and Control registers, or the EDM bit of DBCR0 require **m_clk** to be running for proper operation. The OnCE control register provides a means of signaling this need to a system level clock control module via the OCR_{WKUP} control bit.

In addition, since the CPU may cause multiple bits of certain registers to change state, reads of certain registers while the CPU is running (DBSR, etc.) may not have consistent bit settings unless read twice with the same value indicated. In order to guarantee that the contents are consistent, the CPU should be placed into debug mode, or multiple reads should be performed until consistent values have been obtained on consecutive reads.

The following table provides a list of access requirements for OnCE registers.

Table 64-27. OnCE Register Access Requirements

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires EDBCR0 $_{EDM}$ = 1 for write access	Requires m_clk active for Write Access	Requires CPU to be halted for Read Access	Requires CPU to be halted for Write Access	
Enable_OnCE	N	N	N	N	—	
Bypass	N	N	N	N	N	
CPUSCR	Y	Y	Y	Y	Y	
DAC1–4	Y	Y	Y	N	*[1, 1]	
DBCR0	Y	Y	Y	N	*[1]	*EDBCR0$_{EDM}$ access only requires **jd_en_once** asserted
DBCR1–8	Y	Y	Y	N	*[1]	
DEVENT	Y	Y	Y	N	*[1]	
EDBRAC0 (DBERC0)	Y	N	Y	N	*[1]	
DBSR	Y	Y	Y	N[2, 2]	*[1]	
EDBCR0	Y	N	N	N	N	
EDBSR0	Y	N	Y	N	N	
EDBSRMSK0	Y	N	N	N	N	
IAC1–8	Y	Y	Y	N	*[1]	
JTAG ID	N	N	—	N	—	Read-only
MPU0CSR0	Y	N	Y	N	N	
OCR	Y	N	N	N	N	
OSR	Y	N	—	N	—	Read-only, accessed by scanning out IR while **jd_en_once** is asserted
Cache Debug Access Control (CDACNTL)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Cache Debug Access Data (CDADATA)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Nexus3-Access	Y	N	N	N	N	
PMR-Access	Y	N	N	N	N	
PMR Registers	Y	Y	Y	N	N	
External GPRs	Y	N	N	N	N	
LSRL Select	Y	N	?	?	?	System Test logic implementation determines LSRL functionality

1. Writes to these registers while the CPU is running may have unpredictable results due to the pipelined nature of operation, and the fact that updates are not synchronized to a particular clock, instruction, or bus cycle boundary, therefore it is strongly recommended to ensure the processor is first placed into debug mode before updates to these registers are performed.
2. Reads of these registers while the CPU is running may not give data that is self-consistent due to synchronization across clock domains.

64.5.8 Methods of Entering Debug Mode

The OnCE Status Register indicates that the CPU has entered the debug mode via the DEBUG status bit. The following sections describe how Debug Mode is entered assuming the OnCE circuitry has been enabled. OnCE operation is enabled by the assertion of the **jd_en_once** input See OnCE Enable (jd_en_once).

64.5.8.1 External Debug Request During RESET

Holding the **jd_de_b** signal asserted during the assertion of **p_reset_b,** and continuing to hold it asserted following the negation of **p_reset_b** causes the core to enter Debug Mode. After receiving an acknowledge via the OnCE Status Register DEBUG bit, the external command controller should negate the **jd_de_b** signal before sending the first command. Note that in this case the core does not execute an instruction before entering Debug Mode, although the first instruction to be executed will be fetched prior to entering Debug Mode in order to properly initialize the CPUSCR.

64.5.8.2 Debug Request During RESET

Asserting a debug request by setting the DR bit in the OCR during the assertion of **p_reset_b** and then negating **p_reset_b** causes the chip to enter debug mode. In this case the CPU will fetch the first instruction of the reset exception handler in order to properly initialize the CPUSCR, but does not execute an instruction before entering debug mode.

64.5.8.3 Debug Request During Normal Activity

Asserting a debug request by setting the DR bit in the OCR during normal chip activity causes the chip to finish the execution of the current instruction and then enter the debug mode. Note that in this case the chip completes the execution of the current instruction and stops after the newly fetched instruction enters the CPU instruction register. This process is the same for any newly fetched instruction including instructions fetched by the interrupt processing, or those that will be aborted by the interrupt processing.

64.5.8.4 Debug Request During Waiting, Halted or Stopped State

Asserting a debug request by setting the DR bit in the OCR when the chip is in the Waiting state (**p_waiting** asserted), Halted state (**p_halted** asserted) or Stopped state (**p_stopped** asserted) causes the CPU to exit the state and enter the debug mode once the CPU clock **m_clk** has been restored. Note that in this case, the CPU will negate the **p_waiting, p_halted** and **p_stopped** outputs. Once the debug session has ended, the CPU will return to the state it was in prior to entering debug mode.

To signal the chip-level clock generator to re-enable **m_clk**, the **p_wakeup** output will be asserted whenever the debug block is asserting a debug request to the CPU due to OCR_{DR} being set, or **jd_de_b** assertion, and will remain set from then until the debug session ends (**jd_debug_b** goes from asserted to negated). In addition, the status of the **jd_mclk_on** input (after synchronization to the **j_tclk** clock domain) may be sampled along with other status bits from the **j_tdo** outputs during the Shift_IR TAP controller state. This status may be used if necessary by external debug firmware to ensure proper scan sequences occur to registers in the **m_clk** clock domain.

64.5.8.5 Software Request During Normal Activity

Upon executing a '**bkpt**' pseudo-instruction (defined to be an all 0's instruction opcode) when the OCR register's (FDB) bit is set (debug mode enable control bit is true), and $EDBCR0_{EDM}$=1, the CPU enters the debug mode after the instruction following the '**bkpt**' pseudo-instruction has entered the instruction register.

64.5.8.6 Debug Notify Halt Instructions

The **e_dnh** and **se_dnh** instructions allow software to transition the core from a running state to a debug halted state if enabled by $EDBCR0_{DNH_EN}$, and provide the external debugger with bits reserved in the instruction itself to pass additional information. Entry into debug mode is ***not*** conditioned on $EDBCR0_{EDM}$, allowing for debug of software debug handlers as well as other software. When the CPU enters a debug halted state due to a **e_dnh** or **se_dnh** instruction, the instruction will be stored in the CPUSCR[IR] portion, and the CPUSCR[PC] value will point to the instruction. The external debugger should update the CPUSCR prior to exiting the debug halted state to point past the **e_dnh** or **se_dnh** instruction.

64.5.9 CPU Status and Control Scan Chain Register (CPUSCR)

A number of on-chip registers store the CPU pipeline status and are configured in a single scan chain for access by the OnCE controller. The CPUSCR register contains these processor resources, which are used to restore the pipeline and resume normal chip activity upon return from the debug mode, as well as a mechanism for the emulator software to access processor and memory contents. The following figure shows the block diagram of the pipeline information registers contained in the CPUSCR. Once debug mode has been entered, it is required to scan in and update this register prior to exiting debug mode.

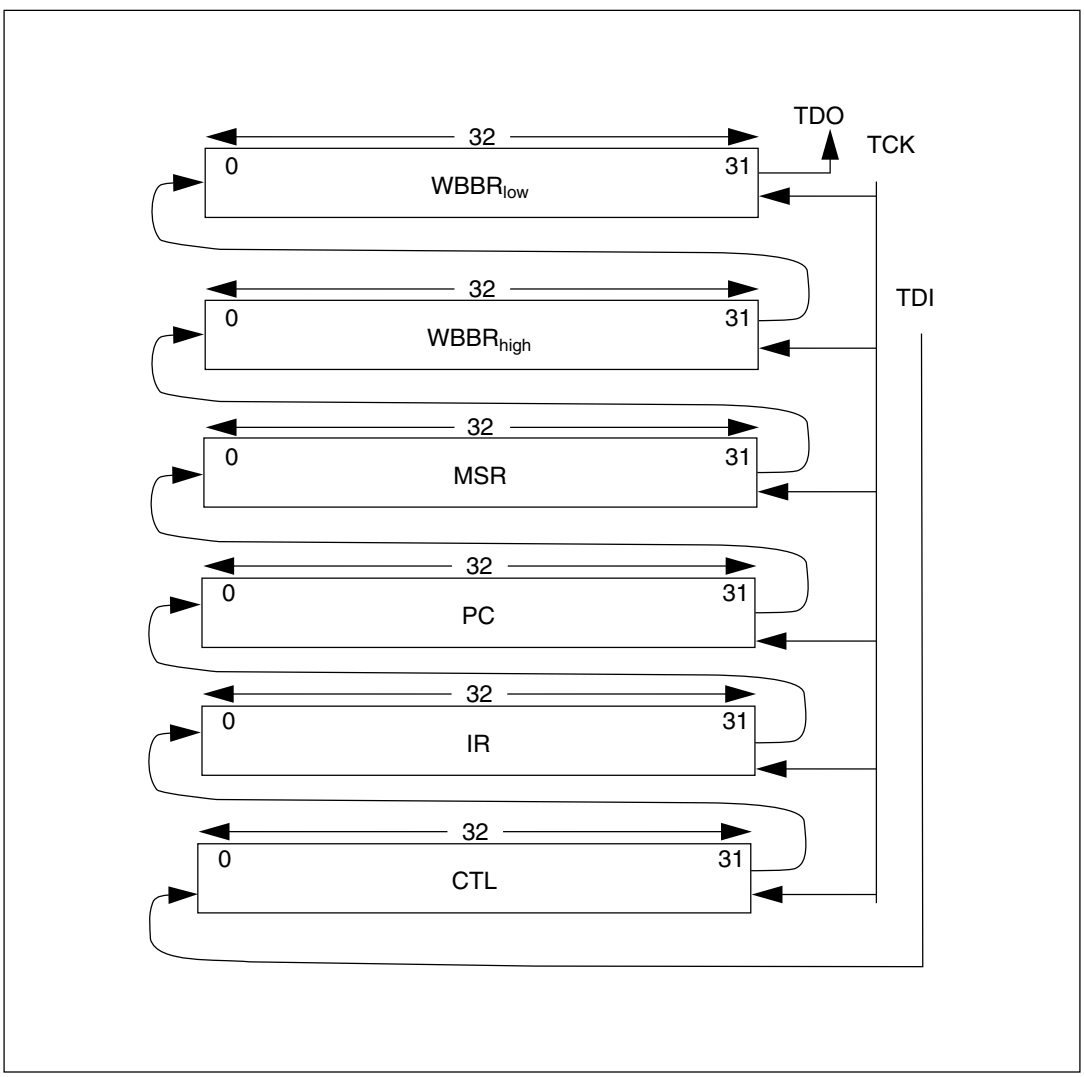


Figure 64-25. CPU Scan Chain Register (CPUSCR)

64.5.9.1 Instruction Register (IR)

The Instruction Register (IR) provides a mechanism for controlling the debug session by serving as a means for forcing in selected instructions, and then causing them to be executed in a controlled manner by the debug control block. The opcode of the next instruction to be executed when entering debug mode is contained in this register when the scan-out of this chain begins. This value should be saved for later restoration if continuation of the normal instruction stream is desired.

On scan-in, in preparation for exiting debug mode, this register is filled with an instruction opcode selected by debug control software. By selecting appropriate instructions and controlling the execution of those instructions, the results of execution may be used to examine or change memory locations and processor registers. The debug control module external to the processor core will control execution by providing a single-step capability. Once the debug session is complete and normal processing is to be resumed, this register may be loaded with the value originally scanned out.

64.5.9.2 Control State Register (CTL)

The Control State Register (CTL) is a 32-bit register that stores the value of certain internal CPU state variables before the debug mode is entered. This register is affected by the operations performed during the debug session and should normally be restored by the external command controller when returning to normal mode. In addition to saved internal state variables, two of the bits are used by emulation firmware to control the debug process. In certain circumstances, emulation firmware must modify the content of this register as well as the PC and IR values in the CPUSCR before exiting debug mode. These cases are described below.

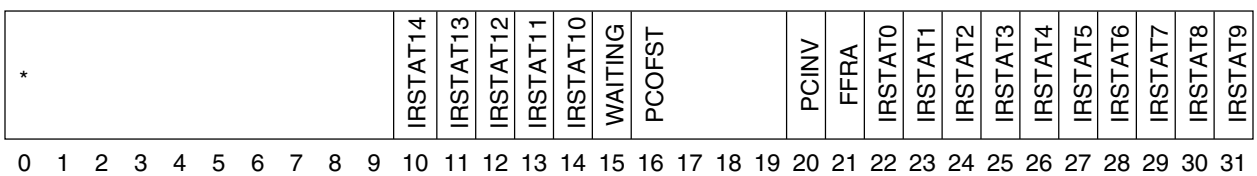


Figure 64-26. Control State Register (CTL)

WAITING — WAITING State Status

This bit indicates whether the CPU was in the waiting state prior to entering debug mode. If set, the CPU was in the waiting state. Upon exiting a debug session, the value of this bit in the restored CPUSCR will determine whether the CPU re-enters the waiting state on a go+exit.

0- CPU was not in the waiting state when debug mode was entered

1- CPU was in the waiting state when debug mode was entered

PCOFST — PC Offset Field

Table continues on the next page...

This field indicates whether the value in the PC portion of the CPUSCR must be adjusted prior to exiting debug mode. Due to the pipelined nature of the CPU, the PC value must be backed-up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored into the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction instead of the original IR value, other wise the original value of IR should be restored. (But see PCINV which overrides this field)

0000 - No correction required.

0001 - Subtract 0x04 from PC.

0010 - Subtract 0x08 from PC.

0011 - Subtract 0x0C from PC.

0100 - Subtract 0x10 from PC.

0101 - Subtract 0x14 from PC.

all other encodings are reserved

* — Internal State Bits

These control bits represent internal processor state and should be restored to their original value after a debug session is completed, i.e when a OnCE command is issued with the GO and EX bits set and not ignored. When performing instruction execution during a debug session. See OnCE Debug Output (jd_debug_b), which is not part of the normal program execution flow, these bits should be set to a 0.

PCINV — PC and IR Invalid Status Bit

This status bit indicates that the values in the IR and PC portions of the CPUSCR are invalid. Exiting debug mode with the saved values in the PC and IR will have unpredictable results. Debug firmware should initialize the PC and IR values in the CPUSCR with desired values prior to exiting debug mode if this bit was set when debug mode was initially entered.

0= No error condition exists.

1= Error condition exists. PC and IR are corrupted.

FFRA— Feed Forward RA Operand Bit

This control bit causes the content of the $WBBR_{lo}$ to be used as the RA operand value (RS for logical, mtspr, mtdcr, cntlzw, and shift operations, RX for VLE se_ instructions, RT for e_{logical_op}2i type instructions, RB for evaddiw, evsubifw, and the value to use as the PC for calculating the LR update value for branch with link type instructions) of the first instruction to be executed following an update of the CPUSCR. For most LSP instructions using rA||rB as a 64-bit source operand, $WBBR_{hi, lo}$ is used to supply the 64-bit source value.

This allows the debug firmware to update processor registers — initialize the $WBBR_{lo}$ with the desired value, set the FFRA bit, and execute a ori Rx,Rx,0 instruction to the desired register.

Note: not all instructions support using the FFRA control. FFRA is mainly intended for use with the ori instruction to allow the debugger to write to a GPR. Support for other instructions is implementation-dependent.

0= No action.

1= Content of $WBBR_{lo}$ or $WBBR_{hi,lo}$ used as operand value

IRStat0 — IR Status Bit 0

This control bit indicates a TEA status for the IR.

0= No TEA occurred on the fetch of this instruction.

1= TEA occurred on the fetch of this instruction.

IRStat1 — IR Status Bit 1

This control bit is reserved.

IRStat2 — IR Status Bit 2

Table continues on the next page...

This control bit indicates an Instruction Address Compare 1 event status for the IR.

0= No Instruction Address Compare event 1 occurred on the fetch of this instruction.

1= An Instruction Address Compare event 1 occurred on the fetch of this instruction.

IRStat3 — IR Status Bit 3

This control bit indicates an Instruction Address Compare 2 event status for the IR.

0= No Instruction Address Compare 2 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 2 event occurred on the fetch of this instruction.

IRStat4 — IR Status Bit 4

This control bit indicates an Instruction Address Compare 3 event status for the IR.

0= No Instruction Address Compare event 3 occurred on the fetch of this instruction.

1= An Instruction Address Compare event 3 occurred on the fetch of this instruction.

IRStat5 — IR Status Bit 5

This control bit indicates an Instruction Address Compare 4 event status for the IR.

0= No Instruction Address Compare 4 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 4 event occurred on the fetch of this instruction.

IRStat6 — IR Status Bit 6

This control bit indicates a Parity Error status for the IR.

0= No Parity Error occurred on the fetch of this instruction.

1= Parity Error occurred on the fetch of this instruction from the I-Cache or the IMEM.

IRStat7 — IR Status Bit 7

This control bit indicates a Precise External Termination Error status for the IR.

0= No Precise External Termination Error occurred on the fetch of this instruction.

1= A Precise External Termination Error occurred on the fetch of this instruction.

IRStat8 — IR Status Bit 8

This control bit indicates the PowerISA VLE status for the IR.

0= IR contains a BookE instruction. (unused encoding)

1= IR contains a PowerISA VLE instruction, aligned in the Most Significant Portion of IR if 16-bit.

- this bit should not be modified by the debugger, otherwise the resulting operation is boundedly undefined. It should always indicate VLE (=1).

IRStat9 — IR Status Bit 9

This control bit is reserved.

IRStat10 — IR Status Bit 10

This control bit indicates an Instruction Address Compare 5 event status for the IR.

0= No Instruction Address Compare 5 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 5 event occurred on the fetch of this instruction.

IRStat11 — IR Status Bit 11

This control bit indicates an Instruction Address Compare 6 event status for the IR.

0= No Instruction Address Compare 6 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 6 event occurred on the fetch of this instruction.

IRStat12 — IR Status Bit 12

This control bit indicates an Instruction Address Compare 7 event status for the IR.

Table continues on the next page...

0= No Instruction Address Compare 7 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 7 event occurred on the fetch of this instruction.

IRStat13 — IR Status Bit 13

This control bit indicates an Instruction Address Compare 8 event status for the IR.

0= No Instruction Address Compare 8 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 8 event occurred on the fetch of this instruction.

IRStat14 — IR Status Bit 14

This control bit indicates an MPU Instruction Address Compare event status for the IR.

0= No MPU Instruction Address Compare event occurred on the fetch of this instruction.

1= An MPU Instruction Address Compare event occurred on the fetch of this instruction.

Emulation firmware should modify the content of the CTL, PC, and IR values in the CPUSCR during execution of debug related instructions as well as just prior to exiting debug with a go+exit command. During the debug session, the CTL register should be written with the FFRA bit set as appropriate, and all other bit set to '0', and the IR set to the value of the desired instruction to be executed. IRStat8 will be used to determine the type of instruction present in the IR.

Just prior to exiting debug mode with a go+exit, the PCINV status bit which was originally present when debug mode was first entered should be tested, and if set, the PC and IR initialized for performing whatever recovery sequence is appropriate for a faulted exception vector fetch. If the PCINV bit is cleared, then the PCOFST bits should be examined to determine whether the PC value must be adjusted. Due to the pipelined nature of the CPU, the PC value must be backed-up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored in to the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction (such as **ori r0,r0,0**) instead of the original IR value, otherwise the original value of IR should be restored. Note that when a correction is made to the PC value, it will generally point to the last completed instruction, although that instruction will not be re-executed. The nop instruction is executed instead, and instruction fetch and execution will resume at location PC+4. IRStat8 will be used to determine the type of instruction present in the IR, thus should be cleared in this case. Note that debug events which may occur on the nop (ICMP) will be generated if enabled.

For the CTL register, the internal state bits should be restored to their original value. The IRStatus bits should be set to '0's if the PC was adjusted. If no PC adjustment was performed, emulation firmware should determine whether other IRStat flags should be set to '0' to avoid re-entry into debug mode for an instruction breakpoint request. Upon exiting debug mode with go+exit, if one of these bits is set, debug mode will be re-entered prior to any further instruction execution.

64.5.9.3 Program Counter Register (PC)

The PC is a 32-bit register that stores the value of the program counter which was present when the chip entered the debug mode. It is affected by the operations performed during the debug mode and must be restored by the external command controller when the CPU returns to normal mode. PC normally points to the instruction contained in the IR portion of CPUSCR. If debug firmware wishes to redirect program flow to an arbitrary location, the PC and IR should be initialized to correspond to the first instruction to be executed upon resumption of normal processing. Alternatively, the IR may be set to a nop and the PC set to point to the location prior to the location at which it is desired to redirect flow to. On exiting debug mode, the nop will be executed, and instruction fetch and execution will resume at PC+4.

64.5.9.4 Write-Back Bus Register ($WBBR_{low}$, $WBBR_{high}$)

WBBR is used as a means of passing operand information between the CPU and the external command controller. Whenever the external command controller needs to read the contents of a register or memory location, it will force the chip to execute an instruction that brings that information to WBBR. $WBBR_{low}$ holds the 32-bit result of most instructions including load data returned for a load or load with update instruction. For LSP instructions which generate 64-bit results, $WBBR_{low}$ holds the low-order 32 bits of the result. $WBBR_{high}$ holds the updated effective address calculated by a load with update instruction. For LSP instructions which generate 64-bit results, $WBBR_{high}$ holds the high-order 32 bits of the result. It is undefined for other instructions.

As an example, to read the 32 bits of processor register **r1**, an **ori r1,r1,0** instruction is executed, and the result value of the instruction will be latched into $WBBR_{low}$. The contents of $WBBR_{low}$ can then be delivered serially to the external command controller. To update a processor resource, this register is initialized with a data value to be written, and an **ori** instruction is executed which uses this value as a substitute data value. The Control State register FFRA bit forces the value of the $WBBR_{low}$ to be substituted for the normal RS source value of the **ori** instruction, thus allowing updates to processor registers to be performed. (Refer to Control State Register (CTL) for more detail on the CTL_{FFRA} bit.)

$WBBR_{low}$ and $WBBR_{high}$ are generally undefined on instructions which do not writeback a result, and due to control issues are not defined on **lmw** or branch instructions as well.

64.5.9.5 Machine State Register (MSR)

The MSR is a 32-bit register used to read/write the Machine State Register. Whenever the external command controller needs to save or modify the contents of the Machine State Register, this register is used.This register is affected by the operations performed during the debug mode and must be restored by the external command controller when returning to normal mode.

64.5.9.6 Exiting Debug Mode and Interrupt Blocking

When exiting debug mode with a Go+Exit, "asynchronous" interrupts are blocked until the first instruction to be executed begins execution. This includes External and Critical input, NMI, and machine check interrupts. Asynchronous debug interrupts are not blocked however, and the CPU will re-enter debug mode without executing an instruction following Go+Exit, although it may fetch an instruction and discard it. Exceptions due to an illegal instruction or error flags set within the CPUSCR CTL register are not blocked, since they apply to the instruction in the CPUSCR IR.

64.5.10 Reserved Registers (Reserved)

The Reserved Registers are used to control various test control logic. These registers are not intended for customer use. To preclude device and/or system damage, these registers should not be accessed.

64.6 Watchpoint Support

supports the generation and signalling of watchpoints when operating in internal debug mode ($DBCR0_{IDM}$=1) or in external debug mode ($EDBCR0_{EDM}$=1). Watchpoints are indicated with a dedicated set of interface signals. The **jd_watchpt[0:31]** output signals are used to indicate that a watchpoint has occurred. Certain watchpoints however (DEVENT-based, DTC-based, and Performance Monitor watchpoints) are not qualified with $EDBCR0_{EDM}$ or $DBCR0_{IDM}$.

Each debug address compare function (IAC1–8, DAC1–4), as well as other event types are capable of triggering a watchpoint output. The DBCRx control fields are used to configure watchpoints, regardless of whether events are enabled in DBCR0. Watchpoints may occur whenever an associated event would have been posted in the Debug Status Register if enabled. No explicit enable bits are provided for watchpoints; they are always enabled by definition, although the data address compare watchpoints may be controlled

for read and write accesses via configuration fields in DBCR4, 7, and 9. During a debug session, debug events (other than PMI, DEVT1 and DEVT2) with a corresponding DBSR bit are blocked from asserting a watchpoint. The Performance Monitor, DEVNT-based and DTC-based watchpoints are not blocked during a debug session. If not desired, for address-based events the base address values for these events may be programmed to an unused system address. MSR_{DE} has no effect on watchpoint generation.

External logic may monitor the assertion of these signals for debugging or triggering purposes. Watchpoints are signaled in the clock cycle following the occurrence of the actual event. The Nexus3 module also monitors assertion of a portion of these output signals (**jd_watchpt[0:31]**) for various development control purposes. (See the "Watchpoint Trace Messaging" section in the Nexus 3 Module chapter.)

Note that these signals are **m_clk** domain signals, not **j_tclk** domain signals.

The following table shows the **jd_watchpt[0:31]** output signal assignments to various debug events.

Table 64-28. Watchpoint Output Signal Assignments

Signal Name	Type	Description
jd_watchpt[0]	IAC1	Instruction Address Compare 1 watchpoint
		Asserted whenever an IAC1 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[1]	IAC2	Instruction Address Compare 2 watchpoint
		Asserted whenever an IAC2 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[2]	IAC3	Instruction Address Compare 3 watchpoint
		Asserted whenever an IAC3 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[3]	IAC4	Instruction Address Compare 4 watchpoint
		Asserted whenever an IAC4 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[4]	DAC1	Data Address Compare 1 watchpoint
		Asserted whenever a DAC1 compare occurs regardless of being enabled to set DBSR status, based on $DBCR4_{DAC1CFG}$ settings
jd_watchpt[5]	DAC2	Data Address Compare 2 watchpoint
		Asserted whenever a DAC2 compare occurs regardless of being enabled to set DBSR status, based on $DBCR4_{DAC2CFG}$ settings
jd_watchpt[6]	DAC3	Data Address Compare 3 watchpoint
		Asserted whenever a DAC3 compare occurs regardless of being enabled to set DBSR status, based on $DBCR7_{DAC3CFG}$ settings
jd_watchpt[7]	DAC4	Data Address Compare 4 watchpoint
		Asserted whenever a DAC4 compare occurs regardless of being enabled to set DBSR status, based on $DBCR7_{DAC4CFG}$ settings

Table continues on the next page...

Table 64-28. Watchpoint Output Signal Assignments (continued)

Signal Name	Type	Description
jd_watchpt[8]	IAC5	Instruction Address Compare 5 watchpoint
		Asserted whenever an IAC5 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[9]	IAC6	Instruction Address Compare 6 watchpoint
		Asserted whenever an IAC6 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[10]	DEVT1	Debug Event Input 1 watchpoint
		Asserted whenever a DEVT1 debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[11]	DEVT2	Debug Event Input 2 watchpoint
		Asserted whenever a DEVT2 debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[12]	DEVNT0	Debug Event Output 0 watchpoint
		Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[12]
jd_watchpt[13]	DEVNT1	Debug Event Output 1 watchpoint
		Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[13]
jd_watchpt[14]	IAC7	Instruction Address Compare 7 watchpoint
		Asserted whenever an IAC7 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[15]	IAC8	Instruction Address Compare 8 watchpoint
		Asserted whenever an IAC8 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[16]	IRPT	Interrupt watchpoint
		Asserted whenever an IRPT debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[17]	RET	Return watchpoint
		Asserted whenever a RET debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[18]	CIRPT	Critical Interrupt watchpoint
		Asserted whenever a CIRPT debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[19]	CRET	Critical Return watchpoint
		Asserted whenever a CRET debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[20]	DEVNT2	Debug Event Output 2 watchpoint
		Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[20]
jd_watchpt[21]	DEVNT3	Debug Event Output 3 watchpoint
		Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[21]

Table continues on the next page...

Table 64-28. Watchpoint Output Signal Assignments (continued)

Signal Name	Type	Description
jd_watchpt[22]	PMEVENT	Performance Monitor Event input watchpoint
		Asserted whenever **p_pm_event** transitions from a '0' to a '1' while **m_clk** is running
jd_watchpt[23]	PMC0	Performance Monitor Counter 0 watchpoint
		Asserted whenever PMC0 triggers an event based on $PMLCa0_{PMP}$
jd_watchpt[24]	PMC1	Performance Monitor Counter 1 watchpoint
		Asserted whenever PMC1 triggers an event based on $PMLCa1_{PMP}$
jd_watchpt[25]	PMC2	Performance Monitor Counter 2 watchpoint
		Asserted whenever PMC2 triggers an event based on $PMLCa2_{PMP}$
jd_watchpt[26]	PMC3	Performance Monitor Counter 3 watchpoint
		Asserted whenever PMC3 triggers an event based on $PMLCa3_{PMP}$
jd_watchpt[27]	MPU	Memory Protection Unit watchpoint
		Asserted whenever a the MPU generates a debug event based on region descriptor matches with DEBUG control enabled.
jd_watchpt[28]	TRAP	TRAP watchpoint
		Asserted whenever an TRAP debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[29]	DTC1	Data Trace Control Range 1 watchpoint
		Asserted whenever an access meets the conditions for DTC Range 1
jd_watchpt[30]	DTC2	Data Trace Control Range 2 watchpoint
		Asserted whenever an access meets the conditions for DTC Range 2
jd_watchpt[31]	DTC3	Data Trace Control Range 3 watchpoint
		Asserted whenever an access meets the conditions for DTC Range 3

64.7 MPU Operation During Debug

A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may substitute default values obtained from the OnCE Control Register for Access Attribute (I, G) bits. Refer to the bit definitions in the OCR (OnCE Control Register (OCR) for more detail.

Normal operation of the MPU may be modified during a 'debug session' via the OnCE Control Register (OCR). A debug session begins when the CPU initially enters debug mode, and ends when a OnCE command with GO+EXIT is executed, releasing the CPU for normal operation. If desired during a debug session, the debug firmware may disable

the MPU protection mechanism and may substitute default values for the Access Protection (UX, UR, UW, SX, SR, SW) bits, and values obtained from the OnCE Control Register for Memory Attributes (I, G) bits normally provided by a matching MPU entry.

When disabled during a debug session, no MPU-related storage interrupt conditions will occur. If the debugger desires to use the normal protection mechanism, the MPU may be left enabled in the OnCE OCR, and normal protections provided by the MPU (including the possibility of a storage interrupt) will remain in effect.

The OCR control bits are used when debug mode is entered. Refer to the bit definitions in the OCR (OnCE Control Register (OCR) for more detail. When the MPU is disabled for instruction accesses (OCR_{I_DMDIS}) or for data accesses (OCR_{D_DMDIS}), substituted access attribute bits will control operation on respective accesses initiated during debug.

64.8 Cache Array Access During Debug

The cache arrays may be read and written during debug mode via the CDACNTL and CDADATA debug registers.

64.9 Basic Steps for Enabling, Using, and Exiting External Debug Mode

The following steps show one possible scenario for a debugger wishing to use the external debug facilities. *This simplified flow is intended to illustrate basic operations, but does not cover all potential methods in depth.*

Enabling External Debug Mode and initializing Debug registers

- The debugger should ensure that the **jd_en_once** control signal is asserted in order to enable OnCE operation
- Select the OCR and write a value to it in which OCR_{DR}, OCR_{WKUP}, are set to '1'. The tap controller must step through the proper states as outlined earlier. This step will place the CPU in a debug state in which it is halted and awaiting single-step commands or a release to normal mode
- Scan out the value of the OSR to determine that the CPU clock is running and the CPU has entered the Debug state. This can be done in conjunction with a Read of the CPUSCR. The OSR is shifted out during the Shift_IR state. The CPUSCR will be shifted out during the Shift_DR state. The debugger should save the scanned-out value of CPUSCR for later restoration.

- Select the DBCR0 register and update it with the $EDBCR0_{EDM}$ bit set
- Clear the DBSR status bits
- Write appropriate values to the DBCR0–8, IAC, and DAC registers. Note that the initial write to DBCR0 will only affect the EDM bit, so the remaining portion of the register must now be initialized, keeping the EDM bit set

At this point the system is ready to commence debug operations. Depending on the desired operation, different steps must occur.

- Optionally, ensure that the values entered into the MSR portion of the CPUSCR during the following steps cause interrupt to be disabled (clearing MSR_{EE} and MSR_{CE}). This will ensure that external interrupt sources do not cause single-step errors.

To single-step the CPU:

- debugger scans in either a new or a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Noexit OnCE Command value.
- The debugger scans out the OSR with "no-register selected", Go cleared, and determines that the PCU has re-entered the Debug state and that no ERR condition occurred

To return the CPU to normal operation (without disabling external debug mode)

- The OCR_{DR} control bit should be cleared, leaving the OCR_{WKUP} bit set
- The debugger restores the CPUSCR with a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared

To exit External Debug Mode

- The debugger should place the CPU in the debug state via the OCR_{DR} with OCR_{WKUP} asserted, scanning out and saving the CPUSCR
- The debugger should write the DBCR0–8 registers as needed, likely clearing every enable except the $EDBCR0_{EDM}$ bit
- The debugger should write the DBSR to a cleared state

- The debugger should re-write the DBCR0 with all bits including EDM cleared
- The debugger should clear the OCR_{DR} bit
- The debugger restores the CPUSCR with the previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared

Note

These steps are meant by way of examples, and are not meant to be an exact template for debugger operation.

Chapter 65
Debug and Calibration Interface (DCI)

65.1 Introduction

The Debug and Calibration Interface (DCI) module provides debug and calibration features for the MCU. It includes a standard IEEE 1149.1/1149.7 compatible JTAG interface which is used to connect with an external JTAG tool using a low frequency clock interface.

It also features a software based debug mode which can be controlled by the MCU without using an external tool. Additionally, the DCI provides features for debug control using break-points and synchronous restart of the cores. This chapter describes the DCI features for the Production device (PD) only.

65.1.1 Features

The DCI features are the following:

- Debug mode enable control for connected tool or software
- IEEE 1149.1 and IEEE 1149.7 controllers
- Debug break and cross-triggering control
- Synchronous restart control for CPU(s) when exiting debug mode
- Tool hot plug capability
- Security access control
- Debug reset control

65.1.2 Overview

The DCI can take the following different inputs for its JTAG signals:

- P.JTAG—JTAG signals coming from the Production device JTAG pads. This is the standard 5-pin JTAG interface and is the default operating mode.
- M.JTAG—JTAG signals generated by JTAGM module using software debug mode.
    - In software debug mode, there is no tool, and debug software running on the MCU generates the JTAG packets and writes them to JTAGM using special debug registers in the JTAGM module.

By default the DCI operates in standard 5-pin JTAG mode (IEEE 1149.1). It can be configured in 3-pin reduced pin JTAG mode (IEEE 1149.7) after starting from standard 5-pin JTAG mode.

When reduced pin JTAG mode (IEEE 1149.7) is configured TMSC pin is configured in bidirectional mode during operation.

The DCI also provides a centralized break control for system IPs and cores, which can be controlled by an external tool as well as the internal debug peripherals such as timers/ SPU, etc. The DCI also provides the Event Out ($\overline{\text{EVTO}}$) signals to the debug tool in case of the occurrence of any internal debug event.

This figure shows the DCI block diagram.

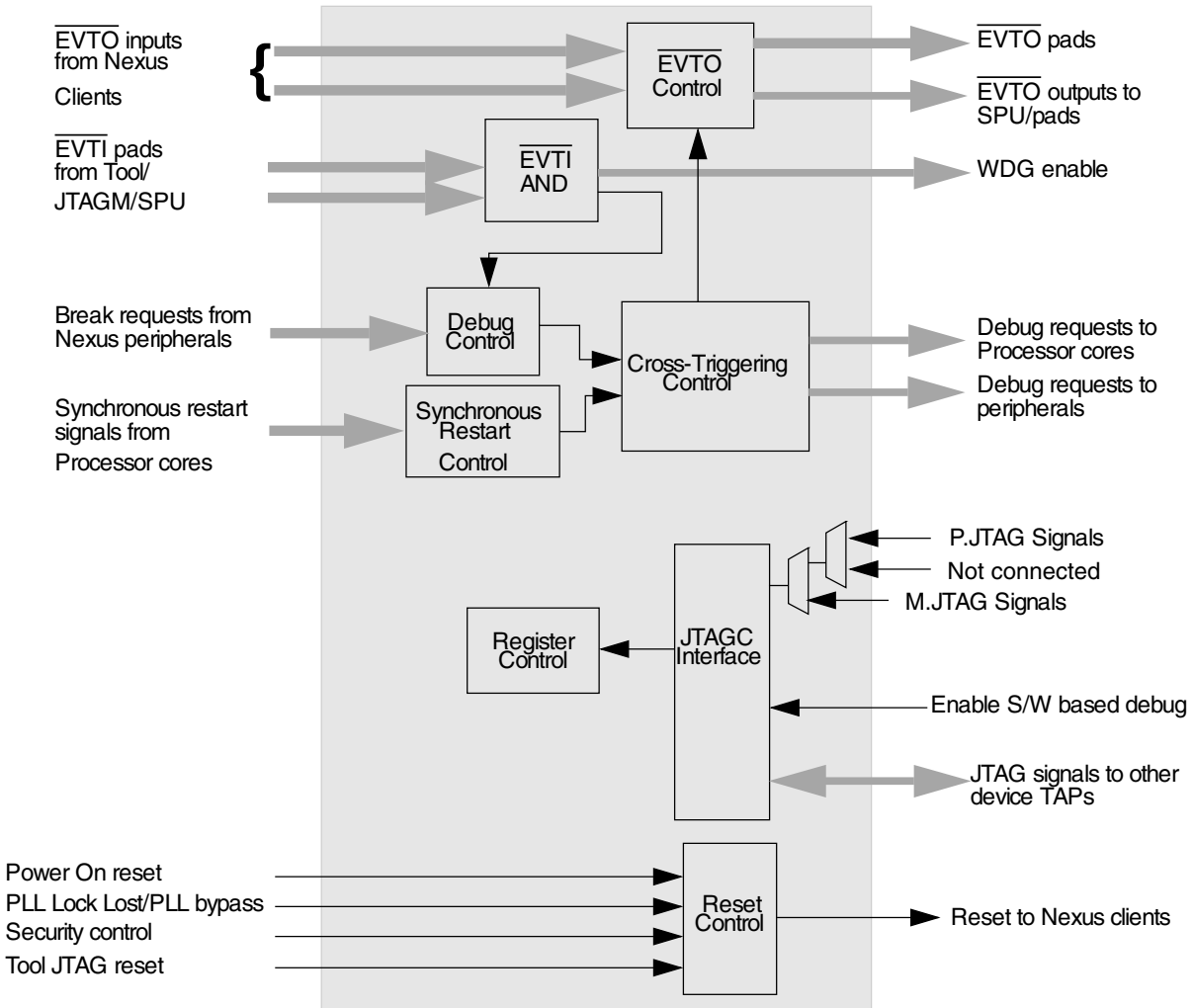


Figure 65-1. DCI block diagram

This figure shows the DCI configuration for the standard production package.

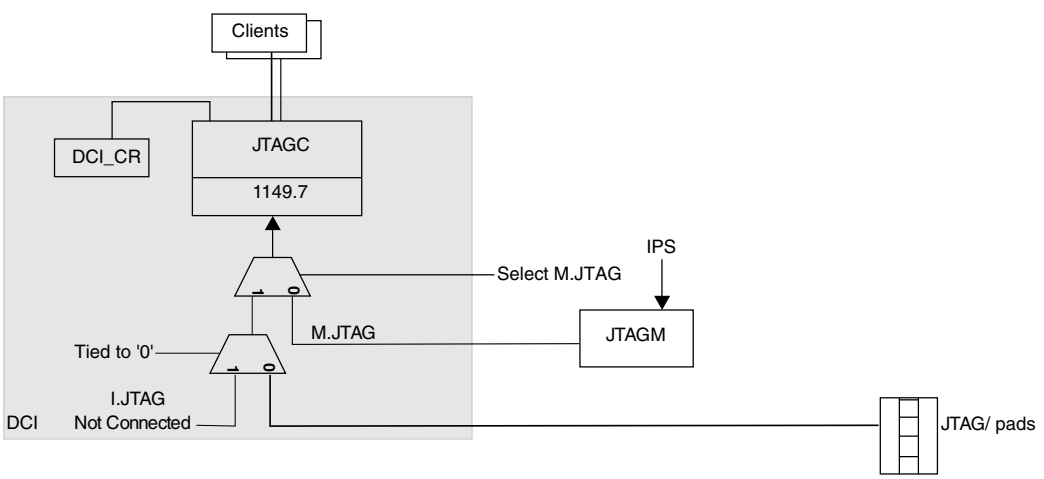


Figure 65-2. DCI in a standard configuration

65.1.3 Operating modes

Selection between various JTAG sources is done using a JTAGM register bit, JTAGM_MCR[DTM]. The JTAGM bit, JTAGM_MCR[DTM], is programmable via memory-mapped registers. JTAGM is selected as a JTAG source when JTAGM_MCR[DTM] is set.

This table shows a summary of the configuration options.

Table 65-1. DCI JTAG source selection

JTAGM_MCR[DTM]	DCI Source	JCOMP
0	P.JTAG (Standard JTAG)	Asserted
1	M.JTAG (Software)	Negated

The reset signal to the DCI block consists of the following:

- Power-on reset, Low/High voltage detection on IO/Core supplies
- JCOMP reset

The DCI puts the JTAGC TAP in reset when either power-on/low-voltage-detect reset is asserted or JCOMP is negated.

The M.JTAG provides an alternate interface to the debug logic.

When no debug tool is connected, device power-on reset puts the DCI in standard 5-pin JTAG mode. This configures the TDO pin in output mode with the reset value (1'b0). There is no change in port values if no debug tool is connected.

This table contains the operating mode descriptions.

Table 65-2. DCI operating modes

Mode	Description
DCI_JTAG	JTAG mode. This is the default state after power-on reset. In this mode the DCI is in JTAG mode (controlled with P.JTAG) and is controlled by an external JTAG tool.
DCI_SW_DBG	Software Debug State. This state is achieved when no tool is present and software sets the JTAGM_MCR[DTM] bit. The DCI is then controlled using M.JTAG signals.

65.1.3.1 Software enabled debug and calibration mode

Apart from the JTAG sources, such as a JTAG-based tool, it is possible to generate JTAG sequences through software programming.

To support a software debug session via a CAN or SPI link, the software can enable debug and calibration mode via an memory-mapped register. It is done first by setting a control bit in the JTAGM (JTAGM_MCR[DTM]). For more details, refer to the JTAGM documentation. When enabled, the DCI is configured to receive M.JTAG signals which are driven by JTAGM registers accessible through software. Software can control all DCI features like any other JTAG source.

After a system reset, the JTAGM_MCR[DTM] bit is reset and the JCOMP pad is pulled-low, thus keeping the DCI in the reset state. Therefore, software is required to first change to a non-secured mode. Then it can program the JTAGM_MCR[DTM] bit, to select M.JTAG as the source inside the DCI, and then set the JTAGM_MCR[jtagm_JCOMP] bit present in the JTAGM, which takes all debug modules out of reset.

When MCR[jtagm_JCOMP] is programmed to 0, the JTAGC block reset is asserted, thus putting the debug and calibration logic in reset state. This bit is used during software based debug to initialize the debug and calibration logic.

Software based debug is not possible when a tool is connected. The presence of the tool is determined using the rising edge of the JCOMP pad to set an internal flag which, when set, disables the selection of JTAGM as a JTAG source.

NOTE

It is recommended to utilize either hardware based debug (JTAG) or software based debug. If only one of the debug method is used, all three connection methods (connect and reset target, connect and break application, connect and keep application running) are supported.

65.1.3.2 Nexus reset control

The JCOMP input that is used as the primary reset signal for the DCI is also used by the DCI to generate a single-bit reset signal for other Nexus blocks. It is used to reset all debug functions without affecting any system level functions. The DCI provides the reset signal for all Nexus blocks. This reset is controlled using the following signals:

- Power-on reset/Low voltage detect

- JCOMP
- SoC Debug_Enable signal
- System PLL is out of lock state and not bypassed

Asserting power-on reset, negating JCOMP, or negating the Debug_Enable signal, results in asynchronous entry into reset state. Following negation of power-on reset, the Nexus reset signal remains asserted until the system clock achieves lock or in bypass mode, as indicated by the system clock lock status signal, and Debug_Enable is asserted and JCOMP is asserted. Any subsequent loss of PLL lock does not generate a Nexus reset. This reset signal is unaffected by other sources of reset.

The following table shows how the Nexus reset is asserted/negated based on various signals.

Table 65-3. Nexus Reset Truth Table

POR	JCOMP	Debug_Enable	PLL lock	Nexus reset
Assert	X	X	X	Assert
X	Negate	X	X	Assert
X	X	Disable	X	Assert
X	X	X	Unlocked	Assert
Negate	Assert	Enable	Locked	Negate

65.1.3.3 System watchdog control

The DCI is also used to enable or disable the Software Watchdog Timer (SWT). The DCI provides a single bit SWT enable signal which is used to control all SWTs in the MCU. For this, the $\overline{\text{EVTI}}$[0] pin is used. If the $\overline{\text{EVTI}}$[0] pin remains high when the tool asserts JCOMP, then the SWT is enabled. But if the tool drives the $\overline{\text{EVTI}}$[0] pin low before asserting JCOMP pin, then the SWT is disabled. By default the SWT is enabled and a tool has an option to enable/disable the SWT using the $\overline{\text{EVTI}}$[0] pin.

65.1.3.4 Break control

The DCI provides central break control among the CPUs and non-CPU debug modules such as timers, DMA, Watchdog and other peripherals.

There are various different input sources that provide debug requests:

- $\overline{\text{EVTI}}$[1:0] signals from EVTI pads

- $\overline{\text{EVTI}}$[1:0] signals from JTAGM
- $\overline{\text{EVTI}}$[1:0] signals from the Sequence Processing Unit (SPU)

The DCI generates following debug request signals:

- Debug Request signal to be used for all non-CPU peripherals
- External Debug Request to be used to request external debug event to processor cores
- OnCE Debug Request to be used to request OnCE debug event to processor cores
- All break control requests are issued simultaneously to CPUs and peripherals, then synchronized to each clock domain

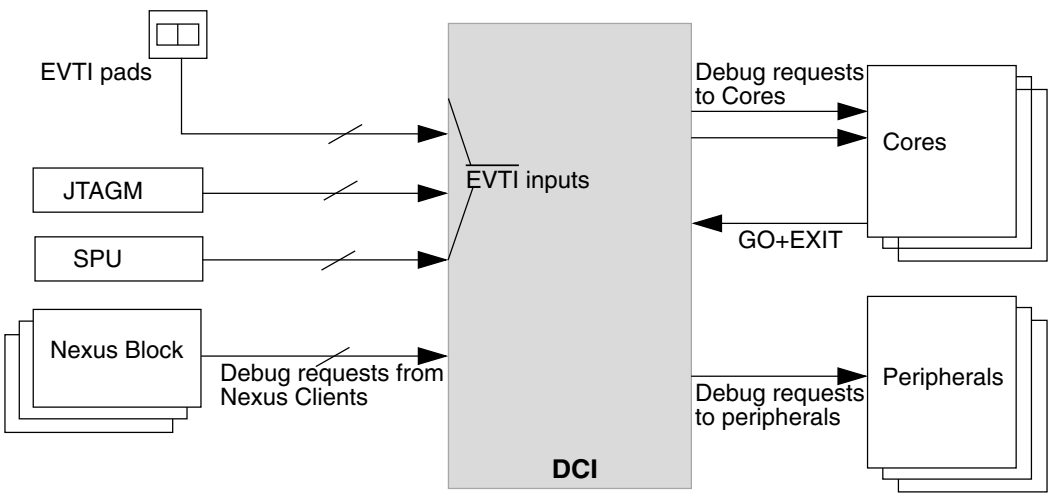


Figure 65-3. DCI debug control mechanism

The various sources of $\overline{\text{EVTI}}$[1:0] inputs are synchronized and logically ANDed in the DCI to generate two debug requests $\overline{\text{DCI_EVTI}}$[1:0] for the processor cores and non-CPU peripherals. Of these two inputs, $\overline{\text{EVTI}}$[0] is used to generate debug requests for non-CPU peripherals and for processor cores through the external debug request signal. The other input, $\overline{\text{EVTI}}$[1], is also used to generate debug requests for non-CPU peripherals and it generates debug requests to processor cores through the OnCE debug request signal.

The external debug request to processor core is asserted as a single cycle pulse and is synchronized on the respective core clock. The OnCE debug request is negated when the acknowledge signal from the Cores is asserted. The CPU(s) remain in the debug state until GO+EX command is executed for that CPU. Exiting a CPU out of debug state by executing GO+EX command does not take other CPUs out of debug state. The peripheral

debug request is negated when the processor core(s) execute GO+EXIT command. The $\overline{\text{EVTI}}$[1:0] signals coming from the pads are enabled for debug control using DCI_CR[5:4] bits.

In order to selectively enable/disable debug requests to system peripherals, the respective debug enable bit inside the peripheral should be used.

This figure shows the timing sequence.

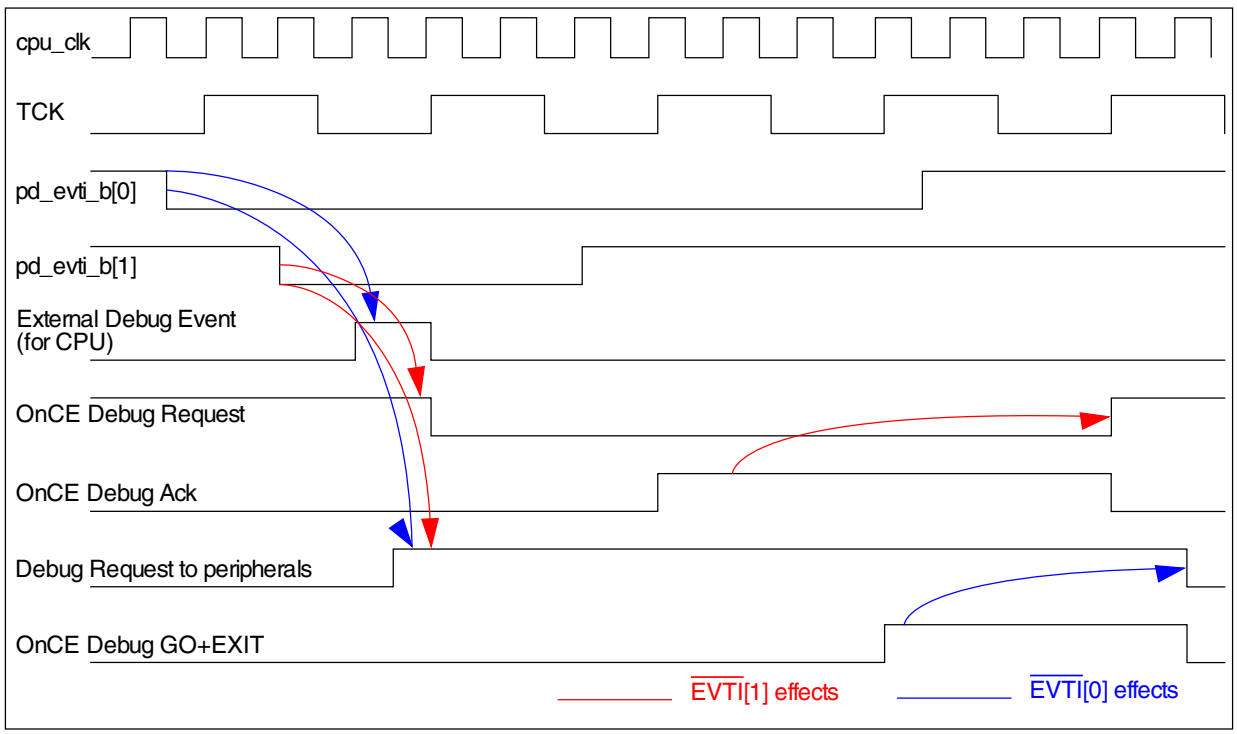


Figure 65-4. DCI debug signal generation

65.1.3.5 Cross-triggering control

Many of the blocks in the device, including the Nexus trace blocks and more sophisticated peripherals, have debug features that allow them to request device-wide breaks to enter debug mode. Many blocks can be programmed to halt operation upon debug mode entry or ignore it and continue normal operation. All debug mode entry requests are input to the DCI, ORed together and the result is used to produce the debug requests to the CPUs and non-CPU peripherals.

The cross-triggering feature is implemented through selectively enabling debug control bits in the respective modules. Processor cores (enabled using DBCR0[DEVT1] bit) and non-core peripherals can be programmed to halt operation through their respective control bits. Cross-triggering provides a selective feature to put modules in debug mode upon receiving halt request signals from Nexus modules.

65.1.3.6 Synchronous restart control

The DCI implements a synchronous restart control feature to synchronously restart all debug peripherals (CPU and non-CPU peripherals which are put in debug state). This is accomplished with the following steps:

1. The tool sequentially scans into CPUSCR for each processor core to restore the proper processor state.
2. The tool scans into the Production device tap controller the AUX_PARALLEL instruction, which then selects all processor core tap controllers in parallel.
3. To synchronously exit any combination of processor cores from debug mode, the tool simultaneously scans into all processor core tap instruction registers (OCMD) the 10-bit sequence 0x180 (i.e. set GO and EX, clear RS[0:6]).
4. If a processor core is out of debug mode, it sets a flag OnCE GO+EXIT.
5. When the OR of the processor core OnCE GO+EXIT signals is asserted, then all other selected non-processor cores and peripherals in debug would also be synchronously exited from debug mode. It is done by negating peripheral debug request signal.

65.1.3.7 EVTO management

The EVTO management block diagram is presented in the following figure.

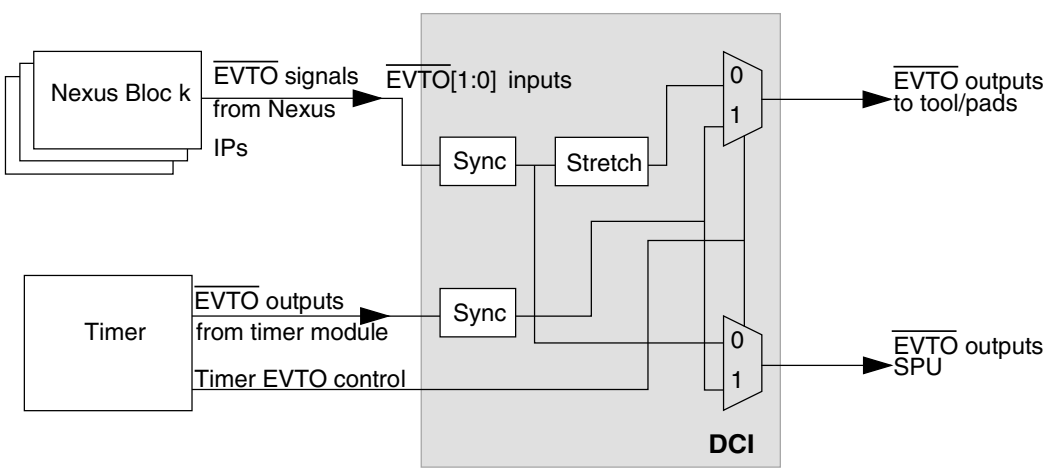


Figure 65-5. DCI EVTO management

The DCI module controls sharing of the EVTO outputs between all Nexus Clients including the cores that produce $\overline{\text{EVTO}}$[1:0] signals. The DCI uses a high frequency clock to sample EVTO signals coming from all modules which might generate EVTO outputs at their own clock frequency. After receiving a single clock period of asserted $\overline{\text{EVTO}}$[1:0] signals from any Nexus client, the DCI immediately asserts $\overline{\text{EVTO}}$[1:0] outputs and stretches it for a minimum of one output clock period for the output to the tool(Figure 65-6). The DCI also generates EVTO outputs in other device clock domains so that other Nexus clients can receive EVTO inputs without the need of any synchronization. EVTO function is active as long as the DCI is not in reset.

The DCI also allows the timer to take control of the EVTO outputs by controlling a timer register. If these fields are programmed to HALT/TIM/TOM/ATOM EVTO value, the timer EVTO outputs are directly available on the ports bypassing the stretch logic.

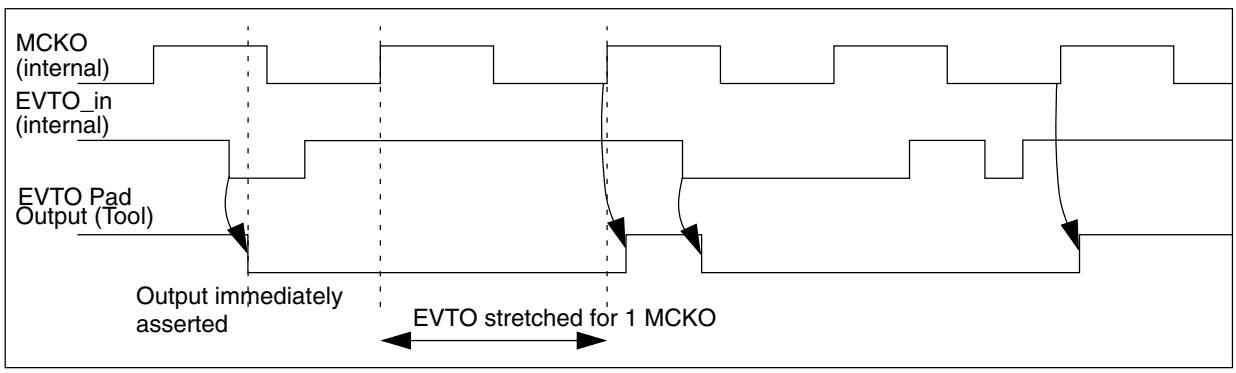


Figure 65-6. DCI EVTO timing diagram

65.1.3.8 Reset control

The DCI includes control bits to allow a connected tool to force a reset of the connected device. The description is as follows:

- Force functional reset—This reset is generated by programming the DCI_CR[FPD_FUNC] bit. When set, it forces functional reset. This bit has to be reset by the tool to negate the reset signal.
- Force POR or destructive reset—This reset is generated by programming DCI_CR[FPD_DEST] bit. When set it forces destructive reset. This bit has to be reset by the tool to negate the reset signal.

These reset signals are directly connected to the respective DCI_CR bits. The DCI does not reset the DCI_CR bits after generation of the respective reset signals. The tool must specifically reset the appropriate DCI_CR bits to remove the reset signals. These reset control bits can be set/reset through JTAG programming, and only reset through JCOMP assertion or Power-on-reset.

65.2 External signal description

The JTAGC module used inside DCI module defines the interface with external signals. It consists of five signals (TMS/TDI/TDO/TCK/JCOMP) that connect to off chip development tools and allow access to test support functions. For more details for JTAG signals, refer to JTAGC documentation.

65.3 Register description

The PD DCI module contains two registers, DCI_CR and DCI_PINCR, which are present inside the JTAGC module. These registers are only controlled using the JTAGC TAP and no memory-mapped registers are present.

65.3.1 DCI control register (DCI_CR)

The DCI_CR is a 32-bit data register. This register is programmed using JTAG shift with the corresponding enable instruction described in the JTAGC chapter. The following table shows the format of the DCI_CR.

	31	30	29	28	27	26	25	24
R W	Reserved							
Reset	1	1	1	1	1	1	1	1

	23	22	21	20	19	18	17	16
R W	Reserved							
Reset	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8
R W	Reserved					Reserved	FPD_FUNC	FPD_DEST
Reset	1	1	1	1	1	0	0	0

	7	6	5	4	3	2	1	0
R W	Reserved	Reserved	EVTI1_PAD_EN	EVTI0_PAD_EN	EVTO1_PAD_EN	EVTO0_PAD_EN	Reserved	Reserved
Reset	0	0	0	0	0	0	0	0

The DCI_CR is described in this table.

Table 65-4. DCI_CR field descriptions

Field	Description	
9 FPD_FUNC	Force functional reset	
	0	Do not force functional reset
	1	Force functional reset
8 FPD_DEST	Force destructive reset	
	0	Do not force destructive reset
	1	Force destructive reset
5 EVTI1_PAD_EN	$\overline{\text{EVTI}}$[1] feature enable	
	0	$\overline{\text{EVTI}}$[1] disabled (default)
	1	$\overline{\text{EVTI}}$[1] enabled
4 EVTI0_PAD_EN[1, 1]	$\overline{\text{EVTI}}$[0] feature enable	
	0	$\overline{\text{EVTI}}$[0] disabled (default)
	1	$\overline{\text{EVTI}}$[0] enabled
3 EVTO1_PAD_EN	Output enable for $\overline{\text{EVTO}}$[1] pad	
	0	$\overline{\text{EVTO}}$[1] pad output not enabled (default)
	1	$\overline{\text{EVTO}}$[1] pad output enabled

Table continues on the next page...

Table 65-4. DCI_CR field descriptions (continued)

Field	Description	
2 EVTO0_PAD_EN	Output enable for $\overline{\text{EVTO}}$[0] pad	
	0	$\overline{\text{EVTO}}$[0] pad output not enabled (default)
	1	$\overline{\text{EVTO}}$[0] pad output enabled

1. In an ED that has both PD and BD, EVTI1_PAD_EN and EVTI0_PAD_EN must be programmed in both BD and PD instances of the DCI in order to allow events generated on the BD to pass through the BD DCI and reach the PD DCI.

65.3.2 DCI EVTx pin multiplexing control register (DCI_PINCR)

The DCI_PINCR is a 32-bit data register. This register is programmed using JTAG shift with the corresponding enable instruction described in the JTAGC chapter. For a description of the DCI_PINCR functionality, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Chapter 66
JTAG Controller (JTAGC)

66.1 Introduction

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format.

66.1.1 Block diagram

The following is a simplified block diagram of the JTAG Controller (JTAGC) block. Refer to the chip-specific configuration information as well as Register description for more information about the JTAGC registers.

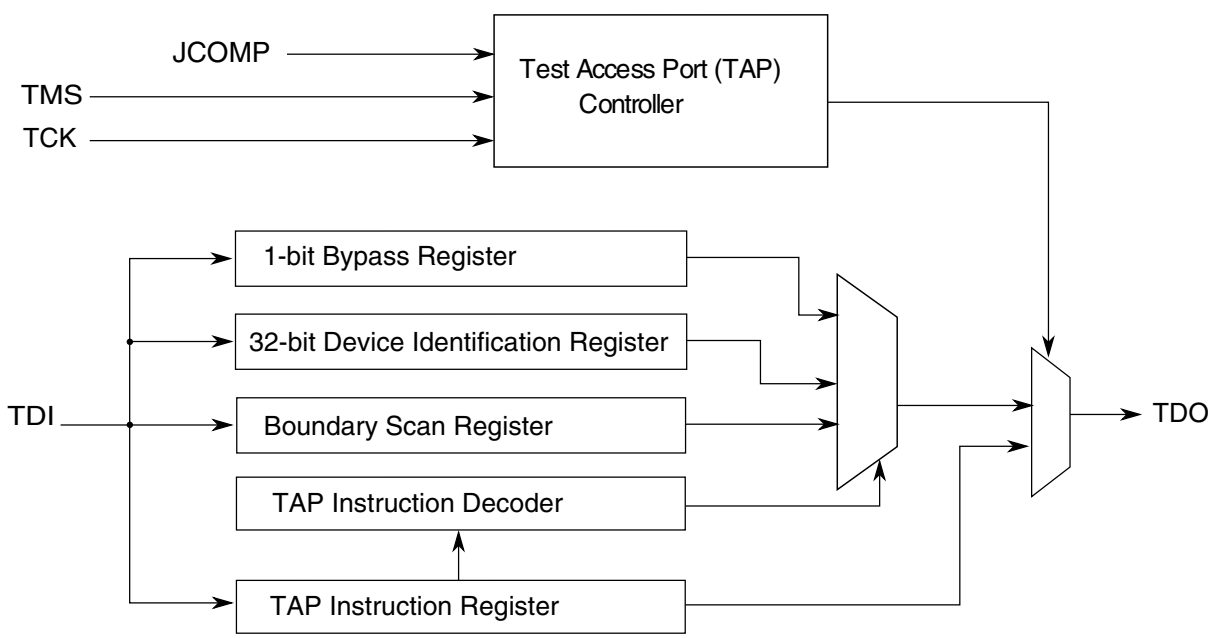


Figure 66-1. JTAG (IEEE 1149.1) block diagram

66.1.2 Features

The JTAGC block is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface
  - 4 pins (TDI, TMS, TCK, and TDO)
- JCOMP input that provides reset control
- Instruction register that supports several IEEE 1149.1-2001 defined instructions as well as several public and private device-specific instructions. Refer to Table 66-4 for a list of supported instructions.
- Sharing of the TAP with other TAP controllers via ACCESS_AUX_x instructions
- JTAG_PASSWORD register
- Bypass register, boundary scan register, data registers, and device identification register.
- TAP controller state machine that controls the operation of the data registers, instruction register and associated circuitry.

66.1.3 Modes of operation

The JTAGC block uses JCOMP and a power-on reset indication as its primary reset signals. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

66.1.3.1 Reset

The JTAGC block is placed in reset when either power-on reset is asserted, JCOMP is negated, or the TMS input is held high for enough consecutive rising edges of TCK to sequence the TAP controller state machine into the Test-Logic-Reset state. Holding TMS high for five consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Asserting power-on reset or setting JCOMP to a value other than the value required to enable the JTAGC block results in asynchronous entry into the reset state. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered
- The instruction register is loaded with the IDCODE instruction

66.1.3.2 IEEE 1149.1-2001 defined test modes

The JTAGC block supports several IEEE 1149.1-2001 defined test modes. A test mode is selected by loading the appropriate instruction into the instruction register while the JTAGC is enabled. Supported test instructions include EXTEST, HIGHZ, CLAMP, SAMPLE and SAMPLE/PRELOAD. Each instruction defines the set of data register(s) that may operate and interact with the on-chip system logic while the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is enabled for serial access between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. The single-bit bypass register shift stage is enabled for serial access between TDI and TDO when the BYPASS, HIGHZ, CLAMP or reserved instructions are active. The functionality of each test mode is explained in more detail in JTAGC block instructions.

66.1.3.3 Bypass mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAGC block into bypass mode. While in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

66.2 External signal description

The JTAGC consists of a set of signals that connect to off chip development tools and allow access to test support functions. The JTAGC signals are outlined in the following table and described in the following sections.

Table 66-1. JTAG signal properties

Name	I/O	Function	Reset State	Pull
TCK	Input	Test Clock	—	Down

Table continues on the next page...

Table 66-1. JTAG signal properties (continued)

Name	I/O	Function	Reset State	Pull
TDI	Input	Test Data In	—	Up
TDO	Output	Test Data Out	High Z[1, 1]	—
TMS	Input	Test Mode Select	—	Up
JCOMP	Input	JTAG Compliancy	—	Down

1. TDO output buffer enable is negated when the JTAGC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented at the TDO pad for use when JTAGC is inactive.

66.2.1 TCK—Test clock input

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the TAP.

66.2.2 TDI—Test data input

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

66.2.3 TDO—Test data output

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is three-stateable and is actively driven only in the Shift-IR and Shift-DR states of the TAP controller state machine, which is described in TAP controller state machine.

66.2.4 TMS—Test mode select

Test Mode Select (TMS) is an input pin used to sequence the IEEE 1149.1-2001 test control state machine. TMS is sampled on the rising edge of TCK.

66.2.5 JCOMP—JTAG compliancy

The JCOMP signal provides IEEE 1149.1-2001 compatibility and provides the ability to share the TAP. The JTAGC TAP controller is enabled when JCOMP is set to the JTAGC enable encoding, otherwise the JTAGC TAP controller remains in reset.

66.3 Register description

This section provides a detailed description of the JTAGC block registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

66.3.1 Instruction register

The JTAGC block uses a 6-bit instruction register as shown in the following figure. The instruction register allows instructions to be loaded into the block to select the test to be performed or the test data register to be accessed or both. Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the IDCODE instruction. During the Capture-IR TAP controller state, the instruction shift register is loaded with the value 000001b, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

	5	4	3	2	1	0
R	0	0	0	0	0	1
W	Instruction Code					
Reset:	0	0	0	0	0	1

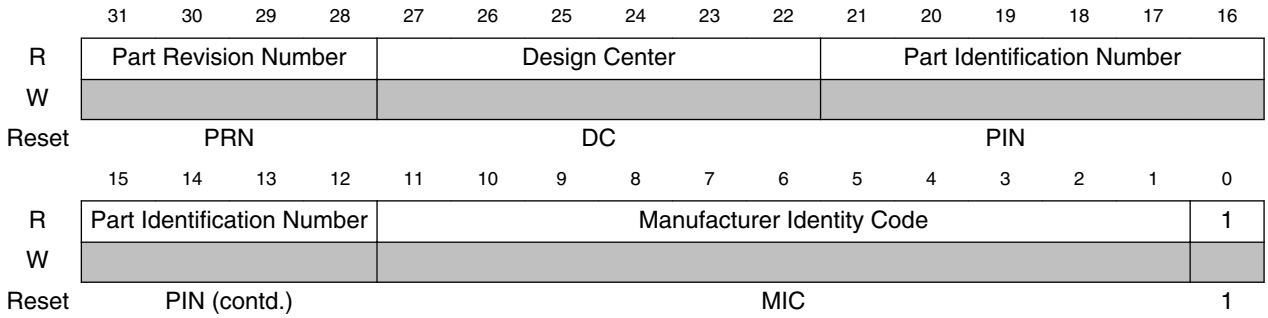
Figure 66-2. Instruction register

66.3.2 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS, CLAMP, HIGHZ or reserve instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

66.3.3 Device identification register

The device identification (JTAG ID) register, shown in the following figure, allows the revision number, part number, manufacturer, and design center responsible for the design of the part to be determined through the TAP. The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is active. Entry into the Capture-DR state while the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the Update-DR state.



The following table describes the device identification register functions.

Table 66-2. Device identification register field descriptions

Field	Description
PRN	Part Revision Number. Contains the revision number of the part. Value is 0x1.
DC	Design Center. Indicates the design center. Value is 0x20.
PIN	Part Identification Number. Contains the part number of the device. 0x346.
MIC	Manufacturer Identity Code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID. Value is 0x00E.
IDCODE ID	IDCODE Register ID. Identifies this register as the device identification register and not the bypass register. Always set to 1.

66.3.4 JTAG_PASSWORD register

The JTAG_PASSWORD register is a 256-bit shift register path from TDI to TDO selected when the ENABLE_JTAG_PASSWORD instruction is active. The default reset value of the JTAG_PASSWORD register is 256'b0. The JTAG_PASSWORD register transfers its value to a parallel hold register on the rising edge of TCK when the TAP controller state machine is in the Update-DR state. Once the ENABLE_JTAG_PASSWORD instruction is executed, the register value remains valid until a JTAG reset occurs. The operation of this register is described in the security documentation.

	255	...	2	1	0
R	JTAG_PASSWORD				
W					
Reset:	*[1]	*	*	*	*

1. The reset value of JTAG_PASSWORD is 256 'b0.

The following table describes the JTAG_PASSWORD register functions.

Table 66-3. JTAG_PASSWORD register field descriptions

Field	Description
JTAG_PASSWORD	JTAG Password. The JTAG_PASSWORD bits are used to provide the JTAG password for security.

66.3.5 Boundary scan register

The boundary scan register is connected between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. It is used to capture input pin data, force fixed values on output pins, and select a logic value and direction for bidirectional pins. Each bit of the boundary scan register represents a separate boundary scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in Boundary scan. The size of the boundary scan register and bit ordering is device-dependent and can be found in the device BSDL file.

66.4 Functional description

This section explains the JTAGC functional description.

66.4.1 JTAGC reset configuration

While in reset, the TAP controller is forced into the Test-Logic-Reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

66.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port

The JTAGC block uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. Ownership of the port is determined by the value of the currently loaded instruction. For more detail on TAP sharing via JTAGC instructions refer to ACCESS_AUX_x instructions.

Data is shifted between TDI and TDO though the selected register starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register, test data registers, and the bypass register.

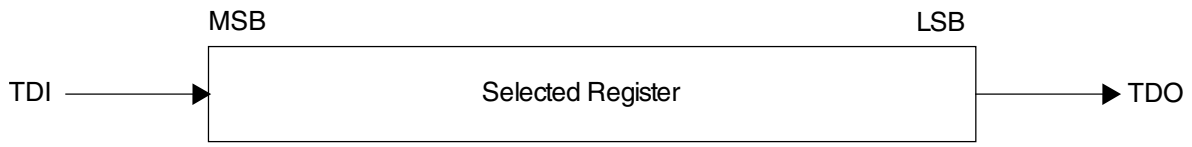
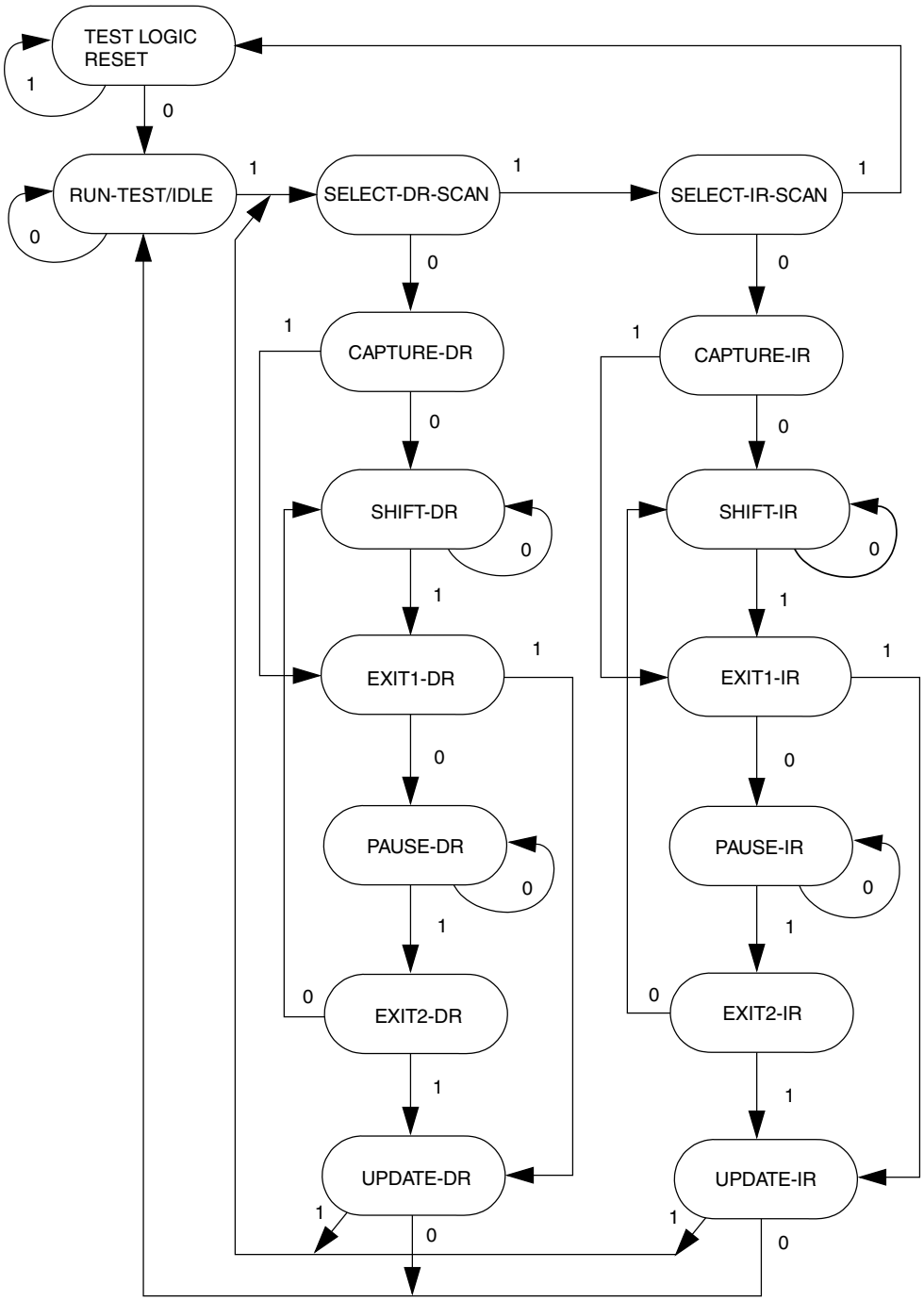


Figure 66-3. Shifting data through a register

66.4.3 TAP controller state machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin. The following figure shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCK signal. As the following figure shows, holding TMS at logic 1 while clocking TCK through a sufficient number of rising edges also causes the state machine to enter the Test-Logic-Reset state.



The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

Figure 66-4. IEEE 1149.1-2001 TAP controller finite state machine

66.4.3.1 Enabling the TAP controller

The JTAGC TAP controller is enabled by setting JCOMP to a logic 1 value.

66.4.3.2 Selecting an IEEE 1149.1-2001 register

Access to the JTAGC data registers is achieved by loading the instruction register with any of the JTAGC block instructions while the JTAGC is enabled. Instructions are shifted in via the Select-IR-Scan path and loaded in the Update-IR state. At this point, all data register access is performed via the Select-DR-Scan path.

The Select-DR-Scan path is used to read or write the register data by shifting in the data (LSB first) during the Shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the Capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the Update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

66.4.4 JTAGC block instructions

The JTAGC block implements the IEEE 1149.1-2001 defined instructions listed in the following table. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 66-4. General 6-bit JTAG instructions

Instruction	Code[5:0]	Instruction Summary
IDCODE	000001	Selects device identification register for shift
SAMPLE/PRELOAD	000010	Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation
SAMPLE	000011	Selects boundary scan register for shifting and sampling without disturbing functional operation
EXTEST	000100	Selects boundary scan register and applies preloaded values to output pins. **NOTE:** Execution of this instruction asserts functional reset.
Factory debug reserved	000101	Intended for factory debug only
Factory debug reserved	000110	Intended for factory debug only
ENABLE_JTAG_PASSWORD	000111	Selects JTAG_PASSWORD register
HIGHZ	001001	Selects bypass register and three-states all output pins. **NOTE:** Execution of this instruction asserts functional reset.
Factory debug reserved	001010	Intended for factory debug only
CLAMP	001100	Selects bypass register and applies preloaded values to output pins. **NOTE:** Execution of this instruction asserts functional reset.

Table continues on the next page...

Table 66-4. General 6-bit JTAG instructions (continued)

Instruction	Code[5:0]	Instruction Summary
ENABLE_DCI_CR	001110	Enables access to DCI CR register
ENABLE_DCI_PINCR	001111	Enables access to DCI PINCR register
BYPASS	111111	Selects bypass register for data operations
Reserved[1]	All other opcodes	Decoded to select bypass register
ACCESS_AUX_x	100000-111110	Grants one of the auxiliary TAP controllers ownership of the TAP

1. The manufacturer reserves the right to change the decoding of reserved instruction codes in the future

The ACCESS_AUX instructions are described in the chip configuration debug information.

66.4.4.1 IDCODE instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAGC block is reset.

66.4.4.2 SAMPLE/PRELOAD instruction

The SAMPLE/PRELOAD instruction has two functions:

- The SAMPLE portion of the instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE/PRELOAD instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. Both the data capture and the shift operation are transparent to system operation.
- The PRELOAD portion of the instruction initializes the boundary scan register cells before selecting the EXTEST or CLAMP instructions to perform boundary scan tests. This is achieved by shifting in initialization data to the boundary scan register during the Shift-DR state. The initialization data is transferred to the parallel outputs of the boundary scan register cells on the falling edge of TCK in the Update-DR state. The data is applied to the external output pins by the EXTEST or CLAMP instruction. System operation is not affected.

66.4.4.3 SAMPLE instruction

The SAMPLE instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. There is no defined action in the Update-DR state. Both the data capture and the shift operation are transparent to system operation.

66.4.4.4 EXTEST External test instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins. Typically, the preloaded data is loaded into the boundary scan register using the SAMPLE/PRELOAD instruction before the selection of EXTEST. EXTEST asserts the internal system reset for the MCU to force a predictable internal state while performing external boundary scan operations.

66.4.4.5 ENABLE_JTAG_PASSWORD instruction

The ENABLE_JTAG_PASSWORD instruction selects the JTAG_PASSWORD register for connection as the shift path between TDI and TDO.

66.4.4.6 HIGHZ instruction

HIGHZ selects the bypass register as the shift path between TDI and TDO. While HIGHZ is active all output drivers are placed in an inactive drive state (e.g., high impedance). HIGHZ also asserts the internal system reset for the MCU to force a predictable internal state.

66.4.4.7 CLAMP instruction

CLAMP allows the state of signals driven from MCU pins to be determined from the boundary scan register while the bypass register is selected as the serial path between TDI and TDO. CLAMP enhances test efficiency by reducing the overall shift path to a

single bit (the bypass register) while conducting an EXTEST type of instruction through the boundary scan register. CLAMP also asserts the internal system reset for the MCU to force a predictable internal state.

66.4.4.8 ACCESS_AUX_x instructions

The JTAGC is configurable to allow other TAP controllers on the device to share the port with it. This is done by providing ACCESS_AUX_x instructions for each of these TAP controllers. When this instruction is loaded, control of the JTAG pins are transferred to the selected TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The JTAGC regains control of the JTAG port during the UPDATE-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive. Instructions not used to access an auxiliary TAP controller on a device are treated like the BYPASS instruction.

66.4.4.9 BYPASS instruction

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions. While the BYPASS instruction is active the system logic operates normally.

66.4.5 Boundary scan

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST, SAMPLE, or SAMPLE/PRELOAD instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

66.5 Initialization/Application information

The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

To initialize the JTAGC block and enable access to registers, the following sequence is required:

1. Set the JCOMP signal to the JTAGC enable value, thereby enabling the JTAGC TAP controller
2. Load the appropriate instruction for the test or action to be performed

Chapter 67
IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)

67.1 References

- IEEE Std 1149.1-2001 (R2008), Standard Test Access Port and Boundary-Scan Architecture
- IEEE Std 1149.7, Standard for Reduced-Pin and Enhanced-Functionality Test Access Port and Boundary-Scan Architecture

67.2 Abbreviations

Table 67-1. Abbreviations

Abbreviation	Expansion
APU	Advanced Protocol Unit
APFC	Auxiliary Pin Function Control
CGM	Conditional Group Member
CGMC	Conditional Group Membership Count
CID	Controller ID
CJTAG	Compact JTAG
CLTAPC	Chip-Level TAPC
CP	Check Packet
CPA	Check Process Active
DCU	Data Channel Unit
EOT	End of Transfer
EPU	Extended Protocol Unit
Jscan	JTAG Scan
Mscan	Maximum-flexibility Scan
nTRST	Active low Test Reset
Oscan	Optimized Scan
PSS	Pause Selection State

Table continues on the next page...

Table 67-1. Abbreviations (continued)

Abbreviation	Expansion
RSU	Reset and Selection Unit
SP	Scan Packet
SPA	Scan Packet Active
Sscan	Segmented Scan
SSD	Scan Selection Directive
SSM	Scan State Machine
STL	System Test Logic
TAP	Test Access Port
TAP.1	An IEEE 1149.1 Test Access Port
TAP.7	An IEEE 1149.7 Test Access Port
TCA	Tap Controller Address
TCK	Test Clock
TCKC	Test Clock Compact
TDI	Test Data Input
TDIC	Test Data Input Compact
TDO	Test Data Output
TDOC	Test Data Output Compact
TMS	Test Mode Select
TMSC	Test Mode Select Compact
TS	Target System
ZBS	Zero Bit Scan

67.3 Introduction

The CJTAG module implements parts of the IEEE 1149.7 standard for test and debug capabilities. IEEE 1149.7 defines enhanced test and debug capabilities which are backwards compatible with IEEE P1149.1. This standard is also known as "Compact JTAG" or "CJTAG".

The module implements the following parts of the IEEE 1149.7 standard:

- IEEE 1149.1 compliant behavior while allowing multiple on-chip TAP controllers
- Ability to assert test reset
- Improved system scan performance with one-bit IR and DR chip bypass paths

- Physical connection of a Debug and Test System (external tool) to a running system without corrupting its operation
- Operation with both 4-pin and 2-pin scan topologies

67.3.1 Types of operation

The 1149.7 standard provides a scalable and flexible solution to meet the needs of varied component and system complexities. There are three types of operation:

- **Compliant**
  - Compliant with the IEEE 1149.1 standard
  - Four-pin operation using the "Standard Protocol" — IEEE 1149.1 signaling where TCK, TMS, TDI and TDO signals transfer control/data information
  - More than one TAPC may be associated with the TAP (once made visible by private instructions)
- **Extended**
  - Compatible with the IEEE 1149.1 standard
  - Four-pin operation using the "Standard Protocol" — IEEE 1149.1 signaling where TCK, TMS, TDI(C) and TDO(C) signals transfer control/data information
  - Additional test/debug functions
- **Advanced**
  - Two-pin operation using the "Advanced Protocol" — signaling where TCK(C) and TMS(C) signals transfer control/data information
  - The TMS(C) signal is operated in a bidirectional manner with the TMS, TDI and TDO TAP information serialized and transferred via this signal

67.3.2 Deployment by class

The TAP.7 capabilities are deployed using five capability classes (T0–T4), each matching the needs of certain scan topologies and mixes of 1149.1 TAPs and 1149.7 TAPs. The classes provide the following capability:

- **Compliant Operation** – provides behavior compliant with the IEEE 1149.1-2001 standard:
  - **Class T0** – 1149.1 compliance when multiple TAPs are used on the same chip.
- **Extended Operation** – extends the capabilities of the IEEE 1149.1-2001 standard:
  - **Class T1** – T0 + Test access port power management, functional reset, and test reset.
  - **Class T2** – T1 + Chip bypass (1-bit) for IR/DR Scans + hot-connection with no errors..
  - **Class T3** – T2 + Star operation (TAPs are paralleled) with series equivalent scans for test.
- **Advanced Operation** – adds capabilities addressing test and debug needs in complex environments:
  - **Class T4** – T3 + Scan with two pins with the Advanced Protocol. A number of protocol optimizations maximize scan performance. TDI(C)/TDO(C) pins are optional and may have programmable function when implemented.

67.3.3 1149.7 TAP signals

The TAP signals for the T0–T4 TAP.7 classes are listed in the following table. Signal names that end in "C" are associated with expanded functionality (above that provided by the IEEE 1149.1 standard).

The notation used in the table is as follows:

- M = Mandatory signal
- MC = Mandatory signal with expanded functionality
- O = Optional signal
- OC = Optional signal with expanded functionality

The CJTAG module provides T4 capability and implements the TCKC, TMSC, TDIC, TDOC and nTRST pins.

Table 67-2. TAP.7 signal list

Signal Name	Description	Class				
		T4	T3	T2	T1	T0
TCK/TCKC	Test clock	MC	M	M	M	M
TMS/TMSC	Test mode select	MC	M	M	M	M

Table continues on the next page...

Table 67-2. TAP.7 signal list (continued)

Signal Name	Description	Class				
		T4	T3	T2	T1	T0
TDI/TDIC	Test data input	OC	MC	M	M	M
TDO/TDOC	Test data output	OC	MC	M	M	M
nTRST	Test reset	O	O	O	O	O
Mandatory signal count (minimum)		2	4	4	4	4

67.3.4 TAP.7 architecture

The TAP.7 hardware architecture is shown in the following figure. It is described with the hardware layers listed below:

- **STL – System Test Logic** – Logic with 1149.1 compliant behavior and underlying TAP hierarchy (T0 capabilities). Provides an IEEE 1149.1 interface for the T0 TAP. 7. This logic is outside the CJTAG module.
- **RSU – Reset and Selection Unit** – A hardware layer that is placed between the APU, EPU, or STL and the TAP.7 signals (Added as an option to support the use of the Control Protocol) Provides reset and TAP.7 Controller selection services.
- **EPU – Extended Protocol Unit** – A hardware layer that is placed between the STL and the TAP.7 signals. (Added for T1, T2, and T3 capabilities). Provides an IEEE 1149.1 interface for the T1–T3 TAP.7s.
- **APU – Advanced Protocol Unit** – A hardware layer that is placed between the STL/ EPU and the TAP.7 signals (added for T4 capabilities). Provides a T4 TAP.7 interface that is either narrow or wide, with the wide version providing an IEEE 1149.1 interface.

The CJTAG module supports T4 functionality that includes STL, RSU, EPU, and APU hardware layers.

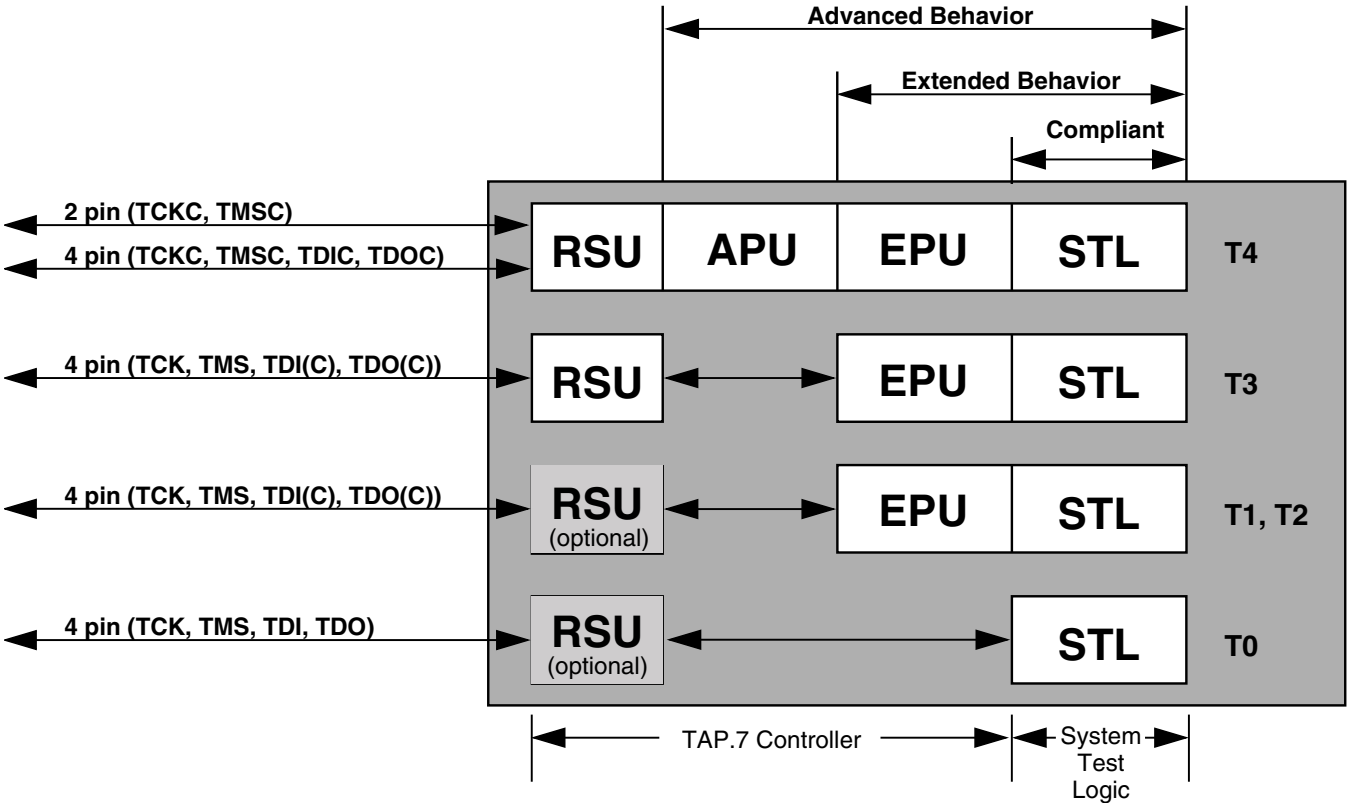


Figure 67-1. TAP.7 Controller Architecture

67.3.5 Protocols

The TAP.7 architecture utilizes the three protocols shown in the following figure.

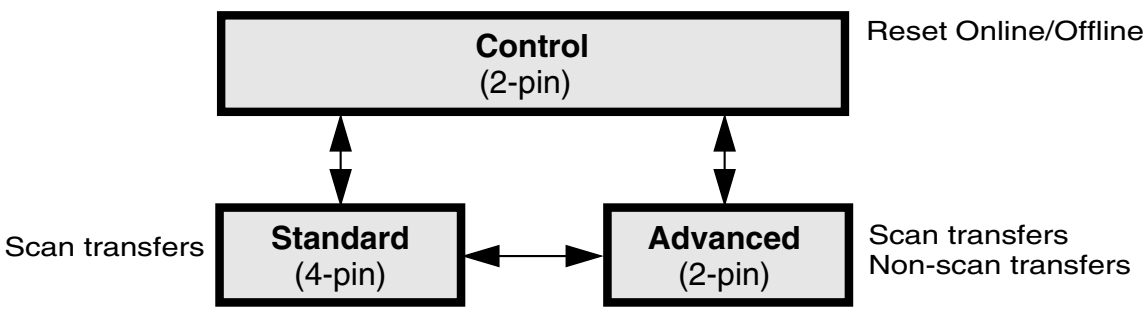


Figure 67-2. TAP.7 Behaviors

The Standard Protocol transfers data with the TMS(C) signal value being sampled with a TCK(C) edge. The Advanced and Control Protocols use both data transferred with the TMS(C) signal value being sampled with a TCK(C) signal edge and escape sequences. The Standard Protocol may be used to switch to the use of the Advanced Protocol and

vice versa. The Control Protocol can also be used to switch between the use of the Standard and the Advanced Protocol. The mandatory and optional deployment of these protocols with the TAP.7 classes is shown in the following table.

Table 67-3. Class/protocol relationship

Class	Protocol		
	Standard	Control	Advanced
T0 – T2	Mandatory	Optional	N/A
T3	Mandatory	Mandatory	N/A
T4	Mandatory	Mandatory	Mandatory

The CJTAG module implements T4 functionality and supports the Standard, Advanced, and Control Protocols.

67.3.5.1 Standard Protocol

The Standard Protocol is the signaling defined by the IEEE 1149.1 standard. This protocol is also used to implement TAP.7 commands using only the TCK(C) and TMS(C) signals in a manner that does not require either TAP.7 controller instruction or data registers. The TAP.7 controller does not add bits to the scan paths of underlying technology. The Standard Protocol can be used to manage the TAP.7 controller functionality independent of the function associated with the availability of the TDI and TDO signals. The Standard Protocol supports STL data transfers only when these signals are present and provide the 1149.1 functionality defined by the IEEE 1149.1 standard.

67.3.5.2 Advanced Protocol

The Advanced Protocol supports both STL data transfers and management of the TAP.7 controller functionality with both the 2-signal (narrow) and 4-signal (wide) configurations. The Advanced Protocol serializes the TMS, TDI, and TDO information used with the Standard Protocol using only the TCK(C) and TMS(C) signals. A number of additional scan formats provide bit sequences optimized for specific use cases. Each of these additional scan formats affects the TMSC signaling sequences.

67.3.5.3 Control Protocol

The Control Protocol uses Escape sequences to reset the TAP.7 controller, place a TAP.7 controller offline or online, and indicate End of Transmission (EOT) with the Advanced Protocol. The characteristics of this protocol provide Hot-Connect-Protection by requiring a preamble of alternating ones and zeroes preceding a selection escape when a TAP controller is started up offline.

67.4 Operating models

The following figure illustrates a simplified view of the CJTAG operating model.

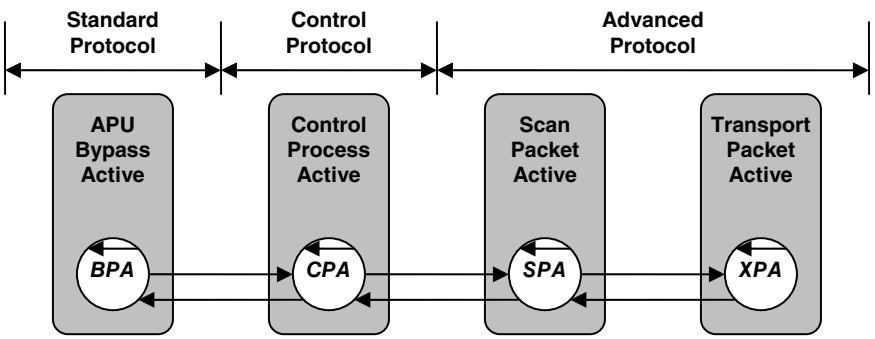


Figure 67-3. CJTAG operating model

67.5 CJTAG implementation summary

The CJTAG module is a T4 TAP.7 controller that implements the following functions.

67.5.1 T0 functions

- IDCODE. - implemented in chip level TAPC (JTAGC module)
- 1149.1 Compliance at start-up
- Isolating / Excluding of embedded TAPs – implemented in chip level TAPC (JTAGC module)
- Operation with other technologies and scan topologies
- Multiple embedded TAPs

67.5.2 T1 functions

- EPU Command processing
- EPU Class-Specific Registers
- Asserting a test reset to the STL (TRESET Register)

67.5.3 T2 functions

- "Super Bypass" function – JScan0, JScan1, JScan2 scan formats. Allows the STL to be coupled or decoupled.

67.5.4 T3 functions

- Generation of a TAP.7 controller reset with a Reset Escape Sequence
- JScan3 Scan Format

67.5.5 T4 functions

- TMSC sampling on rising edge or falling edge of TCKC
- MScan (For test and debug)
- OScan 0-7
- TDIC and TDOC / Alternate pin function multiplexing (wide 4 only)

67.6 Ancillary services

67.6.1 Overview

Ancillary services are pertinent to any TAP.7 controller implemented with an RSU, such as the CJTAG module.

Ancillary services include:

- Resets
- Start-up options

- Escape sequences
- TAPC state machine

67.6.2 Resets

The TAP.7 controller reset function expands the reset function provided by the IEEE 1149.1 standard to provide additional reset types. It supports six reset types, Type-0 – Type-5, listed below. It can be implemented with as few as two reset types (Type-2 and Type-4), or as many as six reset types (all types) of reset inputs: This is determined by the class and options implemented.

The CJTAG module utilizes Type-0, Type-2, Type-3, Type-4, and Type-5 resets.

- **Type-0** – Reset of the TAP.7 by power management logic (power-on reset)
- **Type-1** – Chip level generated Test Reset. (equivalent to Type-2)
- **Type-2** – Externally generated Test Reset (nTRST/nTRST_PD pin)
- **Type-3** – TAP.7 controller generated reset
- **Type-4** – *Test-Logic-Reset* state
- **Type-5** – TAP.7 Controller TReset Register reset of the CLTAPC.

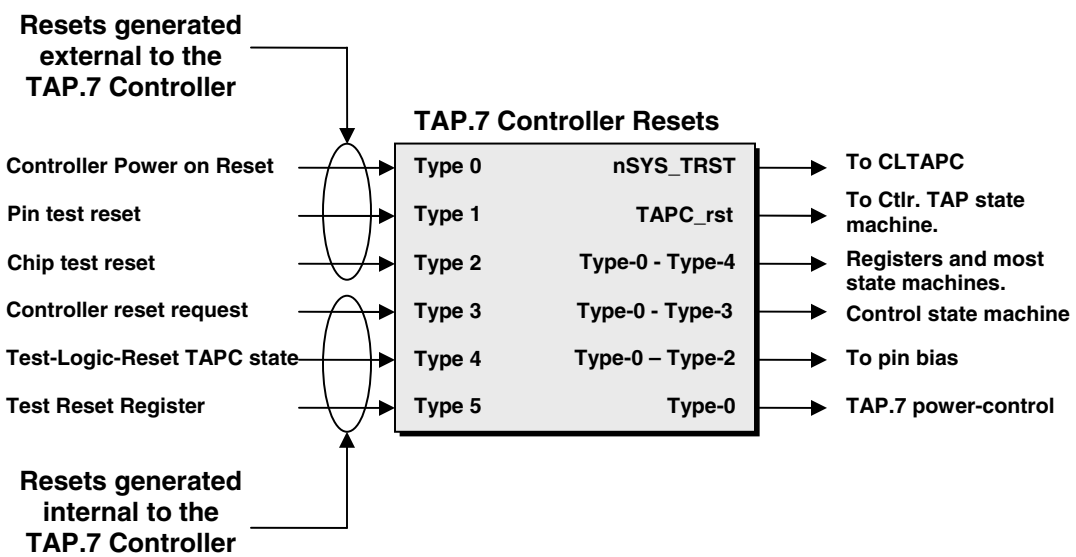


Figure 67-4. Conceptual view of the EPU reset logic and state machines

67.6.2.1 Type 5 reset

A Type-5 reset affects only the STL via the nSYS_TRST pin. It is asserted when the TRESET Register value is a logic 1. This register is set to its inactive state by reset Types 0-4.

67.6.2.2 Type 4 reset

A Type-4 reset is created by the *Test-Logic-Reset* state. It does not affect the TAP.7 TAPC state machine state as this state is the source of this reset, nor does it cause the assertion of the nSYS_TRST signal. It does however affect the coupling and decoupling of the STL. The Type-4 reset initializes the TAP.7 registers and most controller state machines. It affects but does not initialize the control state machine managing TAP.7 controller selection. It also establishes the default operating conditions.

67.6.2.3 Type 3 reset

A Type-3 reset is created by reset requests generated by the TAP.7 controller. A Type-3 reset performs the functions associated with a Type-4 reset and creates the *Test-Logic-Reset* state.

A Type-3 reset is generated in response to reset requests generated by EPU and APU logic

- A reset escape sequence
- Reset protocol sequences (delay packet directive, check packet directive)

A Type-3 reset request is generated with a reset escape sequence. This function is similar to but not the same as the reset function generated by a test reset pin. The Advanced Protocol includes certain bit patterns that initiate Type-3 resets. These bit patterns assist in the initialization of the TAP.7 controller when the TCKC signal is sourced by the TS and the external tool/TS connection is broken. The external tool may also generate these sequences, if desired, while the external tool/TS connection remains intact. These reset requests cause a Type-3 reset that is one clock wide.

With this being the case, consideration may be given to integrating the CJTAG module without the use of the Test Reset signal (nTRST/nTRST_PD pin).

67.6.2.4 Type 2 reset

A Type-2 reset is created by chip level logic to initialize the TAP.7 controller when TAP.7 power control is not implemented. It is unimplemented otherwise. It performs all of the functions of the Type-3 reset in addition to initializing the escape sequence detection logic.

67.6.2.5 Type 1 reset

A Type-1 reset is created with either the nTRST or nTRST_PD pin. It is not created when neither of these pins is implemented. It performs all of the functions of the Type-2 reset. It initiates TAP.7 controller power down provided the TAP.7 power control mode permits power down.

67.6.2.6 Type 0 reset

A Type-0 reset is implemented when TAP.7 controller power down is implemented. It performs all of the functions of a Type-1 reset and also initializes the TAP.7 power control logic.

67.6.2.7 Reset State Machine

The Reset State Machine assists in the translation of the reset inputs to the reset outputs. It handles Type-3 reset controller reset requests. The state machine state assures Type-0, Type-1, and Type-2 resets are forwarded to the STL without modifying their duration.

This state machine allows a Type-3 reset only when its state is “T3 Reset” (two TCK(C) falling edges have occurred in the absence of a Type-0, Type-1, and Type-2 reset).

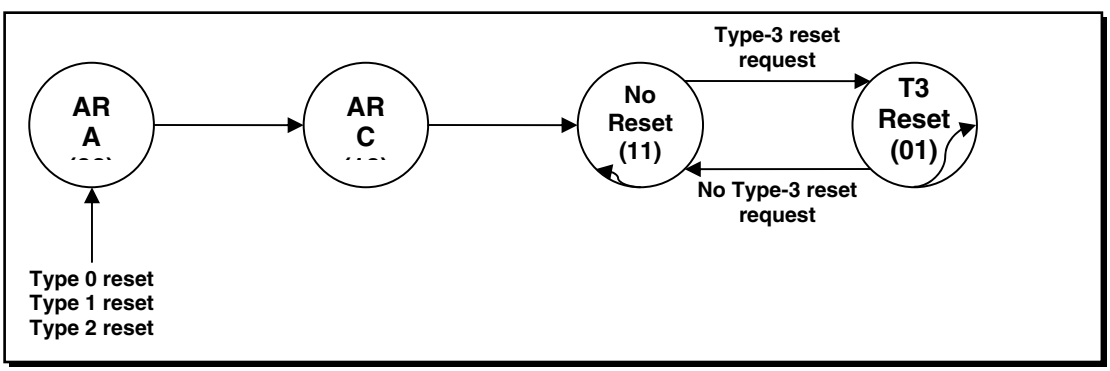


Figure 67-5. Reset state machine

Table 67-4. Reset state descriptions

Reset state		Description
ARA	Async. Reset Asserted	An async. reset has occurred or is occurring
ARC	Async. Reset Completed	An async. reset has occurred and has been released
No Reset	No Reset	No Reset
T3 Reset	Type-3 Reset	A Type-3 reset request has been processed.

Type-0, Type-1, and Type-2 resets are fully asynchronous. These resets are passed directly to nSYS_TRST without modification. The Type-3, Type-4, and Type-5 resets are fully synchronous.

The effects of the TAP.7 controller reset types are shown in the following table.

Table 67-5. Reset effects

Reset	CLTAPC	Pin-Hi-Z	State Machines other than CSM	Control State Machine (CSM)	TAPC SM State	Escape Sequence Detection	Power Control State
Type-5	Yes	–	–	–	–	–	–
Type-4	–	Yes	Yes	–	–	–	–
Type-3	Yes	Yes	Yes	Yes	Yes	–	–
Type-2	Yes	Yes	Yes	Yes	Yes	Yes	–
Type-1	Yes	Yes	Yes	Yes	Yes	Yes	–
Type-0	Yes	Yes	Yes	Yes	Yes	Yes	Yes

67.6.3 Start-up Options

67.6.3.1 Overview

The TAP.7 start-up options are shown in the following table. The only startup option currently supported by the CJTAG module is 1149.1 Compliant.

Table 67-6. Start-up Options

Start-up Option	Description
1149.1 Compliant	TAP.1 signal functionality, the CLTAPC is coupled
1149.1 Compatible	TAP.7 signal functionality, the CLTAPC is decoupled

Table continues on the next page...

Table 67-6. Start-up Options (continued)

Start-up Option	Description
1149.1 Protocol Compatible	No, or alternate TDIC/TDOC signal functionality, the CLTAPC is decoupled.
Offline-at-Start-up (Dormant)	The TAP.7 controller is offline and awaits synchronization to external tool operation so as to be placed online, the CLTAPC is decoupled after its state is moved to *Run-Test/Idle*

67.6.3.2 1149.1 Compliant start-up

Start-up behavior with the 1149.1 Compliant behavior start-up option:

- 1149.1-specific behavior / Standard protocol
- The TAPC finite state machine operated in lock step with the CLTAPC finite state machine.
- The TDIC and TDOC pins are present and provide the TDI and TDO function.
- The CLTAPC is coupled at start-up.

With this start-up option, the TAP.7 controller may be programmed and the scan topology may be interrogated. If the scan topology is known to be the Series Scan Topology at power-up, the TAP.7 can be operated as though it is a TAP.1.

Table 67-7. Reset values for TAP.7 startup options

Start-up Option	Type-0 – Type-3 Reset?	Default TAP.7 and TAP.7 controller characteristics				
		TAP.7 Controller online	CLTAPC Coupled	SGC Reg. == 1	SCNFMT	Initial TMSC pin bias
1149.1 Compliant	x	Yes	Yes	Yes	JScan0	PU

67.6.4 RSU operation

The RSU uses TCKC and TMSC with the control protocol for configuration control.

67.6.4.1 General Operation

From a very high level, the RSU enables the remainder of the TAP.7 functionality when the TAP.7 controller is Online and disables the remainder of the TAP.7 functionality otherwise. The RSU either consumes or discards the TMS(C) information while the TAP. 7 controller is Offline. The relationship of the escape sequence detection, technology selection logic, and the underlying technology is shown in the following figure.

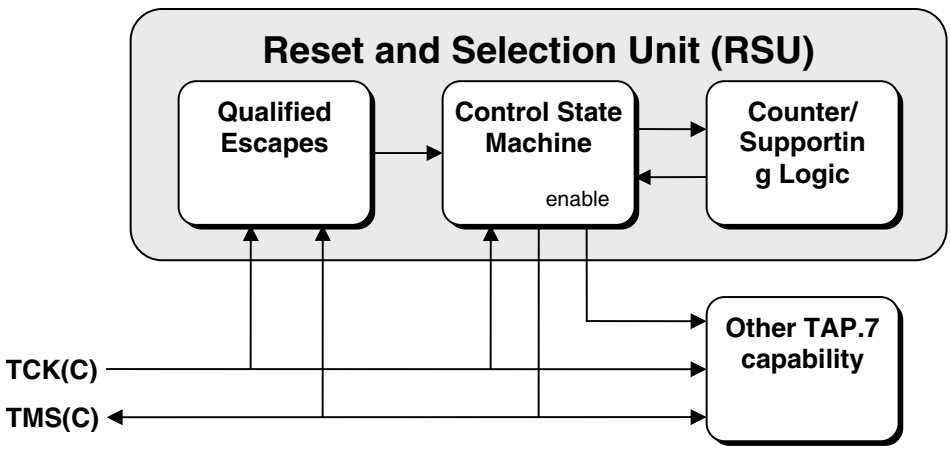


Figure 67-6. Conceptual view of technology selection mechanism

67.6.4.2 Common Signaling Across Technologies

With the TAP.7 architecture, the signaling conventions chosen for technology selection is multiple data transitions of the TMS(C) signal while the TCK(C) signal is a logic 1. The common signaling space is available when the external tool sources the TCK(C) signal as it is capable of creating these signaling conventions. It is unavailable when the TS sources the TCK(C) signal as the external tool cannot stop the TCKC signal at a logic 1 to create these signaling conventions.

67.6.4.3 Escape Sequence Detection

TAP.7 TAPC interprets the count of the number of TMS(C) edges while TCK(C) is a logic 1 as one of four escape sequences. Each escape sequence has a different function and TMSC edge count. These escape sequences, their edge counts, and their functions are described below:

- **Custom** (2 or 3 edges) – Ends scan and transport transfers with a TAP.7 controller, may be used for other purposes with another technology.
- **Reset** (> 7 edges) – Resets all technologies (generates a Type-3 TAP.7 TAPC reset)

The external tool creates an escape sequence by generating one or more TMS(C) edge pairs while the TCK(C) signal is a logic 1 value. Even and odd TMS(C) edge counts beginning with two are given the same meaning.

An escape sequence:

- Can be detected only when a Type-0, Type-1 or Type-2 reset is **not** asserted.
- Begins and ends while the TCK(C) is a logic 1.
- Uses TMS(C) as a clock while the TCK(C) is a logic 1.
- Overlays additional control information onto the normal information that is transferred with the TCK(C) and the TMS(C) without changing the normal information.
- Must be detected when TCK(C) is a logic 1 regardless of the TCK(C) and TMS(C) drive histories.

67.6.4.3.1 Custom escape sequence

An online technology can use the custom escape sequence in any manner it chooses. It is used as the End of Transmission (EOT) of certain data types (formats OScan4-7) with T4 and above TAP.7s.

67.6.4.3.2 Reset escape sequences

A reset event initializes all technologies sharing the TCK(C) and TMS(C) connectivity. A reset escape sequence resets the technology. It is recommended that this reset occur with the falling edge of TCKC. With a TAP.7 controller a reset escape sequence generates a Type-3 reset beginning with the falling edge of TCK(C) following the asynchronous detection of the event.

67.6.5 TAPC State Machine

T1 and above TAP.7s require knowledge of the TAPC state machine state. With T2 and above TAPs this knowledge must be developed independently of the CLTAPC. This is generally accomplished by implementing a TAPC state machine within the EPU. This TAPC state machine provides the function of the state machine within a standard 1149.1 TAPC. It is recommended that this state machine is implemented to change state on the falling edge of TCK(C) as this has numerous advantages to a TAP.7 controller. The EPU does not contain 1149.1 Instruction or Data Registers. The APU controls the state changes of this TAPC state machine with T4 and above TAP.7s.

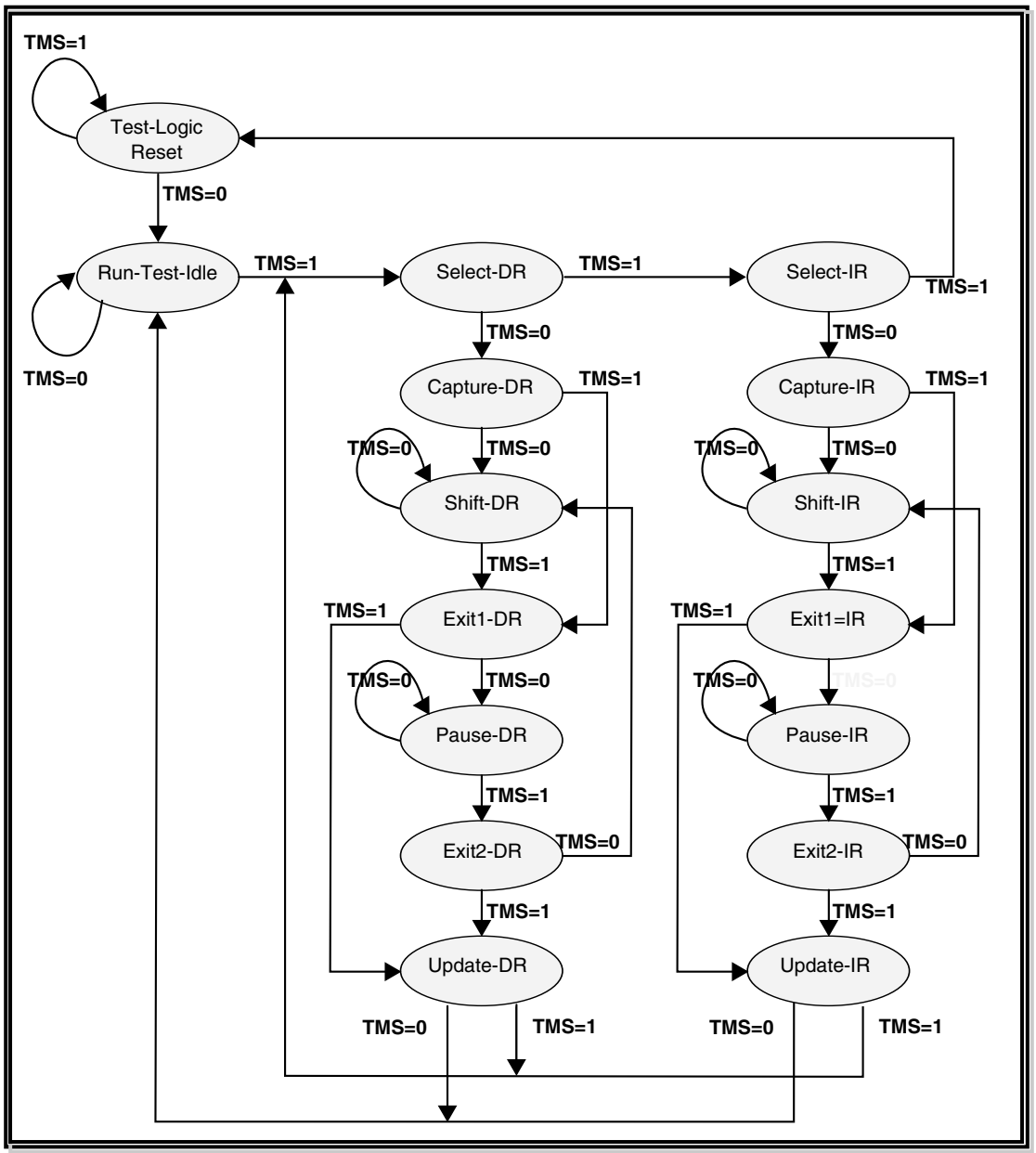


Figure 67-7. Falling edge TAP controller state machine

67.7 EPU (Extended Protocol Unit) Operation

67.7.1 EPU Operation

The EPU adds functionality to the BYPASS and IDCODE instructions.

67.7.1.1 Zero-bit DR-Scans (ZBS)

A zero-bit DR-Scan (ZBS) is a TAPC-state sequence that begins with the *Select-DR-Scan* state and ends with an exit from the *Update-DR* state without an intervening *Shift-DR* state. The external tool controls the EPU using ZBSs.

The use of ZBSs can begin immediately following the *Test-Logic-Reset* state since this state initializes the Instruction Registers of TAPs with either the *BYPASS* or *IDCODE* instructions.

67.7.1.2 Utilizing ZBSs for TAP.7 Controller Functionality

Beginning with a ZBS count of zero, the ZBS count is incremented with each consecutive occurrence of a ZBS without encountering a *Shift-DR* state. This count cannot be incremented past seven. *Run-Test/Idle* is the only state that may occur between consecutive occurrences of ZBSs without terminating the count of consecutive ZBSs.

67.7.1.3 Locking the ZBS Count

When a DR-Scan containing a *Shift-DR* state occurs, and the ZBS count is greater than zero, the ZBS count is not incremented when the *Update-DR* state is reached. It is instead locked at its current value. ZBSs occurring after locking the ZBS count do not affect the locked ZBS count. Locking the ZBS count is the equivalent of storing the count for subsequent use.

67.7.1.4 Control levels

Locking the ZBS count activates a control level that is equal to the locked ZBS count (1-7) when ZBSs are being used for TAP.7 functionality and merely locks the ZBS count otherwise.

1. ZBSs may be used by the STL or another function
2. TAP.7 command generation
3. Reserved control level
4. Accesses to optional TAP.7 scan paths
5. Accesses to optional TAP.7 scan paths

6. Force offline operation (optional)
7. External tool uses

Control level one may be used by the STL since there is no TAP.7 controller functionality associated with this control level.

Control level two enables DR-Scans to create TAP.7 commands. A portfolio of these commands supports the deployment of TAP.7 controllers in the Series or Star Scan Topologies. These commands manage TAP.7 controller registers and perform other functions.

Control level three is reserved for use with a subsequent specification revision.

Control levels four and five provide access to custom (private) scan paths when access to these paths is permitted by a TAP.7 register bit used for this purpose. The EPU provides a 1-bit scan path otherwise. No other EPU functionality is associated with these control levels. The CJTAG module does not support the use of this control level.

Control level six may be optionally used to force offline operation with a TAP.7 controller that supports this capability. The EPU provides a 1-bit scan path for this control level. The CJTAG module does not support the use of this control level.

Control level seven is reserved for external tool use. With this control level, external tool capabilities may be managed using the same infrastructure used to manage TAP.7 controller capability. The EPU provides a 1-bit scan path for this control levels.

67.7.1.5 Exiting a Control Level

A control level is exited when one of the following occurs:

- The *Select-IR-Scan* TAP.7 controller state
- The *Test-Logic-Reset* state
- Certain TAP.7 controller commands (Exit Control Level – ECL register bit)
- An event that synchronizes the operation of TAP.7 controllers

67.7.1.6 Zeroing the ZBS Count

Events causing an exit from a control level also zero the ZBS count. Scan Selection Directives (SSDs) associated with T3 and above TAP.7s zero the ZBS count when the control level has not been locked.

67.7.1.7 Utilizing ZBSs within the STL

After the *Test-Logic-Reset* state, ZBSs are used for basic EPU control. ZBSs that are intended for TAP.7 use are henceforth called "***TAP.7 ZBSs***". ZBSs that are intended for other uses are henceforth called "***Other ZBSs***". Two TAP.7 registers enable the STL's use of ZBSs. Storing these registers causes a control level exit, zeroes the ZBS count, and enables the use of Other ZBSs. Storing the Suspend Register (**SUSPEND**) suspends the use of control levels. Storing the ZBS Inhibit Register (**ZBSINH**) inhibits ZBS detection.

When either of these methods is used, the *Test-Logic-Reset* state enables the use of TAP. 7 ZBSs in addition to initializing the TAP.7 controller.

The operation of the TAP.7 controller changes with the use of these two paths.

When the System Path is used:

- A count of eight consecutive ZBSs is used to create a logic 0 SUSPEND Register value.
- The CLTAPC supplies the TAP.7 scan path and controls the drive of TDO(C), provided it is not bypasses.
- The EPU supplies the TAP.7 scan path (a one bit bypass) when the CLTAPC is bypassed.
- Functions supporting Boundary Scan in a Star Scan Topology are enabled.

When the EPU Path is used:

- When the control level has not been locked, the TDO(C) signal remains high-impedance.
- The command control level may be created
- When commands are used, the TDO(C) signal remains high-impedance except during *Shift-DR* states within CR-Scans, and remains high impedance at other times.
- Auxiliary scan paths may be accessed with control levels four and five.

67.7.2 EPU Registers

67.7.2.1 Global

A global register is a register whose value is stored at the same time and with the same value as registers with the same name in all TAP.7 TAPCs sharing an external tool connection. They are stored using TAP.7 TAPC commands dedicated for this purpose. These commands store a single global register. Global registers manage TAP.7 TAPC functions that affect the synchronized operation of a TAP.7 TAPC. An example of a global register is the Scan Format (SCN_FMT) Register used to define the TAPC signal protocol.

A second means of storing all global registers is available. The selection of the TAP.7 TAPC with the technology selection function stores either the Scan Format register value (short form) or all global registers (long form). Storing all global registers provides a simple means to synchronize the values of all global registers when Technology Selection causes a non-operating TAP.7 TAPC to become operational while other TAPCs are operational.

67.7.2.2 Local

Local registers manage TAP.7 TAPC functions that do not affect the synchronized operation of TAP.7 TAPCs. Local registers are conditionally stored using TAP.7 controller commands dedicated for this purpose. These commands provide for changing the value of these registers one a time with a unique value in a Series Scan Topology based on the TAPC's position on the scan chain. Local registers are stored with a unique value in a Star Scan Topology based on a TAPC address assigned to the TAPC.

Registers programmed with commands are described in this section. The CJTAG module implements all mandatory T4 registers as well as the optional APFC, TRESET, and RDBACK0 registers.There are 16 mandatory and 10 optional registers that can be grouped in classes as shown in Read-only – Configuration and register read back..

- **Control** – TAP.7 controller behaviors and characteristics.
- **Options** – Control of optional functions.
- **Select** – Controls the selection of a TAP.7 for Scan and execution of conditional commands.
- **Read-only** – Configuration and register read back.

Table 67-8. TAP.7 controller register list (managed with commands)

Register type	Global/local	Width	Register mnemonic	Name
Control	Global	1	ECL	Exit Control Level
		1	SUSPEND	Suspend
		1	ZBSINH	ZBS Detect Inhibit
		5	SCNFMT	Scan Format
		1	SSDE	SSD Enable
		2	DLYC	Delay control
		2	RDYC	Ready Control
Options	Local	1	TRESET	Test Reset
		2	APFC	Aux. Pin Func. Cntl.
Select	Local	1	CGM	Cond. Group Member
		1	SGC	Scan Group Candidate
		1	SEDGE	Sampling Edge
Read-only	Local	32	RDBACK0	Read-back 0
		32	RDBACK1	Read-back 1
		32	CNFG0	Configuration Register 0

M = Mandatory register

O = Optional register

67.7.2.3 Register Descriptions

67.7.2.3.1 Exit Control Level (ECL)

Storing this register zeroes the ZBS count and unlocks the control level.

This register is set using the STMC command. It remains logic 1 only momentarily before immediately returning to logic 0.

This register is cleared by a global register load.

67.7.2.3.2 Suspend (SUSPEND)

Setting this register suspends the use of control levels. When this register is a logic 1, the "System Path" (STL) is used.

1 – Suspend use of control levels. Enables "System Path."

0 – Enable use of control levels.

SUSPEND is set with the STMC command.

This register is cleared when 8 consecutive ZBSs are detected without the locking of the ZBS count.

This register is cleared by a global register load.

67.7.2.3.3 ZBS Inhibit (ZBSINH)

Storing to ZBSINH inhibits ZBS detection.

1 – Inhibit ZBS detection.

0 – Enable ZBS detection.

ZBSINH is set with the STMC command.

This register is cleared by a global register load.

67.7.2.3.4 Scan Format (SCNFMT[4:0])

Set using Store Scan Format (STFMT) command.

Read using Scan String (SCNS) command.

Table 67-9. Scan Format Register Values

TAPC.7 Class	SCNFMT	Description
T2–T4	00000	**JScan0:** • 1149.1 Compliant Operation. • Requests the coupling of the CLTAPC when the Run-Test/Idle state is reached.
	00001	**JScan1:** • 1149.1 Compatible Operation • Requests the decoupling of the CLTAPC when the Run-Test/Idle state is reached.
	00010	**JScan2:** • 1149.1 Compatible Operation. • Requests the coupling or decoupling of the CLTAPC when the Run-Test/Idle state is reached based on the value of the CACT register

Table continues on the next page...

Table 67-9. Scan Format Register Values (continued)

TAPC.7 Class	SCNFMT	Description
T3–T4	00011	**JScan3:** • Non-compatible 1149.1 Operation • Allows the detection of a T2 TAP.7 controller that is deployed in a Star-4 topology. • Enables TDOC signal behavior that prevents drive conflicts in a Star-4 Scan Topology. • Enables the use of SSDs
T4	01000	**OScan0** • Data rate dependent components • Single TAP.7 communication • Debug and Test applications
	01001	**OScan1** • TAP.1 components • Single/Multi-TAP.7 communication • Debug and Test applications
	01010	**OScan2** • TAP.1 components • No TDI or TDO in non-shift states • Debug applications
	01011	**OScan3** • TAP.1 components • No TDI in non-shift states • No TDO in any state

Table continues on the next page...

Table 67-9. Scan Format Register Values (continued)

TAPC.7 Class	SCNFMT	Description
T4	01100	**OScan4** • Data rate dependent components • Single TAP.7 communication • Debug and Test applications
	01101	**OScan5** • TAP.1 components • Single/Multi-TAP.7 communication • Debug and Test applications
	01110	**OScan6** • TAP.1 components • No TDI or TDO in non-shift states • Debug applications
	01111	**OScan7** • TAP.1 components • No TDI in non-shift states • No TDO in any state
	10000	**MScan** • Data rate dependent components • Multi TAP.7 communication • Directed CID assignment • Virtually any IP with compliant or non-compliant 1149.1 behavior
	10001 – 11111	**Reserved**

The use of a Scan Format value other than 00000b – 10000 shall not change the value of the SCNFMT Register.

67.7.2.3.5 Scan Selection Directive Enable (SSDE)

This register specifies whether the Scan Selection State may be managed using the scan selection directives.

0 – Scan selection directives are not enabled.

1 – Scan selection directives are enabled.

This register is set using the STMC command.

This register is read using SCNS (Scan String), RDBACK0.

67.7.2.3.6 Delay Control (DLYC[1:0])

The Delay Control register provides a means for the external tool to delay the completion of an SP. This delay is essentially an external tool stall. This delay may be zero, one, two, or n TCKC periods. This delay is especially useful when the TCKC is sourced by the TS and a Standard-to-Advanced Protocol adapter is added to an external tool supporting only the Standard Protocol. It allows an external tool design additional time to receive TDO, advance the external tool TAPC state, receive TMS and TDI information, and begin generation of a new SP payload. The DLYC register should be programmed before the use of the Advanced Protocol begins after the use of the Standard Protocol.

Set using the STMC command.

Read using SCNS, RDBACK0

Bits 18-17 of the "Global Register."

67.7.2.3.7 Ready Control (RDYC[1:0])

The Ready Control register defines behavior exhibited in the SP payload output bit frames when the STL stall opportunities (RDY bits) are included as control information. This register defines the number of additional bits inserted in these bit frames. These bits provide more time for a high-performance external tool to ascertain whether the TAP.7 controller is ready to complete the SP payload. The use of these bits makes a high-performance external tool design easier as input and output buffer delays play less of a role in limiting the TAP.7 controller performance. The RDYC register should be programmed before the use of the Advanced Protocol begins.

Set using the STMC command.

Read using SCNS, RDBACK0.

Bits 18-17 of the Global register state.

67.7.2.3.8 Test Reset (TRESET)

Asserts a test reset to the STL.

0 – The nSYS_TRST and SYS_TMS signals presented to the STL is not influenced by this bit.

1 – The nSYS_TRST presented to the STL is asserted and the SYS_TMS signal is a logic 1.

Set using the STC1 command.

The effects of the TRESET Register may be modified by a private TAP.7 controller register. When this private register is unimplemented or has a value of zero, the TRESET register causes the maximum initialization of the STL with nSYS_TRST and the SYS_TMS logic 1 value. Otherwise, the value of the private register may modify the function of the TRESET Register bit.

When the TRESET Register is a logic 1, the STL scan path appears broken as the state of CLTAPC remains *Test-Logic-Reset* while the EPUTAPC state progression continues. When the operation of the EPUTAPC and CLTAPC controllers is coupled, as with a T1 TAP.7, the state of these two TAPCs may be resynchronized to the *Run-Test/Idle* state by:

- An STC1 Command that sets the TRESET Register value to a logic 0.
- A stay in the Run-Test/Idle state of two or more SYS_TCK periods immediately following the Update-DR state of this command

A failure to follow the guidelines is considered a programming error and will result in erroneous system operation. The EPUTAPC and CLTAPC states will not be synchronized as they progress through the state diagram.

67.7.2.3.9 Auxiliary Pin Function Control (APFC[1:0])

This register enables TDI and TDO alternate functions when 2-pin TAP.7 operation is selected.

0x – TDI and TDO have JTAG function.

1x – TDI and TDO have alternate function.

Set using STC2 command.

Read using SCNS, RDBACK0.

67.7.2.3.10 Conditional Group Member (CGM)

Used by STC1, STC2, STTESTM, and EXC3 – EXC0 commands as the “condition” for updating registers.

Set and cleared using MCM and SCNB commands.

Read using SCNS, RDBACK0.

67.7.2.3.11 Scan Group Candidate (SGC)

This register determines whether the STL is coupled when the *Run-Test/Idle* state is reached when scan formats other than JScan0 and JScan1 are used.

- **0** – Decoupling action.
- **1** – Coupling action.

Set and cleared using STMC, MSC, and SCNB commands.

Read using SCNS, RDBACK0.

67.7.2.3.12 Sampling Edge (SEDGE)

The Sampling Edge register defines the TCKC edge used to sample the TMSC input.

- **0** – Sample TMSC on the TCKC falling edge.
- **1** – Sample TMSC on the TCKC rising edge.

Set and cleared using STC1 command.

Read using SCNS, RDBACK0

67.7.2.3.13 Read Back 0 (RDBACK0)

Non-transport related register information.

Read using SCNS command.

Unimplemented registers shall read back as "0".

Table 67-10. RDBACK register format

Bit	Width	Register Mnemonic	Name
4:0	5	SCNFMT	Scan Format
6:5	2	PWRMODE	Power-control Modes
7	1	FRESET	Functional Reset
8	1	TRESET	Test Reset
9	1	SGC	Scan Group Candidate
		CGM	Conditional Group Member
11	1	SSDE	Scan Selection Directive Enable
13:12	2	TOPOL	Topology
14	1	SREDGE	Sampling Edge
16:15	2	DLYC	Delay Control
18:17	2	RDYC	Ready Control
20:19	2	APFC	Auxiliary Pin Functional Control

Table continues on the next page...

Table 67-10. RDBACK register format (continued)

Bit	Width	Register Mnemonic	Name
22:21	2	STCKDC	SYS_TCK Duty Cycle
31:23	11	Reserved	Read as a logic 0

67.7.2.3.14 Configuration Register 0 (CNFG0)

Table 67-11. Configuration Register 0 Format

Field	Bit #	Width	Name	Description
CNFG (Class)	3:0	4	CLASS[3:0]	TAP.7 Class 0000 – TAP.1 or TAP.7 Class T0 0001 – Class T1 0010 – Class T2 0011 – Class T3 0100 – Class T4 0101 – Class T5
CNFG (Revision)	7:4	4	REV[3:0]	TAP.7 specification revision. 0000 – Reserved 0001 – IEEE Std 1149.7-2008 0010 – 1111 - Reserved
CNFG (Format)	11:8	4	CNFGFMT[3:0]	Configuration Register Format xxx1 – CNFG0[31:12] are implemented xx1x – CNFG1 implemented x1xx – CNFG2 implemented 1xxx – CNFG3 implemented

Table continues on the next page...

Table 67-11. Configuration Register 0 Format (continued)

Field	Bit #	Width	Name	Description
T1 Options	12	1	RDBKS	RDBACK supported
				0 – Unsupported
				1 – Supported
	13	1	TRESETS	TRESET supported
				0 – Unsupported
				1 – Supported
	14	1	FRESETS	TRESET supported
				0 – Unsupported
				1 – Supported
	17:15	3	PWRMODES[2:0]	PWRMODES[x] – Power Control Mode
				0 – Unsupported
				1 – Supported
	18	1	PCMR	PCMR – PM default POWRMODE value
				0 – No
				1 – Yes
T2 Options	19	1	DCAS	DCAS – Decouple at start-up
				0 – No
				1 - Yes

Table continues on the next page...

Table 67-11. Configuration Register 0 Format (continued)

Field	Bit #	Width	Name	Description
T4 Options	21:20	2	SSCANS[1:0]	x0 – SScan1:0 Unsupported
				x1 – SScan1:0 Supported
				0x – SScan3:2 Unsupported
				1x – SScan3:2 Supported
	24:22	3	OSCANS[2:0]	xx0 – OScan3:2 Unsupported
				xx1 – OScan3:2 Supported
				x0x – OScan5:4 Unsupported
				x1x – OScan5:4 Supported
				0xx – OScan7:6 Unsupported
				1xx – OScan7:6 Supported
	25	1	APFCS	Auxiliary Pin Function Control supported
				0 – Unsupported
				1 – Supported
	26	1	TAPWIDS	TAP Width Default
				0 – 2-pin supported
				1 – 4-pin supported
	27	1	TAPWLCK	TAP Width Locked
				0 – Not locked
				1 - Locked
	28	1	STCKDCS	SYS_TCK Duty Cycle supported
				0 – Unsupported
				1 – Supported
Reserved	31:29	3	Reserved	Reserved, read as zero

67.7.2.3.15 Register reset values

Table 67-12. Reset values of TAP.7 controller registers managed with commands

Register type	Width	Register mnemonic	Name	Values for reset type			
				0	1 – 3	4	5
Control	1	ECL	Exit Control Level	0	0	0	NC
	1	SUSPEND	Suspend	0	0	0	
	1	ZBSINH	ZBS Detect Inhibit	0	0	0	
	5	SCNFMT	Scan Format	DF	DF	DF	
	1	SSDE	SSD Enable	0	0	0	
	2	DLYC	Delay control	00	00	00	
	2	RDYC	Ready Control	00	00	00	
Options	1	TRESET	Test Reset	0	0	0	NC
	2	APFC	Aux. Pin Functional Cntl.	DP	DP	NC	
Select	1	CGM	Conditional Group Member	0	0	0	NC
	1	SGC	Scan Group Candidate	DCA	DCA	DCA	
	1	SREDGE	Sampling Rising Edge	0	0	0	
Read-only	32	RDBACK0	Read-back 0	N/A	N/A	N/A	N/A
	32	CNFG0	Configuration Register 0				

NC = No Change

DF = Default Scan Format

DM = Default Power Control Mode

DCA = Default Coupling Action

DT = Default Topology

DP = Default Pin Function

67.7.3 EPU Commands

There are two command types:

- Two Part Commands
- Three Part Commands

Any number of two-part and three part commands may be issued in any combination before the command control level is exited. If the command control level is exited before a two- or three-part command is completed, the command is aborted and becomes a NOP.

67.7.3.1 Two Part Commands

TAP.7 controller commands are 10-bit values. The 10-bit commands are created by two consecutive DR-Scans when the control level is locked at the command control level (two). The first DR-Scan provides a five-bit command opcode. The second DR-Scan provides a five-bit operand. These two scans are called the command part one (CP1) and command part two (CP2), respectively. A command created entirely with these two scans is called a two-part command.

67.7.3.2 Three Part Commands

Some commands are used to send and/or receive data values other than values embedded in the command's operand. These commands are called three-part commands. With these commands, CP1 and CP2 are followed by an additional DR-Scan to transport a data value to or from an EPU scan path. When used for this purpose, the DR-Scan is called a Control Register Scan (CR-Scan). The CR-Scan is a minimum of zero bits in length with there being no maximum length. The CR-Scan path is described in SECTION.

67.7.3.3 Command Sequence

The two 5-bit values concatenated to create the 10-bit command represent the number of Shift-DR states between the *Capture-DR* and *Update-DR* states of CP1 and CP2. The count within CP1 creates the MSBs of the command while the count within CP2 creates the LSBs of the command. The command is decoded when the *Update-DR* state of the second DR-Scan is reached. A determination as to whether the command is a two- or three-part command is made at this point. The command format is shown in the following figure.

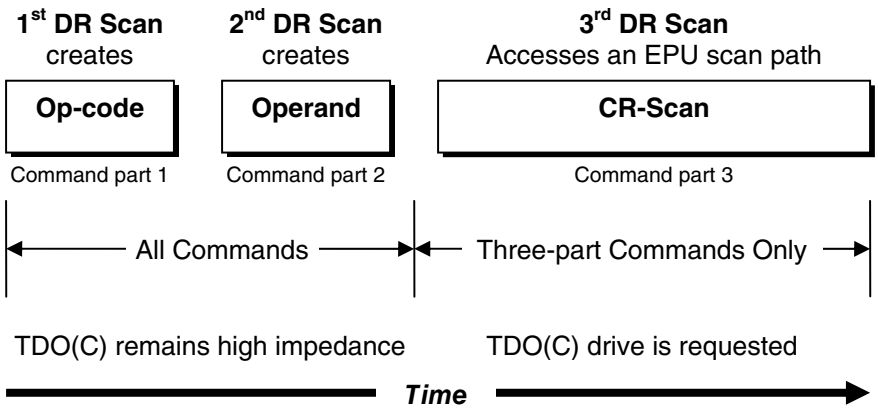


Figure 67-8. DR_Scan sequence used for command creation

The 10-bit command created by CP1 and CP2 determines the command's function and whether the command is a two- or three-part command. If the command is a two part command, the function specified by the command is performed when command part 2 completes. If the command is a three-part command, the DR-Scan following CP2 performs a TAP.7 controller function designated by the 10-bit controller command. This DR-Scan is called a ***Control-Register Scan*** (CR-Scan) and can be any length. A CR-Scan has many of the attributes of the DR-Scan as it moves data between the external tool and TAP.7 controller and accesses various EPU scan paths. There are many two part commands, but only three three-part commands, each having a special purpose.

67.7.3.4 Commands

There are 11 mandatory and 5 optional commands that can be grouped in classes as shown in the following list:

- Store – The operand is stored in a register or causes an action.
- Select – Controls participation of Scan and conditional command execution.
- Scan – Inputs and outputs with a CR-Scan.
- Enumerate – Controller ID allocation and de-allocation.
- Private – Commands available for chip-specific definition.

The definition of the commands with their operands is shown in the following table.

Table 67-13. – TAP.7 controller command list

Command class	Opcode count	Mnemonic	Name	TAP.7 class			
				1	2	3	4
Store	0x00	STMC	Store Miscellaneous Control	M	M	M	M
	0x01	STC1	Store Conditional 1-bit	M	M	M	M
	0x02	STC2	Store Conditional 2-bit	M	M	M	M
	0x03	STFMT	Store Format	-	M	M	M
	0x04	STDCST	Store Data Channel State	-	-	-	-
Select	0x06	MCM	Make Cond. Group Member	M	M	M	M
	0x07	MSC	Make Scan Group Candidate	-	M	M	M
Scan	0x08	SCNB	Scan Bit	M	M	M	M
	0x09	SCNS	Scan String	M	M	M	M
Enumerate[1]	0x0A	CIDA	Allocate Controller ID	-	-	M	M
	0x0B	CIDD	De-allocate Controller ID	-	-	M	M
Reserved	0x0C–0x1F	Reserved	Reserved	-	-	-	-

1. Enumerate commands are not currently supported by the CJTAG module.

67.7.3.4.1 Store Commands

Store commands are two-part commands that are available in both Star and Series Scan Topologies. The Store Miscellaneous Control (STMC) Command performs control functions that affect all TAP.7 controllers sharing an external tool connection. It stores values in one-bit and two-bit registers.

The Store Conditional 1 bit (STC1) and Store Conditional 2 bit (STC2) Commands store one-bit and two-bit values in registers, respectively. These commands operate two ways:

1. Unconditionally storing registers in all TAP.7 controllers or
2. Storing registers in only the TAP.7 controllers that have a logic 1 Conditional Group Member (CGM) Register value.

The Store Format (STFMT) Command unconditionally stores a five-bit Scan Format value.

The Store Data Channel State (STDCST) Command stores a five-bit value identifying the states supporting transport.

67.7.3.4.2 Select Commands

Select commands are two-part commands that are available in both Star and Series Scan Topologies. Their primary use is to select TAP.7 controllers of interest in a Star Scan Topology. These commands define the TAP.7 controller(s) participating in scans and the TAP.7 controllers that conditionally execute STC1 and STC2 Commands.

The Make Conditional Group Member (MCM) Command specifies which TAP.7 controllers execute store conditional commands as it manages the Command and Path Enable (CAPE) Register. This register also affects the execution of private commands and is utilized in EPU scan path selection.

The Make Scan Group Candidate (MSC) command manages the SGC register. This register determines whether the STL is coupled when the *Run-Test/Idle* state is reached when scan formats other than JScan0 and JScan1 are used. Select commands execute only when the CIDI Register is a logic 0, indicating the TAP.7 controller's CID is valid, and are treated as no operations otherwise.

67.7.3.4.3 Scan Commands

Scan commands are three-part commands that are available in both Star and Series Scan Topologies. The CR-Scan portion of Scan Bit (SCNB) and Scan String (SCNS) Commands is used to set and test bits (SCNB) or transfer strings of bits (SCNS). The commands may be used to access both public and private registers.

67.7.3.5 Command Definitions

67.7.3.5.1 Store Commands

67.7.3.5.1.1 Store Miscellaneous Control Command (STMC)

Table 67-14. Store Miscellaneous Control

STMC		Store Miscellaneous Control				
Opcode	**Operand**	**bbb**		**x**	**y**	**Description**
00000	bbb x y	000	StateCtl	0	0	NOP
				0	1	ExitCmdLev(ECL)
				1	0	Exit/Suspend (SUSPEND) = 1
				1	1	ZBS Inhibit (ZBSINH) = 1
		001	ScanCtl	0	-	SGC = y
				1	-	CGM = y
		010	RdyCtl	-	-	RDYC = xy
		011	DlyCtl	-	-	DLYC = xy
		100	ScnFunc	0	-	SSDE = y
				1	0	Star-4 Topology Test
				1	1	Reserved
		101	Reserved	-	-	Reserved
		110				
		111				

67.7.3.5.1.2 Store Conditional 1-bit (STC1)

Table 67-15. Store Conditional 1 bit (STC1)

STC1		Store Conditional 1 bit			
Opcode	**Operand**	**c**	**bbb**	**CAPE**	**Description**
00001	c bbb v	0	000	-	SREDGE = v
			001	-	Reserved
			010	-	TRESET = v
			011 – 111	-	Reserved
		1	–-	0	No change
			000	1	SREDGE = v
			001	1	Reserved
			010	1	TRESET = v
			011 – 111	1	Reserved

67.7.3.5.1.3 Store Conditional 2-bit (STC2)

Table 67-16. Store Conditional 2 bit (STC2)

STC2		Store Conditional 2 bit			
Opcode	**Operand**	**c**	**bb**	**CAPE**	**Description**
00010	c bb vv	0	00	-	Reserved
			01	-	STCKDC = vv
			10	-	APFC = vv
			11	-	Reserved
		1	–-	0	No change
			00	1	Reserved
			01	1	STCKDC = vv
			10	1	APFC = vv
			11	1	Reserved

67.7.3.5.1.4 Store Scan Format (STFMT)

Table 67-17. Store Scan Format (STFMT)

STFMT		Store Scan Format
Opcode	**Operand**	
00011	nnnnn	SCNFMT = nnnnn

67.7.3.5.2 Select Commands

This section describes the select commands with their operands.

67.7.3.5.2.1 Make Conditional Group Member (MCM)

This command operates as a no operation when the CIDI Register is a logic 1 (The Controllers CID is invalid). Otherwise it sets the CGM bit in a TAP.7 controller that has a valid CID and the CID matches the operand CID field (iiii). The CGM Register is also stored with the STMC and SCNB commands.

Table 67-18. Make Conditional Group Member (MCM)

MCM		Conditional Group Member	
Opcode	**Operand**	**m**	**Description**
00110	m iiii	0	CGM bit of the non-targeted controller is cleared
		1	CGM bit of the targeted controller is set. CGM bit of a non-targeted controller is unaffected.

67.7.3.5.2.2 Make Scan Group Candidate (MSC)

This command operates as a no operation when the CIDI Register is a logic 1 (The Controllers CID is invalid). Otherwise it sets the SGC bit in a TAP.7 controller that has a valid CID and the CID matches the operand CID field (iiii). The SGC Register is also stored with the STMC and SCNB commands.

Table 67-19. Make Scan Group Candidate (MSC)

MSC		Make Scan Group Candidate	
Opcode	**Operand**	**m**	**Description**
00111	m iiii	0	SGC bit of the non-targeted controller is cleared
		1	SGC bit of the targeted controller is set. SGC bit of a non-targeted controller is unaffected.

67.7.3.5.3 Scan Commands

67.7.3.5.3.1 Scan Bit (SCNB)

This is a 3-part command. The operand chooses the bit to be read or written by the CR-Scan.

Table 67-20. Scan Bit (SCNB)

SCNB			Scan Bit		
Opcode	**Operand**	**CR-Scan**	**yyyyy**	**Description**	**R/W**
01000	yyyyy	-	0	Scan Group Candidate (SGC)	Write only
		-	1	Conditional Group Member (CGM)	Write only
		-	2	CNFG[0]	Read only
		-	3	CNFG[1] (not implemented)	Read only
		-	4	CNFG[2] (not implemented)	Read only
		-	5	CNFG[3] (not implemented)	Read only
		-	6	Functional reset requested (FRESETR) (not implemented)	Read only
		-	7	Series topology detect	Write only
		-	15 – 8	Reserved	Read only

67.7.3.5.3.2 Scan String (SCNS)

This is a 3-part command. The operand chooses the string to be read or written by the CR-Scan.

Table 67-21. Scan String (SCNS)

SCNS			Scan String		
Opcode	Operand	CR-Scan	yyyyy	Description	R/W
01001	yyyyy	-	0	RDBACK[0] Register [31:0]	Read only
		-	1	RDBACK[1] Register [31:0] (not implemented)	Read only
		-	2	CNFG[0] Register [31:0]	Read only
		-	3	CNFG[1] Register [31:0] (not implemented)	Read only
		-	4	CNFG[2] Register [31:0] (not implemented)	Read only
		-	5	CNFG[3] Register [31:0] (not implemented)	Read only
		-	6 - 7	Reserved (read logic 0)	Read only
		-	31 - 8	Private – available for customization	

67.7.4 EPU operating states

The TAP.7 controller and STL share the use of ZBSs. The EPU has three operating states as shown in the following figure:

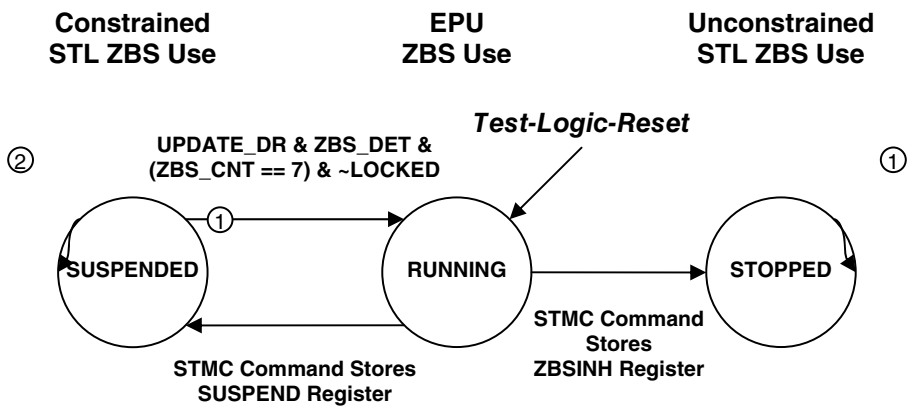


Figure 67-9. EPU operating states

The values of the SUSPEND and ZBSINH Registers determine the type of operation as shown in the following table. The Test-Logic-Reset state sets the SUSPEND and ZBSINH Register values to a logic 0.

Table 67-22. ZBS use state characteristics with both TAP.7 and other ZBS use

ZBSINH	SUSPEND	EPU operating State	ZBS use by:	Until
0	0	RUNNING	EPU	Command allocates to STL
0	1	SUSPENDED	STL	ZBS sequence allocates to EPU
1	0	STOPPED	STL	Test-Logic-Reset state allocates to EPU
1	1	Not possible		

67.7.5 System and EPU Paths

The System path is utilized when the external tool requires access to system (STL) resources. This path is used when either the SUSPEND Register is a logic 1 or the ZBS count is less than or equal to one. The EPU path is used otherwise.

The EPU scan path is constructed from five types of EPU scan paths:

- **Bypass** – A 1-bit scan path
- **Bit** – Supports set and test bit operations (a 1-bit shift register)
- **String** – Support 32-bit reads or 32-bit writes
- **Enumerate** – A 36-bit scan path that transports a TAPC address with T3 and above TAP.7s
- **Auxiliary** – An n-bit scan path that transports 1 – n bits using a conventional IEEE 1149.1 style DR-Scan path using Control Levels 4 and 5.

67.8 APU (Advanced Protocol Unit) Operation

67.8.1 Overview

The Advanced Protocol Unit (APU) is placed between the EPU and the RSU to create the T4 TAP.7 controller. The APU delivers advanced capabilities with a number of operating modes (new scan formats) that utilize the Standard Protocol and the Advanced Protocol in both the narrow and wide T4 TAP.7 versions.

The APU enables scan transfers with two signals by serializing the information related to a TAPC state machine state. It provides a number of serialization formats to balance scan flexibility and scan performance. It multiplexes the use of the TMSC signal for T4 TAP.7 controller functions. The serialization is bypassed to provide T3 TAP.7 capability.

With a wide TAP.7, the chip architect may choose to implement the TDIC and TDOC signal functions as fixed functionality (the T3 TDIC and TDOC functions) or as programmable between the T3 TDIC and TDOC signal functions and an alternate function using a TAP.7 controller register. In this case, the default signal function may be either of the programmable choices. Programmable TDIC and TDOC functionality provides a means to fully utilize these signals in any scan topology as the T3 TAP.7 functionality is fully supported.

The entire EPU infrastructure (i.e., ZBS detection, control-level management, and command generation) is utilized when either the Standard Protocol or the Advanced Protocol is used, as these TAP.7 controller attributes are controlled entirely from the TAPC state machine state progression. The EPU is unaware of the APU's existence. All T4 features are managed outside of the EPU.

The TAP.7 serialization schemes are specified with scan formats that complement the JScan 0-3 Scan Formats inherited from the T3 TAP.7. The scan formats are called advanced scan formats as they support two-signal operation.

Two groups of scan formats (MScan and OScan0-7) provide a number of options for the serialization of scan information for use in a Star-2 scan topology. The OScan Scan Format is broken into two groups with the same function, one group has better performance but may only be used with an external tool sourced TCKC. The external tool chooses a scan format that matches the constraints imposed by the application, chip components, and possibly the external tool itself.

The MScan and OScan formats address Test and Debug use cases. The mandatory MScan and OScan0-1 formats emphasize flexibility over performance. The remaining optional Oscan2-7 formats emphasize performance over flexibility by not transmitting unneeded information such as TDI and TDO in non-shift TAPC state machine states.

The optional capabilities added with T4 and above TAP.7s change the bit sequences seen at the TAP.7 signals. A TAP.7 controller comprehends only the bit sequences generated by the use of the features it supports. It is never exposed to bit sequences it does not comprehend. A check for supporting an enabled feature is made before the feature is used (alters the bit sequences). When a T4 TAP.7 controller detects the use of an unsupported scan format, it halts its operation and places itself in a state where its operation may be resumed. This state is henceforth called "offline". A TAP.7 controller that is not offline is henceforth called "online". While offline the TAP.7 controller awaits the detection of a

special signaling sequence called the Online Activation Protocol. This protocol sequence places an offline TAP.7 controller online and synchronizes the operation of all online TAP.7s.

The Advanced Protocol Unit supports the stall of a transfer by both the external tool and STL at any TAPC state.

The APU provides the following functions:

- Bypass (for operation with the Standard Protocol)
- Scan control (for SP use)
- Configuration Control (for CP use)

The Scan Control, Configuration Control, and Transport Control functions are idled when the Standard Protocol is used as the Bypass Function connects the TCKC and TMSC pins to the EPU.

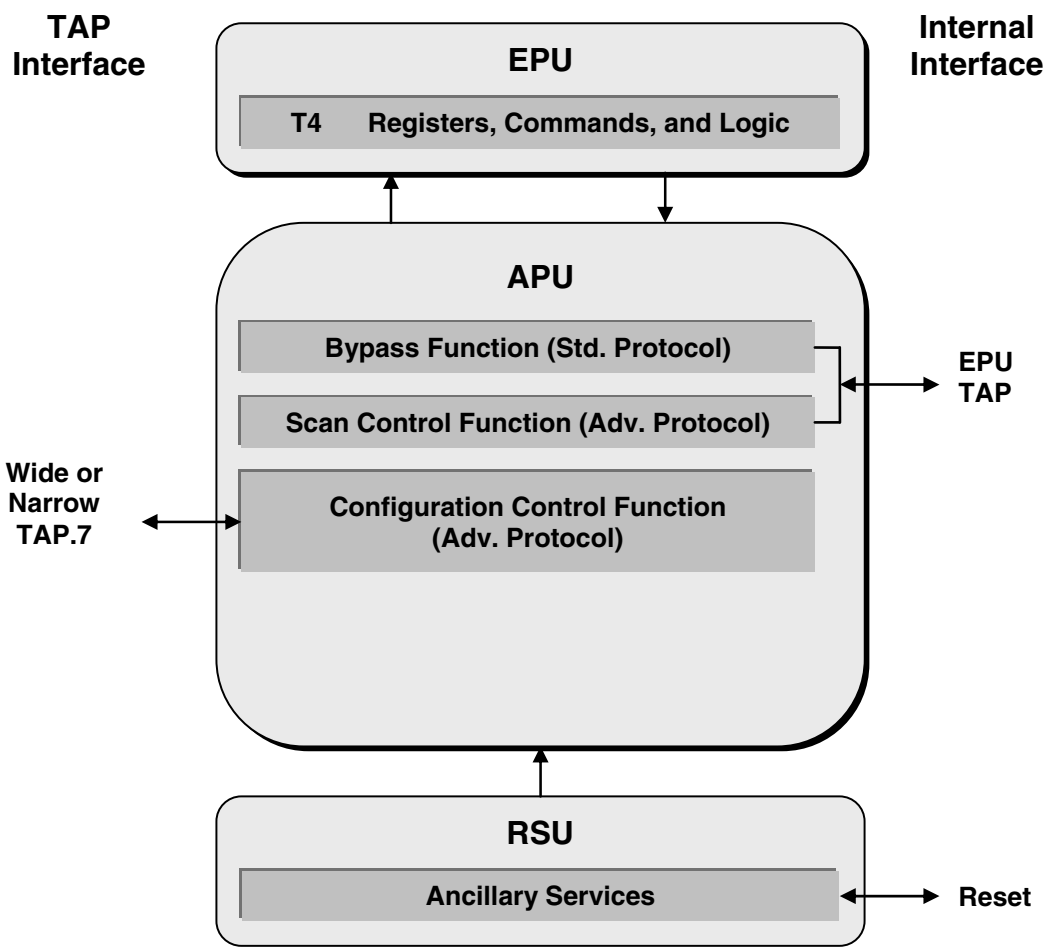


Figure 67-10. Conceptual block diagram of the APU functions

The APU's internal interface includes a connection to EPU registers, the EPU's TAP interface, connections to chip-level data channels (when they are implemented), and reset signaling with the EPU. The APU's TAP interface provides mutually exclusive use of the TMSC pin for:

- Bypass Function
- Scan Control Function
- Configuration Control Function

When the Standard Protocol is used, the APU's Bypass function connects the EPU directly to the TAP.7 pins. When the TDIC and TDOC pin functions are not available (narrow TAP.7 or TDIC and TDOC pin functions are an alternate function), the APU sources a logic 1 to the EPU inputs associated with these pins.

When the Advanced Protocol is used, the APU multiplexes the use of the TMSC pin between the other APU functions. Scan Control acts as a serial to parallel converter, translating the parallel information at the EPU's TAP interface into bidirectional serial transfers. Since the Scan Control is literally a protocol adaptor, the entire EPU infrastructure (ZBS detection, control-level management, and command generation) is utilized when either the Standard or Advanced Protocol is used, as these TAP.7 controller attributes are controlled entirely from the TAPC state progression.

67.8.2 Operation

The conceptual view of the APU operation is shown in the following figure.

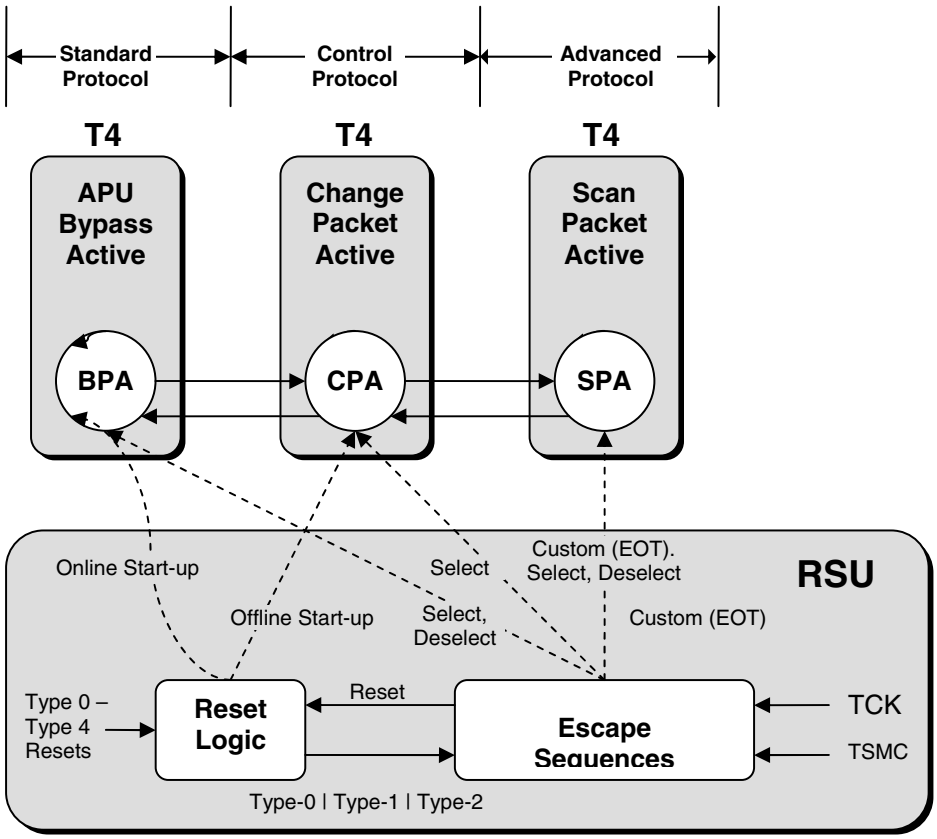


Figure 67-11. Conceptual view of APU operation

The APU operating states managing the use of the TAP pins and the ancillary (RSU) services use of the TMSC pin.

APU operating states:

- ***BPA*** – **B**y**P**ass **A**ctive
- ***CPA*** – **C**hange **P**acket **A**ctive
- ***SPA*** – **S**can **P**acket **A**ctive

67.8.2.1 BPA

The *BPA* state allocates the use of the TMSC pin to the Standard Protocol. This state is entered two ways:

- As a result of a TAP.7 controller reset when the TAP.7 controller starts up online.
- From the CPA state when a TAP.7 controller register write specifies the use of the Standard Protocol.

67.8.2.2 CPA

The *CPA* state allocates the use of the TMSC pin to a CP. This state is entered in one of three ways:

- As a result of a TAP.7 controller reset when the TAP.7 controller starts up offline.
- A value specifying the use of the Advanced Protocol is written to the SCNFMT Register when using the Standard Protocol.
- A TAP.7 controller register write occurs when using the Advanced Protocol.

67.8.2.3 SPA

The SPA state allocates the use of the TMSC pin to an SP. This state is entered when the SCNFMT Register specifies the use of the Advance Protocol when a CP ends.

67.8.3 Escape sequences

Escape-sequence detection provides services to the Reset Logic, Scan, Configuration Control, and Transport Control functions as outlined below:

- **Scan and Transport Control** – End of Transfer (EOT) escape sequence detected.
- **Configuration Control** – Synchronize Advanced Protocol (SAP) escape sequence.
- **Reset Logic** – Reset (RES) escape sequence detected.

An EOT escape sequence is used to increase the efficiency of OScan Scan formats.

An SAP escape sequence is used to place TAP.7 controllers online.

A RES escape sequence asynchronously causes generation of a Type-3 Reset. Following a RES escape sequence, the drive of the TMSC pin is inhibited so it is not driven when TCKC becomes a logic 0 immediately following the RES escape sequence.

67.8.4 Signal behaviors

The TCKC signal:

- The test clock

The TMSC signal:

- An input when the Standard Protocol is used
- Bi-directional when the Advanced Protocol is used

The TDI and TDO signals:

- Deleted with a narrow TAP.7
- One of two functions with a wide TAP.7
  - Fixed – as the TAP.7 TDIC and TDOC pin functions, or
  - Programmable – as the TAP.7 TDIC and TDOC signal functions or alternate functions with a TAP.7 controller reset establishing the default signal function.

67.8.5 APU Functions

67.8.5.1 Bypass Active (BPA) Function

When the Standard Protocol is used, the APU connects the EPU inputs and outputs directly to the TAP.7 pins. If the TDIC and TDOC pin functions are not present (e.g., a narrow TAP.7 or the TDIC and TDOC pins have an alternate function assigned to them), the APU provides a logic 1 to the EPU for unsupported pin functions and ignores the EPU's TDO output data.

67.8.5.2 Scan Packet Active (SPA) Function

The Scan Control function is encapsulated within the Scan Packet Active (SPA) state. With the Standard Protocol, the external tool and TAP.7 exchange the scan information associated with a single TAPC state machine state in one TCK period using the TMS(C), TDI(C), and TDO(C) pins. With the Advanced Protocol all or part of this information is exchanged serially within a Scan Packet (SP) as shown in . There is a one to one correspondence between SPs and TAPC state machine state changes, except when the SPA state precedes the CPA state. In this case the SP preceding a CP does not advance the TAPC state machine state.

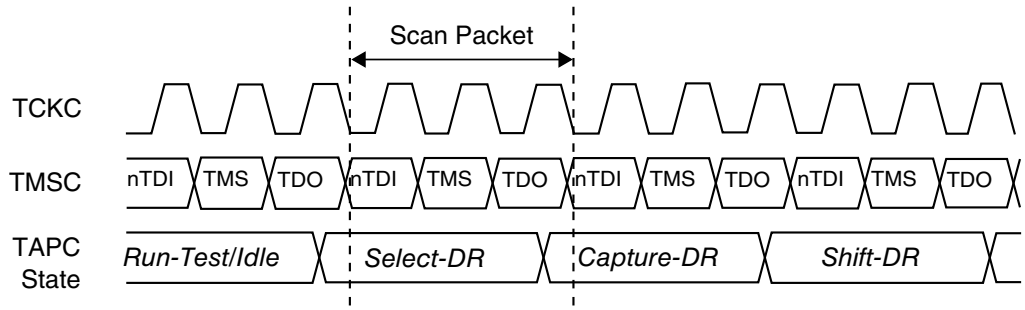


Figure 67-12. Scan Packet Sequence

A comparison of the parallel transfer of TAP information with the Standard Protocol and the serialization and de-serialization of the TAP information using the TCKC and TMSC pins with the Advanced Protocol is shown in FIGURE. The IEEE 1149.1 signals between the EPU and STL are prefixed with SYS_.

The APU manages the serialization and de-serialization of the scan information. The APU converts the TDI and TMS information within a Scan Packet (SP) into the TMS and TDI information presented to the EPU. It converts the combination of the EPU TDO and EPU TDO_OE signals into the TDO information in the same SP. If the EPU indicates that the TDO pin drive is not enabled, logic 1 TDO data is created, otherwise the TDO data is the supplied by either the EPU or STL.

Information within a Scan Packet is first passed from the external tool to the TAP.7 followed by information passed from the TAP.7 to the external tool. In this example the scan format specifies the exchange of the TMS, TDI, and TDO TAPC information. Each TAPC state machine state takes three TCKC periods. Other scan formats specify the exchange of different mixes of information. Control information is added with the most flexible scan formats. Some scan formats send less information and consequently take fewer TCKC periods.

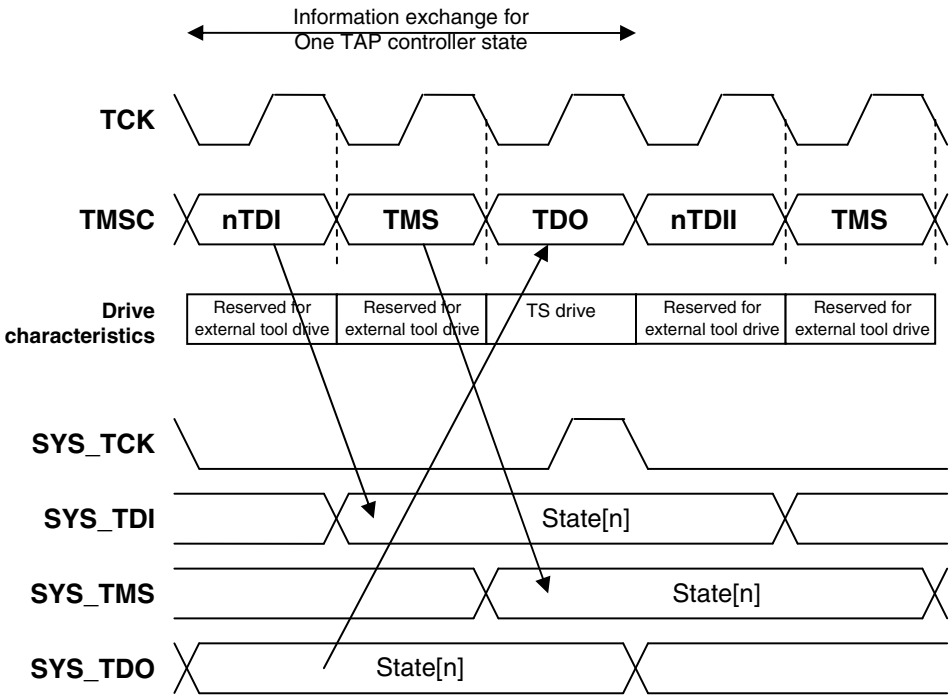


Figure 67-13. Serialization and De-serialization of Scan Packet Information

In order to produce comparable TAPC state machine state rates with a TAP.1 and TAP.7 controller:

- Full-cycle timing between the external tool and TAP.7 allows operation at higher TCKC frequencies.
- The number of bits transferred to advance a TAPC state machine is minimized. (reaching as few as one for some scan formats.)

The TAP.7 controller provides both full-cycle bit timing (falling TCKC edge to falling TCKC edge) and half-cycle bit timing (falling TCKC edge to rising TCKC edge), with a TAP.7 register used to define the type of timing used. Half-cycle timing is used as the default following a TAP.7 controller reset. The TCKC frequency is reduced to accommodate half-cycle timing.

With full-cycle timing, the TMSC input and output data are both output with and sampled with the falling edge of the TCKC. This means that the entire TCKC period may be spent transferring a bit from the external tool to the TAP.7 or vice-versa. The TCKC frequency can therefore be maximized when full-cycle timing is used. With half-cycle timing, the APU samples the TMSC input data with the rising edge of the TCKC and outputs TMSC data with the falling edge of TCKC.

Minimizing the number of bits transferred per TAPC state machine state is affected by a number of factors. The information that must be transferred per TAPC state machine state varies for different test and debug applications. Also, with some test and debug

applications, all TAPC state machine states do not require that the same amount of information be transferred. This is especially true with debug applications where specialized use of the TAP is more likely.

67.8.5.3 Control Process Active (CPA) Function

While using the Advanced Protocol, all TAP.7 configuration changes occur within the CPA state. All online and offline transitions also occur while in the CPA state. When entering the CPA state from the BPA or SPA state, the CPA state checks for the use of supported features affecting the Advanced Protocol. The CPA state is exited when the enabled features affecting the Advanced Protocol are supported.

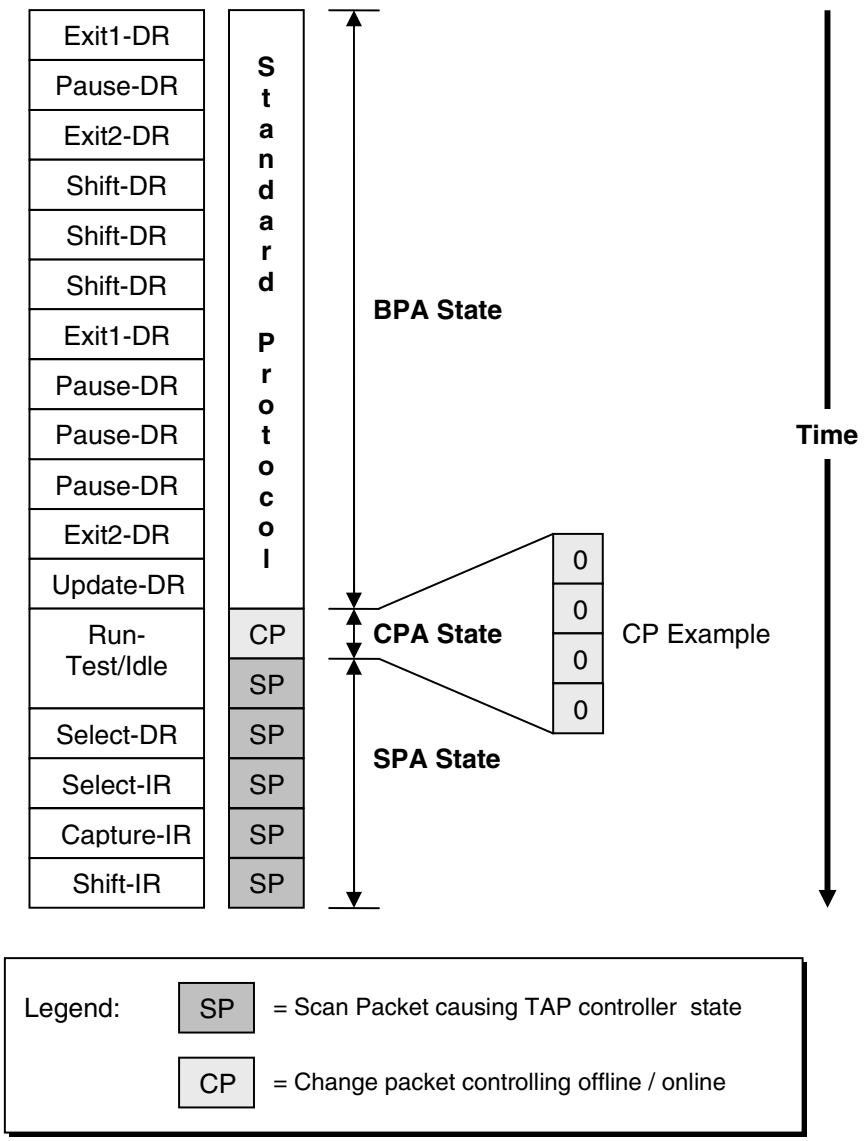


Figure 67-14. CPA State Entry from BPA State with Continued Online Operation

An SP preceding a CP does not advance the TAPC state machine state. This SP performs housekeeping functions that assure the contents of the SP preceding this SP have been fully utilized. This has significance when the SP contains control information that allows the STL to stall the completion of a SP.

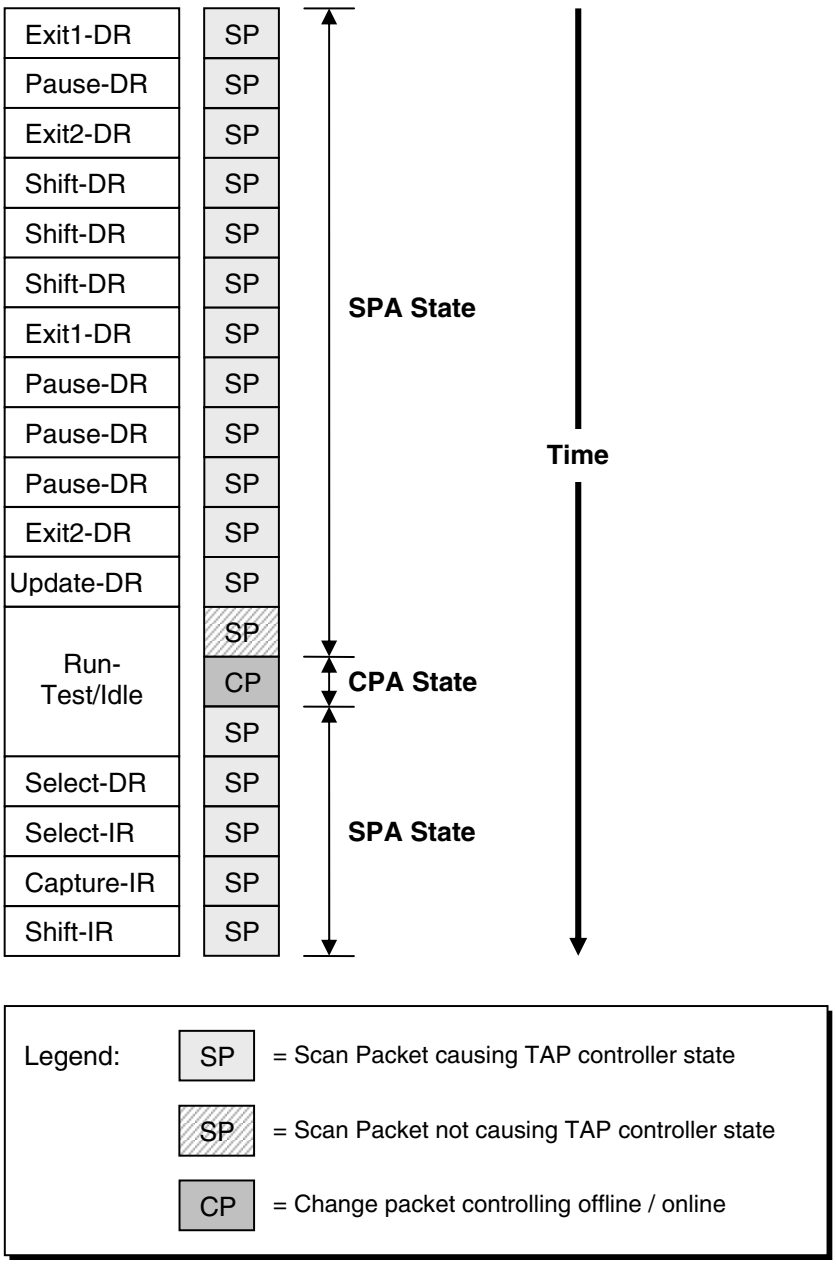


Figure 67-15. CPA State Entry from the SPA State with Continued Online Operation

When an unsupported feature is enabled, the TAP.7 controller is placed offline. This suspends the TAP.7 controller's operation at a point where its operation may later be synchronized to the remainder of the system. The APU stops advancing the EPU's and CLTAPC's TAPC state machine state while offline. It also suspends Data Channels activity while offline. The detection of the appropriate Online Activation Protocol while

the TAP.7 controller is offline places the TAP.7 controller online and causes an exit from the CPA state to the SPA state. An offline to online transition initiates the use of a predefined set of mandatory features in all online TAP.7 controllers to assure they operate synchronously.

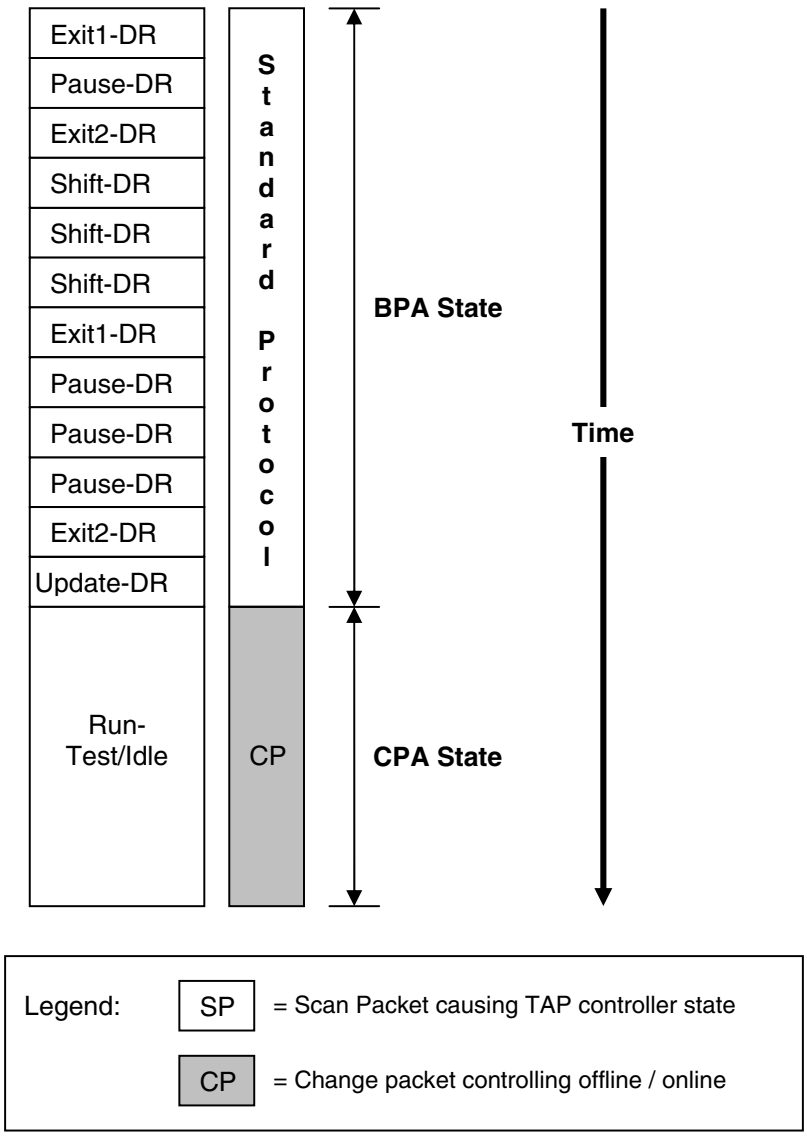


Figure 67-16. CPA State Entry from the BPA State with Offline Operation

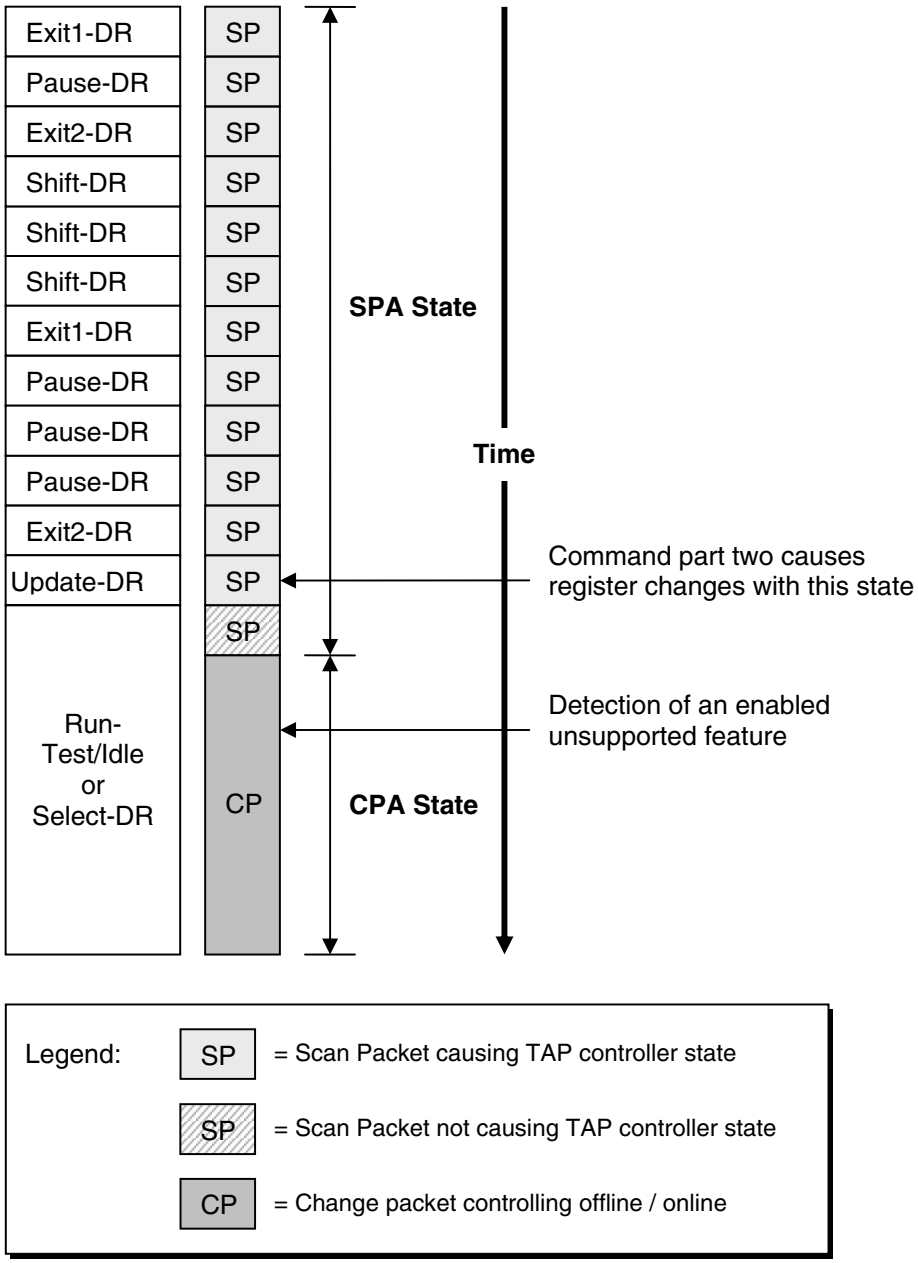


Figure 67-17. CPA State Entry from the SPA State with Offline Operation

A TAP.7 controller reset places the TAP.7 controller offline when the offline at start-up option is used. From this point the TAP.7 controller awaits being placed online.

67.8.6 Configuration Change Packets (CP)

A Change Packet (CP) is composed of three element types:

- **Preamble** – A 1-bit value that is a don't care. This bit facilitates a Standard to Advanced Protocol change with some external tool implementations.

- **Body** – Two or more bits representing one or more CP directives
  - **00** – CP_END
  - **01** – CP_NOP (extends the body by 1-bit)
  - **10** – CP_NOP (extends the body by 1-bit)
  - **11** – CP_RES (Generate a Type-3 TAP.7 controller reset coincident with the post amble bit.
- **Post amble** – A 1-bit value that is a don't care. This bit facilitates and Advanced to Standard Protocol change with some external tool implementations.

A minimum-length CP has a Preamble, a 2-bit Body, and a Post amble. This can be as simple as four logic zero values.

67.8.7 Scan Packet

A Scan Packet defines the period of TMSC pin activity where control information exchanged by the external tool and TAP.7 controller affects the TAPC state, except when the SP precedes a Change Packet. The external tool should view the Scan Packet as:

- Exchanging only the necessary TAPC information (TMS, TDI, and TDO or some subset thereof)
- Causing an advance of the TAP.7 TAPC state when it is not followed by a Change Packet.
- Causing an advance of the STL TAPC state when it is not followed by a Change Packet and when the STL is coupled.

With this perspective, the external tool views Scan Packets as running a virtual TAPC state machine at the TAP.7 pins. The effects of each Scan Packet take effect when the Scan Packet completes. The real TAP.7 and STL controller state machine states may change state before or after the virtual state machine.

67.8.8 SP Format

A SP serially exchanges the TDI, TMS, and TDO or a subset thereof, between the external tool and TAP.7 controller. Control information may be added to this exchange. A SP has three parts:

- **Payload** – Data + Control information (all scan formats), the payload content is varied to trade-off performance and flexibility. RDY bit(s) within the payload are included with some scan formats to provide the STL a means to stall progression of the TAPC state.
- **Delay** – Control information (optional) providing separation of adjacent payloads in time. Delay elements support simple external tool implementations where a delay element is used to stall progression of the TAPC state while it develops a new SP following the completion of the prior payload.

The header elements, delay elements, and control information within the payload element is consumed by the TAP.7 controller. This and other control information within the SP payload is not visible to the EPU and STL TAPCs.

67.8.8.1 Payload Element Content

67.8.8.1.1 Input and Output Bit Frames

An input bit frame is sourced by the external tool and an output bit frame is sourced by the TAP.7 controller. The payload may contain and input bit frame, an output bit frame, or both. When an input bet frame arrives at the TAP.7 controller, the TDI and TMS information within it is presented in parallel to the TAP.1 controllers within the EPU and STL. The output bit frame begins when the input frame is completed. The payload is completed when the EPU and STL are ready to utilize this information and the TDO data needed to complete the output bit frame is available. This process is repeated for each SP.

With input bit frames, MScan and OScan scan formats have no control bits input bit frames.

With output bit frames:

- TDO information passed to the external tool is a logic 1 when the EPU indicates the TDO pin would be high impedance.
- RDY bits provide TAP.7 stall opportunities with MScan and some OScan formats.
- PC0 and PC1 bits are added to MScan scan formats to allow Wire-ANDed output.

67.8.8.1.2 Varying the content of input and output bit frames

The various scan formats (MScan, OScan0-7) provide a number of options for the serialization of scan information for use in a Star-2 scan topology. These scan formats vary the content of SPs to balance flexibility and performance for different use cases. The amount of information transferred per TAPC state is based on the constraints imposed by these use cases. The MScan and OScan scan formats determine the factors affecting the payload content as shown in the following table.

Table 67-23. Factors determining the SP payload content

Scan format	Payload content determined by the:	
	TAPC state	Header
MScan	–	–
OScan	Yes	–

67.8.8.2 Delay Element Content

A delay element can be zero, one, two, or n bits in length. One and two bit delay elements are fixed length with neither the external tool or TAP.7 controller driving the TMSC pin. The TMSC pin is kept at the last value driven by either the external tool or TS. With an n-bit delay elements, the external tool drives the TMSC pin for the first n-1 bits and may drive the TMSC pin the nth bit. With an n-bit delay element the string of delay element bits is a series of directives virtually the same as those used by a CP (DLY_NOP, DLY_END, and DLY_RES).

67.8.8.3 Scan Formats

67.8.8.3.1 Overview

Scan formats utilizing the Advanced Protocol address different use cases. The amount of information transferred per TAPC state is based on the constraints imposed by these use cases. In one extreme two types of control information, the first allowing the stall of the completion of the Scan Packet, and the second supporting the Wire-AND of TAP.7 controller output are added to the TMS, TDI, and TDO information within the Scan Packet. In the other extreme, the external tool may minimize the information transferred for each TAPC state using its understanding of the application and transfer only the data of interest. In some cases the transfer of only one bit of information per TAPC is

required. There are a number of scan formats that operate between these two extremes. The external tool chooses a scan format that best matches the performance/functionality needs of the application.

The scan formats added to specify the use of the Advance Protocol are listed below:

- **MScan** – Maximum flexibility.
- **OScan0-7** – Optimized for test and debug applications.

These scan formats provide seven different functions. Five factors have influenced the definition of these scan formats:

- Supporting the capabilities of the IEEE1149.x and IEEE 1532 Standards.
- Providing the STL and external tool an opportunity to stall the TAPC state progression.
- Providing a minimum pin count and scalable functionality.
- Maximizing the efficiency of scan operations utilized for applications debug operations.
- Providing a rich set of capability for high end applications debug.

The performance and flexibility of MScan and OScan scan formats are shown in Table 67-24 and Table 67-26. The performance and flexibility of these scan formats are given subjective/approximate ratings in these tables. These ratings range from least to best. Use cases for these scan formats are also included in these tables.

67.8.8.3.2 MScan scan formats

The MScan scan format provides a universal scan function. It is the most flexible but lowest-performance scan format. The Scan Packets used by these scan formats are packed with data and control information and is the same for all TAPC states. This scan format accommodates the deployment of IEEE 1149.1 IP with non-compliant behavior within the STL. The STL is allowed to stall the TAPC state progression (to pace the arrival of Scan Packets with its ability to process them), and uses Wire-ANDed Output.

Table 67-24. MScan scan format use case summary

Scan format	Supporting	Performance	Flexibility	TCKC source
MScan	STL Stalls	Least	Best	External tool or TS
	Test			
	Non-compliant behavior STL IP			
	Multi-chip debug communication			
	Controller ID Allocation			

Table 67-25. MScan Scan Packet content

	Non-shift TAP controller states							Shift TAP controller states					
MScan	nTDI	TMS	PC0	RDY	PC1	TDO		nTDI	TMS	PC0	RDY	PC1	TDO

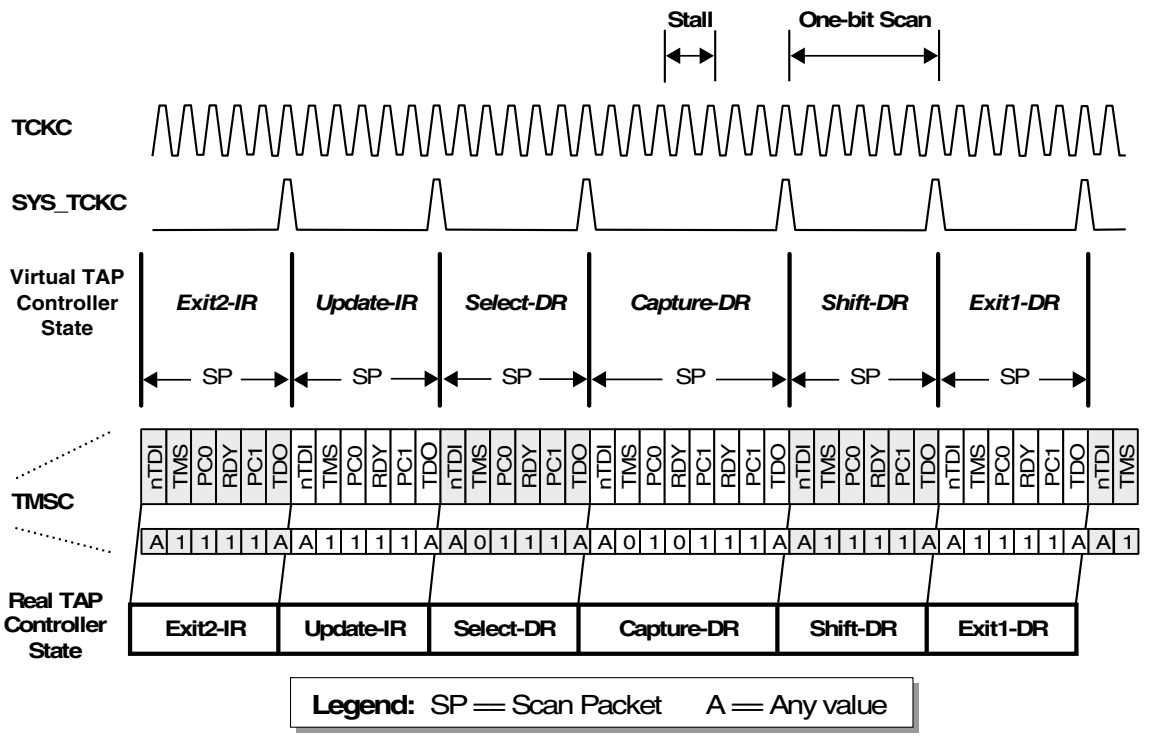


Figure 67-18. MScan Transaction

67.8.8.3.3 OScan Scan Formats

The OScan scan formats provide optimization choices where no knowledge of the data content of the scan exchange is needed. The Scan Packet content of these scan formats is modulated by a combination of the scan format and the TAPC state for some OScan scan formats. For instance, TDI data and TDO data is not exchanged for non-shift TAPC states and included in shift TAPC states in certain OScan formats. The modulation of Scan Packet content with TAPC state varies by scan format.

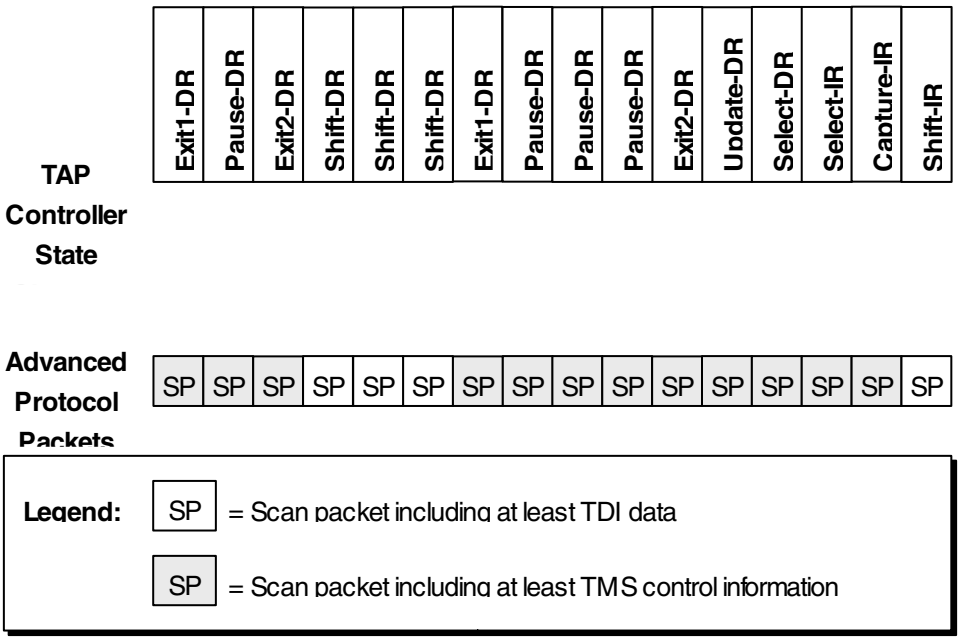


Figure 67-19. OScan Scan Packet content/TAPC state relationships

OScan scan formats replicate four functions in two forms:

- **OScan0 – OScan3:** Providing the best performance for a test clock source by the target system.
- **OScan 4 – OScan7:** Providing the best performance for a test clock sourced by the external tool.

The OScan4 – OScan7 scan formats are more efficient, as EOT escape sequences are used to identify an exit from the Shift-xR TAPC state. A Scan Packet associated with the Shift-xR TAPC state does not include a TMS bit. Instead, and EOT escape sequence overlaid on an nTDI bit causes an exit from shift TAPC state with no exit from this state occurring otherwise. A look at the 32-bit DR-Scan shows the impact of this optimization. A total of 96 bits are required when each Scan Packet associated with a Shift TAPC state when TMS is included in these Scan Packets while a total of 64 bits are required when TMS is not included in these Scan Packets. Assuming the EOT escape adds the equivalent of three bit periods, the number of TCK periods fo this transfer is 67/96 or roughly 70% of the total without this optimization.

These scan formats may only be used with an external tool sourced TCKC as EOTs are utilized to improve its performance. The OScan0 – OScan3 Scan Formats always have a TMS bit included in the Scan Packet. The OScan scan format use case summary is shown in the following table.

Table 67-26. OScan scan format use case summary

Scan Format	Supporting	Performance	Flexibility	TCKC Source
OScan0	Stalls	↓	Best	External tool
OScan1	Test		↑	
OScan2	Debug Performance			
OScan3	Debug Downloads (input only)	Best		
OScan4	Stalls	↓	Best	External tool or TS
OScan5	Test		↑	
OScan6	Debug Performance			
OScan7	Debug Downloads (input only)	Best		

Certain OScan optimizations are applied in a hierarchical manner.

- **Optimization I:** OScan1/OScan5 – RDY bit(s) and their trailing ONE bits are deleted from all Scan Packets.
- **Optimization II:** OScan2/OScan6 – Optimization I + nTDI and TDO bits are deleted from Scan Packets associated with non-shift TAPC states.
- **Optimization III:** OScan3/OScan7 – Optimization II + TDO bits are deleted from Scan Packets.
- **Optimization IV:** OScan4 – OScan7 – TMS bits are deleted from Scan Packets associated with shift TAPC states.

Table 67-27. OScan Scan Format "Payload"

Scan Format	Non-shift TAP controller states					Shift TAP controller states			
OScan0	nTDI	TMS	RDY	TDO		nTDI	TMS	RDY	TDO
OScan1	nTDI	TMS	–	TDO		nTDI	TMS	–	TDO
OScan2	–	TMS	–	–		nTDI	TMS	–	TDO
OScan3	–	TMS	–	–		nTDI	TMS	–	–
OScan4	nTDI	TMS	RDY	TDO		nTDI	–	RDY	TDO
OScan5	nTDI	TMS	–	TDO		nTDI	–	–	TDO
OScan6	–	TMS	–	–	Note 1	nTDI	TMS	–	TDO
					Note 2	nTDI	–	–	TDO
OScan7	–	TMS	–	–		nTDI	–	–	–

Note 1: Following Capture-xR or Exit2-xR Scan Packet

Note 2: Following Shift-xR Scan Packet

67.9 Functional Description

This section combines the information given in the Ancillary Services, EPU Operation, and APU Operation sections into a single reference for performing common TAP.7 tasks.

67.9.1 Switching from Standard Protocol to Advanced Protocol

Upon exit of Type-0 to Type-4 reset, the CJTAG begins operation in Standard Protocol 4-pin mode. The following steps are required to move from Standard Protocol 4-pin mode to Advanced Protocol (either 4-pin or 2-pin) mode.

The first step required to move to the Advanced Protocol is to change the CJTAG control level to control level 2 via zero-bit scans (ZBS), as described in Control levels.

After the control level is set tocontrol level 2, TAP.7 commands can be executed as described in EPU Commands. Execution of the Store Scan Format (STFMT) to set the scan format to any OSCAN0-7 or MSCAN format begins the switch from the Standard Protocol to the Advanced Protocol.

Once the STFMT command is executed selecting an Advanced Protocol scan format, the CJTAG transitions to the intermediate Control Protocol mode. The Control Protocol uses TMS inputs to control Configuration Change Packets as described in Control Process Active (CPA) Function and Configuration Change Packets (CP). Loading a change packet body value of 00 (CP_END) results in the switch to the Advanced Protocol.

If Advanced Protocol 2-pin operation is desired, the APFC register should be written to a value of 10 or 11 prior to the execution of the STFMT command that selects an advanced scan format. The APFC register is written via the Store Conditional 2-bit (STC2) command described in Store Conditional 2-bit (STC2).

Chapter 68
JTAG Data Communication (JDC)

68.1 Introduction

The JTAG Data Communication (JDC) module provides the capability to move register data between the IPS and JTAG domains. This facilitates communication between internal resources that access memory mapped register space and an external tool that accesses the JTAG port.

68.2 Overview

The JDC module consists of IPS accessible registers, JTAG accessible registers, and associated logic to coordinate movement of data from one register domain to another.

The JDC implements the following IPS data registers, occupying separate memory space:

- 32-bit memory mapped register that can be read or written via IPS (JOUT_IPS), and whose contents are ported out for capture into a JTAG register (JOUT) to be read via the JTAG port.
- 32-bit memory mapped register that can only be read via IPS (JIN_IPS), and whose contents are loaded from a JTAG register (JIN).

In addition to the data registers themselves, logic is implemented to indicate when new data has been shifted in via the JTAG port and is ready to be read from the JIN_IPS register, and when new data has been written to the JOUT_IPS register and is ready to be read via the JTAG port. The state of these flags is captured into a status register (MSR), and also provided to a JTAG register (JOUT) for tool visibility.

There is a single bus interface to port register data out to the JTAGC, and a single bus interface to port data in from the JTAGC. The architecture is shown in the following figure.

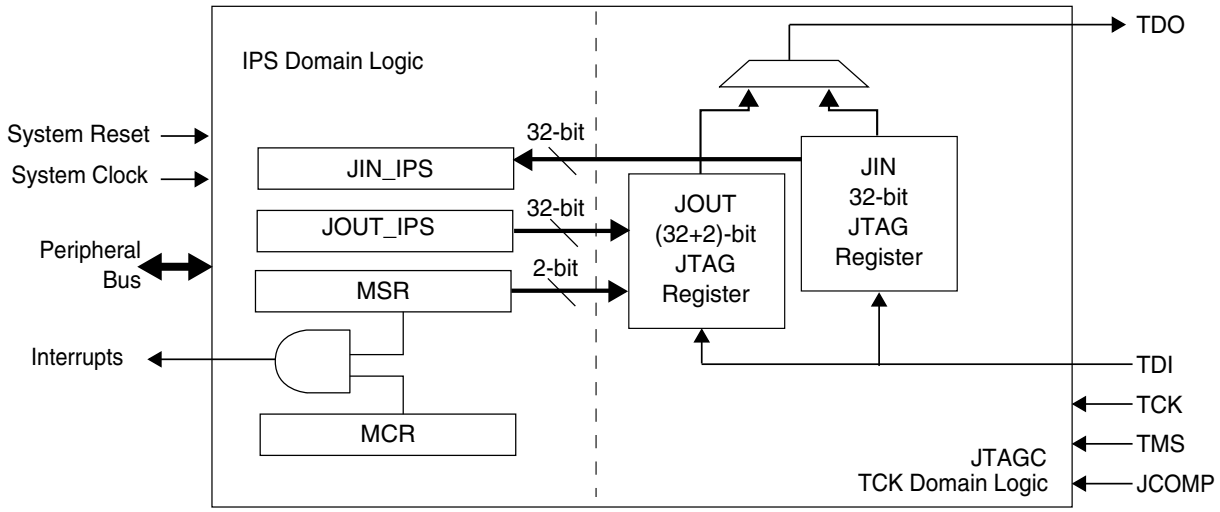


Figure 68-1. JDC block diagram

68.3 Memory mapped registers

Four 32-bit registers are implemented. Only 32-bit accesses are valid. The effects of access that are not 32 bits are not defined.

JDC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (JDC_MCR)	32	R/W	0000_0000h	68.3.1/2957
4	Module Status Register (JDC_MSR)	32	R/W	0000_0000h	68.3.2/2958
8	JTAG Output Data Register (JDC_JOUT_IPS)	32	R/W	0000_0000h	68.3.3/2959
C	JTAG Input Data Register (JDC_JIN_IPS)	32	R	0000_0000h	68.3.4/2960

68.3.1 Module Configuration Register (JDC_MCR)

The MCR is used to enable the interrupt outputs of the JDC. This register is reset by system destructive reset.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															JIN_IEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved															JOUT_IEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

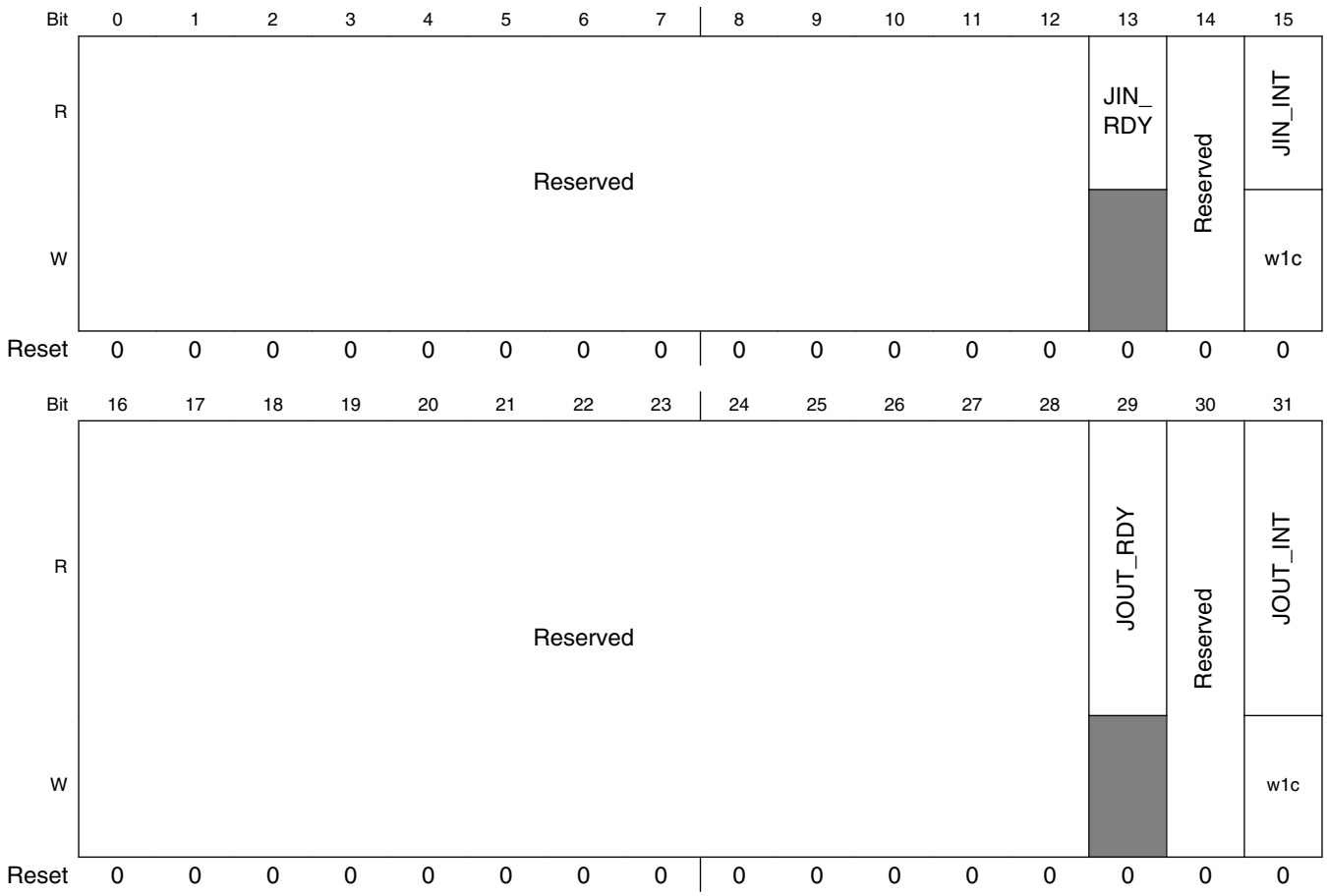
JDC_MCR field descriptions

Field	Description
0–14 Reserved	This field is reserved.
15 JIN_IEN	JIN Interrupt Enable. 0 Setting of MSR[JIN_INT] bit does not assert the JIN interrupt 1 Setting of MSR[JIN_INT] bit asserts the JIN interrupt
16–30 Reserved	This field is reserved.
31 JOUT_IEN	JOUT Interrupt Enable. 0 Setting of MSR[JOUT_INT] bit does not assert the JOUT interrupt 1 Setting of MSR[JOUT_INT] bit asserts the JOUT interrupt

68.3.2 Module Status Register (JDC_MSR)

The MSR holds the JTAG register status and interrupt bits. This register is reset by system destructive reset.

Address: 0h base + 4h offset = 4h



JDC_MSR field descriptions

Field	Description
0–12 Reserved	This field is reserved.
13 JIN_RDY	JIN Ready (read only). 0 Cleared upon software read of JIN_IPS contents via IPS 1 Set when new data is written to the JIN_IPS register
14 Reserved	This field is reserved.
15 JIN_INT	JIN Interrupt.

Table continues on the next page...

JDC_MSR field descriptions (continued)

Field	Description
	0 Cleared by writing logic 1 1 Set when new data is written to the JIN_IPS register
16–28 Reserved	This field is reserved.
29 JOUT_RDY	JOUT Ready (read only). 0 Cleared upon tool read of JOUT register via JTAG port 1 Set when new data is written to the JOUT_IPS register
30 Reserved	This field is reserved.
31 JOUT_INT	JOUT Interrupt. 0 Cleared by writing logic 1 1 Set when JOUT_RDY bit is cleared by tool reading JOUT register

68.3.3 JTAG Output Data Register (JDC_JOUT_IPS)

The JOUT_IPS register holds data written via IPS. The JOUT_IPS contents are ported out and captured into the JOUT register to be read via the JTAG port. This register is reset by system destructive reset.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Data																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

JDC_JOUT_IPS field descriptions

Field	Description
0–31 Data	JOUT_IPS data.

68.3.4 JTAG Input Data Register (JDC_JIN_IPS)

Data written to the JTAG input data register (JIN) via the JTAG port is held in the JIN_IPS register, where it can be read via IPS. This register is reset by system destructive reset.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Data																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

JDC_JIN_IPS field descriptions

Field	Description
0–31 Data	JIN_IPS data.

68.4 Non-Memory mapped register definition

The JDC also implements two JTAG accessible registers that are not memory mapped. One JTAG register is used to shift in data to be placed in the JIN_IPS register. The other JTAG register captures data from the JOUT_IPS register and ready status from the MSR to be shifted out via the JTAG port.

68.4.1 JTAG output data register (JOUT)

The JOUT register captures data from the JOUT_IPS register upon execution of the JOUT_READ JTAG instruction. It also holds the JIN_RDY and JOUT_RDY status bits. This register is reset by system destructive reset and JTAG reset. The following figure shows the format of the JOUT register.

	33	…	2	1	0
R W	JOUT_IPS Data			JIN_RDY	JOUT_RDY
Reset	*[1]	*	*	*	*

1. The reset value of the JOUT register is 34'b0.

The JOUT register is described in the following table.

Table 68-6. JOUT JTAG register field descriptions

Field	Description
JOUT_IPS Data	Data value from JOUT_IPS register
JIN_RDY	State of JIN_RDY bit from MSR
JOUT_RDY	State of JOUT_RDY bit from MSR

68.4.2 JTAG input data register (JIN)

Data is written to the JIN register via JTAG during execution of the JIN_WRITE JTAG instruction. The JIN data is later captured in the JIN_IPS register to be read via IPS. This register is reset by system destructive reset and JTAG reset. The following figure shows the format of the JIN register.

	31	…	0
R W	JIN Data		
Reset	*[1]	*	*

1. The reset value of the JIN register is 32'b0.

The JIN register is described in the following table.

Table 68-7. JIN JTAG register field descriptions

Field	Description
JIN Data	Contains data to be captured in JIN_IPS register upon exit of Update-DR state when executing WRITE_JIN JTAG instruction.

68.5 Functional description

The JDC module provides the ability to shift in data via the JTAG port and capture that data into a memory mapped register that can be accessed via IPS. It also provides the ability to capture data written to a memory mapped register into a JTAG shift register for output via the JTAG port. An overview of the module functionality is described below.

- A software write to the JOUT_IPS register sets the JOUT_RDY flag bit, indicating new data is available to be read from the JOUT register via the JTAG port. The state of the JOUT_RDY bit is reflected in the MSR and also ported out to the JOUT register. A JTAG read of the JOUT register via execution of the JOUT_READ

instruction with a JOUT_RDY bit whose value is logic 1 indicates the register contains new data. The JOUT_RDY flag bit is cleared upon exit of the Capture-DR JTAG state during execution of the JOUT_READ instruction. Clearing the JOUT_RDY bit indicates to software that a new data value can be written to the JOUT_IPS register.

- A JTAG write to the JIN register via execution of the JIN_READ JTAG instruction updates the contents of the JIN_IPS register upon exit of the Update-DR state. An update of the JIN_IPS register sets the JIN_RDY flag bit, indicating new data is available to be read via IPS. The state of the JIN_RDY bit is reflected in the MSR register and also ported out to the JOUT register. The JIN_RDY flag bit is cleared upon software read of the JIN_IPS register. A JTAG read of the JOUT register with a JIN_RDY value of logic 0 indicates new data can be written to the JIN register.

68.5.1 JTAG register access

Refer to the JTAGC documentation for information on how to access the JTAG registers.

68.5.1.1 JDC block instructions

The JDC block implements the IEEE 1149.1-2001 defined instructions listed in the following table. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 68-8. JTAG instructions

Instruction	Code[4:0]	Instruction Summary
Reserved	00001	Factory debug reserved
JOUT_READ	00010	Selects JOUT data register. Data from JOUT_IPS is captured into JOUT data register upon entry to Capture-DR state while JOUT_READ is active.
JIN_WRITE	01110	Selects JIN data register. Data from JIN is captured into JIN_IPS upon exit of Update-DR state while JIN_WRITE is active.
BYPASS	11111	Selects bypass register for data operations
Reserved	All other opcodes	Decoded to select bypass register

Chapter 69
Sequence Processing Unit (SPU)

69.1 Sequence Processing Unit (SPU) chip-specific module information

Although SPU_SS[25] is reserved, it will be automatically asserted when SPU is enabled (SPU_SE[SED]=1).

69.2 Introduction

The Sequence Processing Unit (SPU) provides an on-device trigger functions similar to those found on a logic analyzer. In a complex SoC, performance monitor functions are available at two levels: system level and CPU level. Complex trigger and system performance monitor functions are implemented in the SPU module. System level performance monitor functions are integrated into the SPU complex trigger logic, and thus allow counting and timing of any debug trigger combinations supported by the SPU.

Various clients such as the cores or crossbar switch, generate watchpoints and triggers when operating in their debug mode. The SPU collects these triggers (such as interrupt occurrence, address watchpoint, and so on) and uses them as conditions to sequence through states, with resultant actions (such as start/stop trace, start/stop counter, capture timebase, and so on) The SPU provides the capability to create complex debug triggers. By configuring each of the events to trigger specific actions, complex logic-analyzer-like behavior can be achieved, providing vital real-time visibility and debug ability of the activities of the system.

69.2.1 SPU block diagram

The following figure shows a top-level block diagram of the SPU.

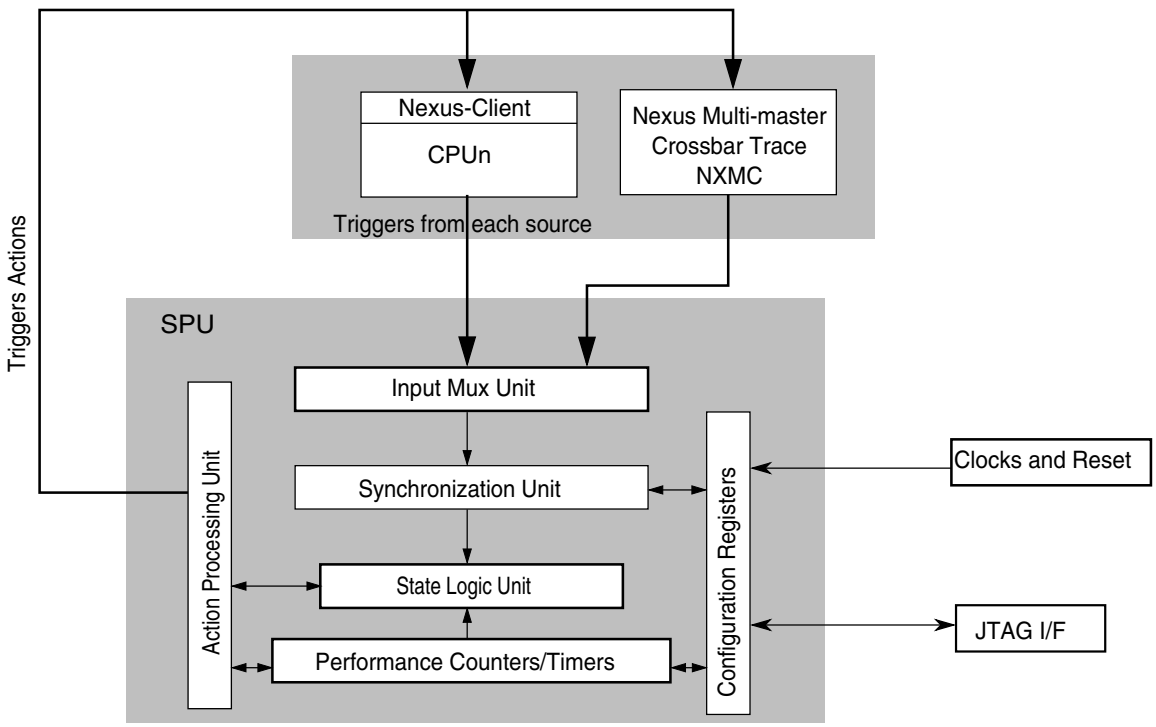


Figure 69-1. SPU block diagram

69.2.2 Overview

The SPU is able to generate complex debug events, based upon input events from sources (CPUs and system bus clients) throughout the SoC. The SPU has the ability to create a state machine to trigger a debug action. Single or multiple actions can be triggered by a state machine, which results in debug events being created. There are a selection of timers and counters available to assist comparison events.

Complex trigger support is implemented in the State Logic Unit (SLU). There are numerous inputs coming from different clients. Users can choose 63 inputs from a set of 167 inputs at the level one Mux. There are actually 64 inputs to the Muxes but one of the inputs is permanently connected to a high level, so users can only choose 63 of the inputs. Out of these 63 inputs, users can select 16 inputs for each SLU. This is shown in Figure 69-4.

One SLU consists of four input AND gates, the output of which are ORed together. One of the inputs of each AND gate has the option to invert it. Also, the output of each AND gate can be optionally inverted. Finally, the output of each SLU has the option of inversion. The block diagram for one SLU state is shown in Figure 69-5. The SPU includes eight SLUs states.

Users can utilize these states to form a configurable state machine. This allows users to join states together with the if-then-else operations to create a sequence. Each state in a sequence has the ability to move to another state based on the true condition from the state logic, and the ability to route to another state based on the a false condition from the state logic. This is shown in Figure 69-6 and Figure 69-7. At a given time, a maximum of four sequences are allowed. Each sequence can have one state or maximum of eight states. Each state can only be used in one unique sequence. The number of states in all the sequences cannot exceed eight. Each state has the ability to trigger one to four actions based on a true/false condition in the active sequence.

An example of typical complex trigger for the SPU would be: if FunctionA is entered, monitor CPU0 writes to VariableB. If CPU0 writes a value to VariableB which is 0x100, then enable trace for CPU1 and halt CPU0.

The SPU consists of the following sub-blocks:

- Input Mux Unit—receives the numerous triggers from various clients and reduces them down into a user selected set. The muxing unit selects sixteen inputs for each sequencing SLU from a set of 63 static inputs. This selection is done using two levels of muxing. At the first level of muxing, users select 63 out of 167 inputs. At the second level of muxing, out of these 63 inputs, one input can be selected for each of the four input AND gates. There are 128 64 × 1 Muxes to cater the needs of 128 inputs of all the eight states. This input mux selection logic is shown in Figure 69-4.
- State Logic Units (SLU)—responsible for performing operations on the selected input events to produce desired events. Each SLU has a control register that manages how the various inputs are combined to produce a desired event. Only one input of each four input AND gate can be independently inverted. The combined event can also be optionally inverted. These optional inversions coupled with DeMorgan's law allows for generalized signal combinations. This is shown in Figure 69-5.
- Action Processing Unit—takes an action request from a SLU state and converts this into one or multiple actions. The action unit can be programmed to trigger various actions in response to each processed event from the SLU unit. The user can define the actions associated with each state from each SLU.
- Counters/Timers—The SPU implements 16 32-bit timer/counter functionality. Timer or counter selection is done via a configurable register in the SPU. Each counter/timer has its own control register. A 32-bit comparator value can optionally be compared against the timer/counter. These registers are accessible via JTAG.

69.2.3 Features

The features of the SPU are as follows:

- Clients:
  - 2 CPU(s)
  - 2 System Bus clients (NXMC)
- Input watchpoints/triggers selection
- Input debug triggers for instruction execution selection
- Input interrupt selection
- Sequences and states criteria
  - Up to 4 user programmable number of active/simultaneous sequences
  - Up to 8 user programmable states that can be split between all four sequences
  - Each state can only be used in one unique sequence
  - A sequence can consist of a single state or any number of states up to 8
- State logic
  - Each state logic consist of sum of products (SOP)
  - SOP consist of four 4-input AND gates, and one 4-input OR gate
  - Optional inversion on one input of each AND gate of each state logic
  - Optional inversion on output of each AND gate.
  - Optional inversion on output of OR gate of each state
  - States are joined together with if-then-else statement to create a sequence
- SLU events
  - Up to 8 user programmable derived events/actions
  - Any state can optionally trigger of 1 to 4 actions based on a true condition
  - Any state can optionally trigger of 1 to 4 actions based on a false condition
  - User selection of trigger actions generated in response to each programmed trigger
- SPU counters/timers

    - 16 32-bit counters/timers
    - Timer/counter configurability selection via a register
    - Start/increment when used as timer/counter respectively
    - Reset when used as timer/counter respectively
    - Counter value compare match
    - Counter overflow event status
    - Generation of a greater than condition against a 32-bit comparator value (no support for range condition from single counter)
    - Range support using two counters
    - Use of counter/timer registers to generate counted or timed triggers
    - Use of counter/timer registers to capture system performance statistics
    - Each timer/counter value has the ability to be inserted into the Nexus trace stream upon an SPU action
    - Access to timer/counter values via JTAG
- Prescaler for timer mode only
    - Configurable prescaler of /1, /4, /16 or /32 of the clock for timer operation inside the SPU
- Asynchronous interfaces
    - SPU to/from CPU (fastest clock)
    - SPU to/from NXMC
    - SPU to/from NAR
    - SPU to/from DCI
    - SPU to/from JTAGM
- Interfaces
    - CPU
    - System bus
    - NAR
    - DCI

- Configuration registers—32-bit double word aligned configuration registers (byte, halfword access not supported)
- SPU actions—SPU block triggers various actions based on the derived events, these actions are defined in SPU actions.

69.3 SPU actions

Sequence actions describe what is triggered. A sequence state may only have a GoTo action to determine the next sequence state. The complete set of actions are as follows:

- Start/stop trace for a source
- Start/stop/increment counter/timer counter/timer[0–15]
- Reset counter/timer[0–15]
- Capture counter/timer values[0–15]—place the specified counter/timer value into the trace stream for counters[0–15]
- Halt1—single signal (EVTI0) that can be used to halt a pre-configured DCI group1
- Halt2—single signal (EVTI1) that can be used to halt a pre-configured DCI group2
- Generate a watchpoint trigger for a respective ID field the NXMC (global timestamp must be enabled from the NAR)
- Capture the global timebase and place it into the trace stream with a watchpoint ID field of respective watchpoints at the NXMC (send TIMESTAMPx where x is related to an watchpoint ID field of NXMC only)
- Single interrupt for INTC to interrupt a CPU—de-assert (cleared) via JTAGM
- Generate a pulse on the EVTO pin—output pulse over EVTO1 and EVTO2
- Start or Stop Performance Counters (PMC) for a CPU source—Start/Stop CPUn
- Optional trace group 1 (Start/Stop Trace a pre-configured set of trace signals)—four (DTM,PTM,WTM,OTM) independent start/stop trace groups can be formed. One or more clients (CPUn and Nexus) can be part of a particular group.

Note

GoTo is not part of the Action list described above but it is implemented in the state logic to move from one state to another. If there is nothing defined for GoTo, it signifies the end of the sequence.

In the condition where a state (State n) updates a counter and the next state in the sequence (State n+1) evaluates the value of the same counter (for example, to compare if the counter is less-than a certain value), the update to the counter (from State n) is not applied before the counter evaluation in the next state (State n+1). This results in the old value of the counter being evaluated (in State n+1), rather than the desired updated value. This is due to hardware limitations of the SPU as the update to the counter does not complete until the clock cycle after the evaluation of the next state. This limitation is only applicable in Div1 mode and not in Div4 mode.

To workaround this hardware limitation, a delay should be inserted between the update to a counter and the evaluation of this counter. A dummy state can be used between the counter update (State n) and the counter evaluation (State n+1). A dummy state is a state with any input, that on both then/else conditions, the action proceeds to the next state (State n+1) where the counter evaluation occurs. The insertion of this dummy state provides sufficient time for the update of the counter to be applied, in time for the counter evaluation. The same outcome is also possible in a complex sequence with multiple states by scheduling states so that counter update and counter evaluation are not performed on consecutive states in a sequence.

69.4 SPU enable/disable mode

By default, the SPU is in disabled mode. In this state, all events/inputs are disabled such that no counting or other actions occur. All of the SPU control registers are accessible in this state. All of the configurations must be done before enabling the SPU by programming the SPU Enable/Disable register.

69.5 SPU module interface

The SPU is connected to many clients such as the cores, system bus, NAR and Nexus clients. The SPU also interfaces to the DCI and the JTAG modules as shown in the following figure.

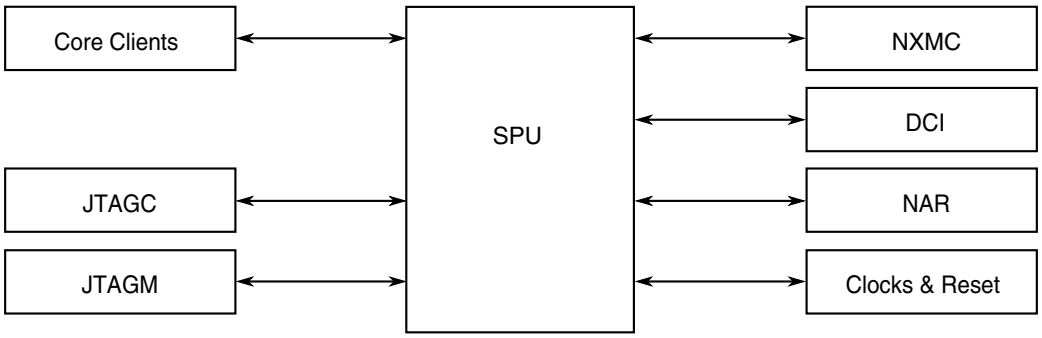


Figure 69-2. SPU module interface

69.6 Register definition

The SPU contains 32-bit aligned registers (byte and half-word access are not supported). All configuration registers in the SPU are JTAG registers such that these registers are accessible via JTAG interface only. Write access to reserved locations have no impact while read access to reserved locations always return 0. These registers have Nexus reset and JTAG reset control.

Note

The SPU does not check for correctness of the programmed values in the registers and programmers must ensure that correct values are being written.

The following table lists the memory mapped registers which control the operation of the SPU. There are two instructions for the SPU registers:

- Instruction to enable the SLU (first set of registers with offsets 0x1–0x49) registers is 4'h3
- Instruction to enable the Counter Control Unit (second set of registers with offsets 0x1–0x48) is 4'h2

Table 69-1. SPU register summary

Offset (hex)	Register	Access	Reset Value
1	Level1 Mux Selection 0 (L1SEL0)	R/W	32'h0
2	Level1 Mux Selection 1 (L1SEL1)	R/W	32'h0
3	Level1 Mux Selection 2 (L1SEL2)	R/W	32'h0
4	Level1 Mux Selection 3 (L1SEL3)	R/W	32'h0
5	Level1 Mux Selection 4(L1SEL4)	R/W	32'h0
6	Level1 Mux Selection 5 (L1SEL5)	R/W	32'h0

Table continues on the next page...

Table 69-1. SPU register summary (continued)

Offset (hex)	Register	Access	Reset Value
7	Level1 Mux Selection 6 (L1SEL6)	R/W	32'h0
8	Level1 Mux Selection 7 (L1SEL7)	R/W	32'h0
9	Reserved for future use	—	—
A	Reserved for future use	—	—
B	Input Trigger Level Detection 2 (ITLD2)	R/W	32'h0
C	Input Trigger Level Detection 3 (ITLD3)	R/W	32'h0
D	CPU0 Processor Except Vector Prefix (C0PEVP)	R/W	32'h0
E	CPU1 Processor Except Vector Prefix (C1PEVP)	R/W	32'h0
F	Reserved	—	—
10	Reserved for future use	—	—
11	Reserved for future use	—	—
12	CPU0 Interrupt Priority Selection (C0PIS)	R/W	32'h0
13	CPU1 Interrupt Priority Selection (C1PIS)	R/W	32'h0
14	Reserved	—	—
15	Reserved for future use	—	—
16	Reserved for future use	—	—
17	Level2 Mux State0 Selection 0 (L20SEL0)	R/W	32'h0
18	Level2 Mux State0 Selection 1 (L20SEL1)	R/W	32'h0
19	Level2 Mux State0 Selection 2 (L20SEL2)	R/W	32'h0
1A	Level2 Mux State0 Selection 3 (L20SEL3)	R/W	32'h0
1B	Level2 Mux State1 Selection 0 (L21SEL0)	R/W	32'h0
1C	Level2 Mux State1 Selection 1 (L21SEL1)	R/W	32'h0
1D	Level2 Mux State1 Selection 2 (L21SEL2)	R/W	32'h0
1E	Level2 Mux State1 Selection 3 (L21SEL3)	R/W	32'h0
1F	Level2 Mux State2 Selection 0 (L22SEL0)	R/W	32'h0
20	Level2 Mux State2 Selection 1 (L22SEL1)	R/W	32'h0
21	Level2 Mux State2 Selection 2 (L22SEL2)	R/W	32'h0
22	Level2 Mux State2 Selection 3 (L22SEL3)	R/W	32'h0
23	Level2 Mux State3 Selection 0 (L23SEL0)	R/W	32'h0
24	Level2 Mux State3 Selection 1 (L23SEL1)	R/W	32'h0
25	Level2 Mux State3 Selection 2 (L23SEL2)	R/W	32'h0
26	Level2 Mux State3 Selection 3 (L23SEL3)	R/W	32'h0
27	Level2 Mux State4 Selection 0 (L24SEL0)	R/W	32'h0
28	Level2 Mux State4 Selection 1 (L24SEL1)	R/W	32'h0
29	Level2 Mux State4 Selection 2 (L24SEL2)	R/W	32'h0
2A	Level2 Mux State4 Selection 3 (L24SEL3)	R/W	32'h0
2B	Level2 Mux State5 Selection 0 (L25SEL0)	R/W	32'h0
2C	Level2 Mux State5 Selection 1 (L25SEL1)	R/W	32'h0
2D	Level2 Mux State5 Selection 2 (L25SEL2)	R/W	32'h0

Table continues on the next page...

Table 69-1. SPU register summary (continued)

Offset (hex)	Register	Access	Reset Value
2E	Level2 Mux State6 Selection 3 (L25SEL3)	R/W	32'h0
2F	Level2 Mux State6 Selection 0 (L26SEL0)	R/W	32'h0
30	Level2 Mux State6 Selection 1 (L26SEL1)	R/W	32'h0
31	Level2 Mux State6 Selection 2 (L26SEL2)	R/W	32'h0
32	Level2 Mux State6 Selection 3 (L26SEL3)	R/W	32'h0
33	Level2 Mux State7 Selection 0 (L27SEL0)	R/W	32'h0
34	Level2 Mux State7 Selection 1 (L27SEL1)	R/W	32'h0
35	Level2 Mux State7 Selection 2 (L27SEL2)	R/W	32'h0
36	Level2 Mux State7 Selection 3 (L27SEL3)	R/W	32'h0
37	Input/Output Inversion Control State0 (IOIC0)	R/W	32'h0
38	Input/Output Inversion Control State1 (IOIC1)	R/W	32'h0
39	Input/Output Inversion Control State2 (IOIC2)	R/W	32'h0
3A	Input/Output Inversion Control State3 (IOIC3)	R/W	32'h0
3B	Input/Output Inversion Control State4 (IOIC4)	R/W	32'h0
3C	Input/Output Inversion Control State5 (IOIC5)	R/W	32'h0
3D	Input/Output Inversion Control State6 (IOIC6)	R/W	32'h0
3E	Input/Output Inversion Control State7 (IOIC7)	R/W	32'h0
3F	Sequence Control (SCTRL)	R/W	32'h0
40	State0 Goto Control (S0GC)	R/W	32'h0
41	State1 Goto Control (S1GC)	R/W	32'h0
42	State2 Goto Control (S2GC)	R/W	32'h0
43	State3 Goto Control (S3GC)	R/W	32'h0
44	State4 Goto Control (S4GC)	R/W	32'h0
45	State5 Goto Control (S5GC)	R/W	32'h0
46	State6 Goto Control (S6GC)	R/W	32'h0
47	State7 Goto Control (S7GC)	R/W	32'h0
48	SLU Status (SS)	R/W	32'h0
49	SPU Enable (SE)	R/W	32'h0
1	State0 True Action 0 (S0TA0)	R/W	32'h0
2	State0 True Action 1 (S0TA1)	R/W	32'h0
3	State1 True Action 0 (S1TA0)	R/W	32'h0
4	State1 True Action 1 (S1TA1)	R/W	32'h0
5	State2 True Action 0 (S2TA0)	R/W	32'h0
6	State2 True Action 1 (S2TA1)	R/W	32'h0
7	State3 True Action 0 (S3TA0)	R/W	32'h0
8	State3 True Action 1 (S3TA1)	R/W	32'h0
9	State4 True Action 0 (S4TA0)	R/W	32'h0
A	State4 True Action 1 (S4TA1)	R/W	32'h0
B	State5 True Action 0 (S5TA0)	R/W	32'h0

Table continues on the next page...

Table 69-1. SPU register summary (continued)

Offset (hex)	Register	Access	Reset Value
C	State5 True Action 1(S5TA1)	R/W	32'h0
D	State6 True Action 0 (S6TA0)	R/W	32'h0
E	State6 True Action 1(S6TA1)	R/W	32'h0
F	State7 True Action 0 (S7TA0)	R/W	32'h0
10	State7 True Action 1(S7TA1)	R/W	32'h0
11	State0 False Action 0 (S0FA0)	R/W	32'h0
12	State0 False Action 1 (S0FA1)	R/W	32'h0
13	State1 False Action 0 (S1FA0)	R/W	32'h0
14	State1 False Action 1(S1FA1)	R/W	32'h0
15	State2 False Action 0 (S2FA0)	R/W	32'h0
16	State2 False Action 1(S2FA1)	R/W	32'h0
17	State3 False Action 0 (S3FA0)	R/W	32'h0
18	State3 False Action 1(S3FA0)	R/W	32'h0
19	State4 False Action 0 (S4FA0)	R/W	32'h0
1A	State4 False Action 1 (S4FA1)	R/W	32'h0
1B	State5 False Action 0 (S5FA0)	R/W	32'h0
1C	State5 False Action 1 (S5FA1)	R/W	32'h0
1D	State6 False Action 0 (S6FA0)	R/W	32'h0
1E	State6 False Action 1 (S6FA1)	R/W	32'h0
1F	State7 False Action 0 (S7FA0)	R/W	32'h0
20	State7 False Action 1 (S7FA1)	R/W	32'h0
21	DTM Trace Group Configuration (DTMTGC)	R/W	32'h0
22	PTM Trace Group Configuration (PTMTGC)	R/W	32'h0
23	OTM Trace Group Configuration (OTMTGC)	R/W	32'h0
24	WTM Trace Group Configuration (WTMTGC)	R/W	32'h0
25	Interrupt Status (INTS)	R/W	32'h0
26	Counter Control 0 (CCTRL0)	R/W	32'h0
27	Counter Control 1 (CCTRL1)	R/W	32'h0
28	Counter Control 2 (CCTRL2)	R/W	32'h0
29	Counter Control 3 (CCTRL3)	R/W	32'h0
2A	Counter Control 4 (CCTRL4)	R/W	32'h0
2B	Counter Control 5 (CCTRL5)	R/W	32'h0
2C	Counter Control 6 (CCTRL6)	R/W	32'h0
2D	Counter Control 7 (CCTRL7)	R/W	32'h0
2E	Counter Control 8 (CCTRL8)	R/W	32'h0
2F	Counter Control 9 (CCTRL9)	R/W	32'h0
30	Counter Control 10 (CCTRL10)	R/W	32'h0
31	Counter Control 11 (CCTRL11)	R/W	32'h0
32	Counter Control 12 (CCTRL12)	R/W	32'h0

Table continues on the next page...

Table 69-1. SPU register summary (continued)

Offset (hex)	Register	Access	Reset Value
33	Counter Control 13 (CCTRL13)	R/W	32'h0
34	Counter Control 14 (CCTRL14)	R/W	32'h0
35	Counter Control 15 (CCTRL15)	R/W	32'h0
36	Counter Compare 0 (CCMP0)	R/W	32'h0
37	Counter Compare 1 (CCMP1)	R/W	32'h0
38	Counter Compare 2 (CCMP2)	R/W	32'h0
39	Counter Compare 3 (CCMP3)	R/W	32'h0
3A	Counter Compare 4 (CCMP4)	R/W	32'h0
3B	Counter Compare 5 (CCMP5)	R/W	32'h0
3C	Counter Compare 6 (CCMP6)	R/W	32'h0
3D	Counter Compare 7 (CCMP7)	R/W	32'h0
3E	Counter Compare 8 (CCMP8)	R/W	32'h0
3F	Counter Compare 9 (CCMP9)	R/W	32'h0
40	Counter Compare 10 (CCMP10)	R/W	32'h0
41	Counter Compare 11 (CCMP11)	R/W	32'h0
42	Counter Compare 12 (CCMP12)	R/W	32'h0
43	Counter Compare 13 (CCMP13)	R/W	32'h0
44	Counter Compare 14 (CCMP14)	R/W	32'h0
45	Counter Compare 15 (CCMP15)	R/W	32'h0
46	Counter Compare Status (CCOMS)	R/W	32'h0
47	Counter Overflow Status (COS)	R/W	32'h0
48	Counter Capture Status (CCAPS)	R/W	32'h0

69.6.1 Level1 input Mux configurations registers

A set of input Mux control registers is used to narrow down the large number of SPU watchpoints to a manageable set. There are sixty-four 8 × 1 (eight inputs and one output) multiplexers. Three bits per Mux are used to select one input out of 8 inputs. 8 Muxes are configured with each 32-bit register; in other words, 8 out of 64 inputs for each register. This provides the flexibility to the programmer to route any input trigger to any state input. The detail of this selection is shown in Figure 69-4.

Note

Users cannot program MUX 63 in L1SEL7 (the 64th mux). All eight inputs of the 63rd mux are tied to high and the output is always high (1'b1).

69.6.1.1 Level1 Mux Selection 0 (L1SEL0)

The following table shows the format of the L1SEL0 register. The values for the fields of the L1SEL0 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured.

Offset: 0x01 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	MUX 7			0	MUX 6			0	MUX 5			0	MUX 4			0	MUX 3			0	MUX 2			0	MUX 1			0	MUX 0		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

69.6.1.2 Level1 Mux Selection 1 (L1SEL1)

The following table shows the format of the L1SEL1 register. The values for the fields of the L1SEL1 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured.

Offset: 0x02 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	MUX 15			0	MUX 14			0	MUX 13			0	MUX 12			0	MUX 11			0	MUX 10			0	MUX 9			0	MUX 8		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

69.6.1.3 Level1 Mux Selection 2 (L1SEL2)

The following table shows the format of the L1SEL2 register. The values for the fields of the L1SEL2 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured.

Offset: 0x03 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	MUX 23			0	MUX 22			0	MUX 21			0	MUX 20			0	MUX 19			0	MUX 18			0	MUX 17			0	MUX 16		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

69.6.1.4 Level1 Mux Selection 3 (L1SEL3)

The following table shows the format of the L1SEL3 register. The values for the fields of the L1SEL3 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured.

Offset: 0x04 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	MUX 31			0	MUX 30			0	MUX 29			0	MUX 28			0	MUX 27			0	MUX 26			0	MUX 25			0	MUX 24		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

69.6.1.5 Level1 Mux Selection 4 (L1SEL4)

The following table shows the format of the L1SEL4 register.The values for the fields of the L1SEL4 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured.

Offset: 0x05 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	MUX 39			0	MUX 38			0	MUX 37			0	MUX 36			0	MUX 35			0	MUX 34			0	MUX 33			0	MUX 32		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

69.6.1.6 Level1 Mux Selection 5 (L1SEL5)

The following table shows the format of the L1SEL5 register. The values for the fields of the L1SEL5 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured.

Offset: 0x06 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	MUX 47			0	MUX 46			0	MUX 45			0	MUX 44			0	MUX 43			0	MUX 42			0	MUX 41			0	MUX 40		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

69.6.1.7 Level1 Mux Selection 6 (L1SEL6)

The following table shows the format of the L1SEL6 register.The values for the fields of the L1SEL6 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured.

Offset: 0x07 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	MUX 55			0	MUX 54			0	MUX 53			0	MUX 52			0	MUX 51			0	MUX 50			0	MUX 49			0	MUX 48		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

69.6.1.8 Level1 Mux Selection 7 (L1SEL7)

The following table shows the format of the L1SEL7 register. The values for the fields of the L1SEL7 register are device dependent. For a description of these fields, see the device-specific chapter that describes how the modules are configured. All eight inputs of MUX 63 are tied to logic high (1'b1) and output is always high.

Offset: 0x08 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	MUX 63			0	MUX 62			0	MUX 61			0	MUX 60			0	MUX 59			0	MUX 58			0	MUX 57			0	MUX 56		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

69.6.1.9 Input triggers level detection control 2 (ITLD2)

Only a rising edge is detected on the outputs of the first 32 level1 Muxes and therefore these values are not programmable. The last 32 level1 Muxes (Mux 32–63) trigger level is programmable in the Input Triggers Level Detection Control 2 and 3 registers. The following table shows the format of the ITLD2 register. All of the MUX fields in the ITLD2 register have the same description as shown in Table 69-2.

Offset 0x0B Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Mux47		Mux46		Mux45		Mux44		Mux43		Mux42		Mux41		Mux40	
W																

Table continues on the next page...

Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Mux39		Mux38		Mux37		Mux36		Mux35		Mux34		Mux33		Mux32	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The ITLD registers Mux field is described in the following table.

Table 69-2. ITLD register field descriptions

Field	Description
Mux*	Mux Output Signal Detection.
	00 Level is passed after synchronization (default)
	01 Posedge detection
	10 Toggle detection
	11 Level is passed after synchronization

69.6.1.10 Input triggers level detection control 3 (ITLD3)

Only a rising edge is detected on the outputs of the first 32 level1 Muxes and therefore these values are not programmable. The last 32 level1 Muxes (Mux 32–63) trigger level is programmable in the Input Triggers Level Detection Control 2 and 3 registers. The following table shows the format of the ITLD3 register. All of the MUX fields in the ITLD3 register have the same description as shown in Table 69-2.

Offset					0x0C								Access: User read/write			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Mux63		Mux62		Mux61		Mux60		Mux59		Mux58		Mux57		Mux56	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Mux55		Mux54		Mux53		Mux52		Mux51		Mux50		Mux49		Mux48	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

69.6.1.11 CPU0 processor exception vector prefix (C0PEVP)

The SPU captures the exception vector from CPU0 when the exception enable signal is asserted. The exception vector decoding is defined in Table 69-28. The SPU compares this captured vector with the value in the C0PEVP register. The following table shows the format of the C0PEVP register.

Offset: 0x0D Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CPU0															
R																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The C0PEVP register is described in the following table.

Table 69-3. C0PEVP register field descriptions

Field	Description
15:0 CPU0	CPU0 processor exception vector. Program this field with one of the vector values from Table 69-28 to select the corresponding processor exception. **Note:** This field must use the byte swapped values in the CnPEVP register CPUn field value column in Table 69-28.
31:16	Reserved

69.6.1.12 CPU1 processor exception vector prefix (C1PEVP)

The SPU captures the exception vector from CPU1 when the exception enable signal is asserted. The exception vector decoding is defined in Table 69-28. The SPU compares this captured vector with the value in the C1PEVP register. The following table shows the format of the C1PEVP register.

Offset: 0x0E Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CPU1															
R																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The C1PEVP register is described in the following table.

Table 69-4. C1PEVP register field descriptions

Field	Description
15:0 CPU1	CPU1 processor exception vector. Program this field with one of the vector values from Table 69-28 to select the corresponding processor exception. **Note:** This field must use the byte swapped values in the CnPEVP register CPUn field value column in Table 69-28.
31:16	Reserved

69.6.1.13 CPU0 interrupt priority selection (C0PIS)

The following table shows the format of the C0PIS register. The SPU compares the input CPR vector from CPU0 with the value programmed to the C0PIS register and generates a match signal if necessary. The typical use case would be for posedge detection on match (to detect when the priority level is set to the compare value) or high detection while match remains.

Offset: 0x012 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CPU0							
R																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The C0PIS register is described in the following table.

Table 69-5. C0PIS register field descriptions

Field	Description
7:0 CPU0	CPU0 Priority Interrupt Selection.
31:8	Reserved

69.6.1.14 CPU1 interrupt priority selection (C1PIS)

The following table shows the format of the C1PIS register. The SPU compares the input CPR vector from CPU1 with the value programmed to the C1PIS register and generates a match signal if necessary. The typical use case would be for posedge detection on match (to detect when the priority level is set to the compare value) or high detection while match remains.

Offset: 0x013 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CPU1							
R																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The C1PIS register is described in the following table.

Table 69-6. C1PIS register field descriptions

Field	Description
7:0 CPU1	CPU1 Priority Interrupt Selection.
31:8	Reserved

69.6.2 Level2 input Mux configurations registers

At level1, the set of input Mux control registers (described in Level1 input Mux configurations registers) narrows down the large number of SPU watchpoints to a manageable set using sixty-four 8 × 1 multiplexers. At Level2, 16 inputs are selected for each SLU from the 64 level1 outputs using 16 64 × 1 multiplexers. Six bits per Mux select one input out of 64 (in fact 63) inputs. Each 32-bit register configures the AND gates of one state; therefore, eight registers are used to configure inputs for the 8 SLUs. All the 63 inputs are static for all the eight states. Any 16 inputs can be selected from the 63 input triggers for a state.

The detail selection is shown in the following figure for State0. Similar logic is provided for States 1–7.

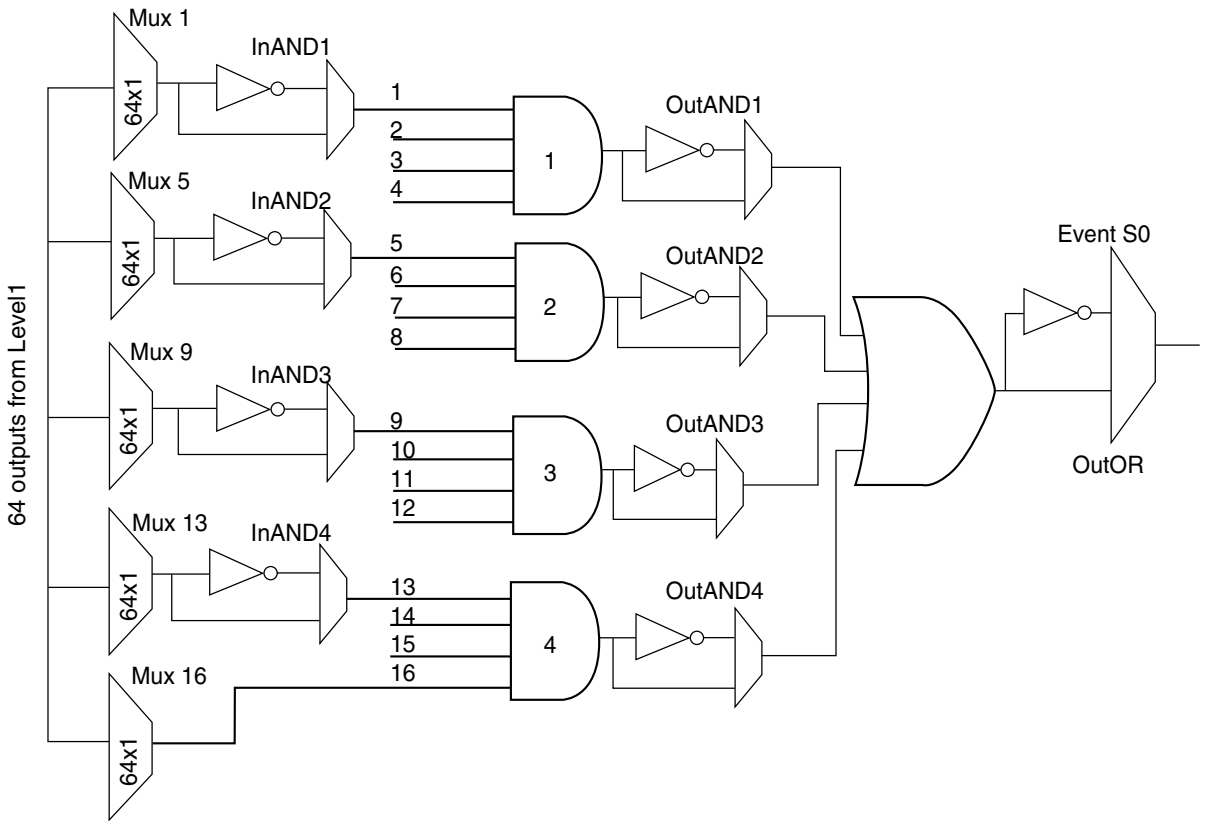


Figure 69-3. Level2 Mux AND gates input pin routing for State0

Note

If any SLU AND gate is unused, then its output is logic low (1'b0). Thus, if all four inputs of a particular AND gate are at the default value 1'b1 (none of the inputs have been routed from the Level 2 mux to the inputs of AND gates), then the output of that AND gate is logic low. This provides protection to avoid the masking of other inputs to the OR gate.

69.6.2.1 Level2 Mux state n selection 0 (L2nSEL0)

The following table shows the format of the L2nSEL0 register, where n, the state number, is equal to 0–7.

Offset:	0x17 (State0)	Access: User read/write
	0x1B (State1)	
	0x1F (State2)	
	0x23 (State3)	
	0x27 (State4)	
	0x2B (State5)	

Table continues on the next page...

0x2F (State6)

0x33 (State7)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	FirstANDInput4						0	0	FirstANDInput3						0	0	FirstANDInput2						0	0	FirstANDInput1					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The L2nSEL0 register fields are described in the following table.

Table 69-7. L2nSEL0 register field descriptions

Field	Description	
5:0	First AND Gate Input1 Selection.	
FirstANDInput1	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0 output
	000010	Level1 MUX 1 output
	000011	Level1 MUX 2 output
	…	
	111110	Level1 MUX 61 output
	111111	Level1 MUX 62 output
7:6	Reserved. Read returns 0.	
13:8	First AND Gate Input 2 Selection.	
FirstANDInput2	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0 output
	000010	Level1 MUX 1 output
	000011	Level1 MUX 2 output
	…	
	111110	Level1 MUX 61 output
	111111	Level1 MUX 62 output
15:14	Reserved. Read returns 0.	
21:16	First AND Gate Input 3 Selection.	
ANDInput3	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0 output
	000010	Level1 MUX 1 output
	000011	Level1 MUX 2 output
	…	
	111110	Level1 MUX 61 output
	111111	Level1 MUX 62 output
23:22	Reserved. Read returns 0.	
29:24	First AND Gate Input 4 Selection.	

Table continues on the next page...

Table 69-7. L2nSEL0 register field descriptions (continued)

Field	Description	
FirstANDInput4	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0 output
	000010	Level1 MUX 1 output
	000011	Level1 MUX 2 output
	…	
	111110	Level1 MUX 61 output
	111111	Level1 MUX 62 output
31:30	Reserved. Read returns 0.	

69.6.2.2 Level2 Mux state n selection 1 (L2nSEL1)

The following table shows the format of the L2nSEL1 register, where n, the state number, is equal to 0–7.

Offset: 0x18 (State0), 0x1C (State1), 0x20 (State2), 0x24 (State3), 0x28 (State4), 0x2C (State5), 0x30 (State6), 0x34 (State7) — Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	SecondANDInput4						0	0	SecondANDInput3						0	0	SecondANDInput2						0	0	SecondANDInput1					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The L2nSEL1 register fields are described in the following table.

Table 69-8. L2nSEL1 register field descriptions

Field	Description
5:0	Second AND Gate Input1 Selection.

Table continues on the next page...

Table 69-8. L2nSEL1 register field descriptions (continued)

Field	Description	
SecondANDInput 1	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0 output
	000010	Level1 MUX 1 output
	000011	Level1 MUX 2 output
	…	
	111110	Level1 MUX 61 output
	111111	Level1 MUX 62 output
7:6	Reserved. Read returns 0.	
13:8	Second AND Gate Input 2 Selection.	
SecondANDInput 2	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0 output
	000010	Level1 MUX 1 output
	000011	Level1 MUX 2 output
	…	
	111110	Level1 MUX 61 output
	111111	Level1 MUX 62 output
15:14	Reserved. Read returns 0.	
21:16	Second AND Gate Input 3 Selection.	
SecondANDInput 3	000000	No Input is selected (tie to 1'b1)
	00000	Level1 MUX 0 output
	000010	Level1 MUX 1 output
	000011	Level1 MUX 2 output
	…	
	111110	Level1 MUX 61 output
	111111	Level1 MUX 62 output
23:22	Reserved. Read returns 0.	
29:24	Second AND Gate Input 4 Selection.	
SecondANDInput 4	000001	Level1 MUX 0 output
	000010	Level1 MUX 1 output
	000011	Level1 MUX 2 output
	…	
	111110	Level1 MUX 61 output
	111111	Level1 MUX 62 output
31:30	Reserved. Read returns 0.	

69.6.2.3 Level2 Mux state n selection 2 (L2nSEL2)

The following table shows the format of the L2nSEL2 register, where n, the state number, is equal to 0–7.

Offset: 0x19 (State0), 0x1D (State1), 0x25 (State3), 0x21 (State2), 0x29 (State4), 0x2D (State5), 0x31 (State6), 0x35 (State7) — Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	ThirdANDInput4						0	0	ThirdANDInput3						0	0	ThirdANDInput2						0	0	ThirdANDInput1					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The L2nSEL2 register fields are described in the following table.

Table 69-9. L2nSEL2 register field descriptions

Field	Description	
5:0	Third AND Gate Input1 Selection.	
ThirdANDInput1	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0 output
	000010	Level1 MUX 1 output
	000011	Level1 MUX 2 output
	…	
	111110	Level1 MUX 61 output
	111111	Level1 MUX 62 output
7:6	Reserved. Read returns 0.	
13:8	Third AND Gate Input 2 Selection.	
ThirdANDInput2	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0 output
	000010	Level1 MUX 1 output
	000011	Level1 MUX 2 output
	…	
	111110	Level1 MUX 61 output
	111111	Level1 MUX 62 output
15:14	Reserved. Read returns 0.	
21:16	Third AND Gate Input 3 Selection.	

Table continues on the next page...

Table 69-9. L2nSEL2 register field descriptions (continued)

Field	Description	
ThirdANDInput3	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0 output
	000010	Level1 MUX 1 output
	000011	Level1 MUX 2 output
	…	
	111110	Level1 MUX 61 output
	111111	Level1 MUX 62 output
23:22	Reserved. Read returns 0.	
29:24	Third AND Gate Input 4 Selection.	
ThirdANDInput4	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0 output
	000010	Level1 MUX 1 output
	000011	Level1 MUX 2 output
	…	
	111110	Level1 MUX 61 output
	111111	Level1 MUX 62 output
31:30	Reserved. Read returns 0.	

69.6.2.4 Level2 Mux state n selection 3 (L2nSEL3)

The following table shows the format of the L2nSEL3 register, where n, the state number, is equal to 0–7.

Offset: 0x1A (State0), 0x1E (State1), 0x22 (State2), 0x26 (State3), 0x2A (State4), 0x2E (State5), 0x32 (State6), 0x36 (State7) — Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	FourthANDInput4						0	0	FourthANDInput3						0	0	FourthANDInput2						0	0	FourthANDInput1					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The L2nSEL3 register fields are described in the following table.

Table 69-10. L2nSEL3 register field descriptions

Field	Description	
5:0	Fourth AND Gate Input1 Selection.	
FourthANDInput1	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0
	000010	Level1 MUX 1
	000011	Level1 MUX 2
	…	
	111110	Level1 MUX 61
	111111	Level1 MUX 62
7:6	Reserved. Read returns 0.	
13:8	Fourth AND Gate Input 2 Selection.	
FourthANDInput2	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0
	000010	Level1 MUX 1
	000011	Level1 MUX 2
	…	
	111110	Level1 MUX 61
	111111	Level1 MUX 62
15:14	Reserved. Read returns 0.	
21:16	Fourth AND Gate Input 3 Selection.	
FourthANDInput3	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0
	000010	Level1 MUX 1
	000011	Level1 MUX 2
	…	
	111110	Level1 MUX 61
	111111	Level1 MUX 62
23:22	Reserved. Read returns 0.	
29:24	Fourth AND Gate Input 4 Selection.	
FourthANDInput4	000000	No Input is selected (tie to 1'b1)
	000001	Level1 MUX 0
	000010	Level1 MUX 1
	000011	Level1 MUX 2
	…	
	111110	Level1 MUX 61
	111111	Level1 MUX 62
31:30	Reserved. Read returns 0.	

69.6.3 Sequence unit registers

69.6.3.1 Input/output Inversion control state n (IOICn)

An input control register is provided for each event generated by the SLU. This register is programmed to specify how the input signals to each SLU are combined to generate the resulting event for that particular sequence. In total, there are 16 inputs for each state and only one (the first) of the inputs can be used with or without inversion. The input can be left unused if not required for that state by selecting 000000 (No input selected option) in the L2nSEL registers. The output of each AND gate has the optional inversion control as well. The output can be routed directly or with inversion from each AND gate to the input of OR gate. The following table shows the format of the IOICn register, where n, the state number, is equal to 0–7.

Offset 0x37 (State0) Access: User read/write
0x38 (State1)
0x39 (State2)
0x3A (State3)
0x3B (State4)
0x3C (State5)
0x3D (State6)
0x3E (State7)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	OutOR	OutAND 4	OutAND 3	OutAND 2	OutAND 1	InAND4	InAND3	InAND2	InAND1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The IOICn register fields are described in the following table.

Table 69-11. IOICn register field descriptions

Field	Description
0	Input 1 AND Gate 1 Control.
InAND1	0 Input 1 is used directly (without inversion) for this event 1 Input 1 is inverted for this event
1	Input 1 AND Gate 2 Control.

Table continues on the next page...

Table 69-11. IOICn register field descriptions (continued)

Field	Description
InAND2	0 Input 1 is used directly (without inversion) for this event
	1 Input 1 is inverted for this event
2	Input 1 AND Gate 3 Control.
InAND3	0 Input 1 is used directly (without inversion) for this event
	1 Input 1 is inverted for this event
3	Input 1 AND Gate 4 Control.
InAND4	0 Input 1 is used directly (without inversion) for this event
	1 Input 1 is inverted for this event
4	Output AND Gate 1 Control.
OutAND1	0 Output is used directly (without inversion) for this event
	1 Output is inverted for this event
5	Output AND Gate 2 Control.
OutAND2	0 Output is used directly (without inversion) for this event
	1 Output is inverted for this event
6	Output AND Gate 3 Control.
OutAND3	0 Output is used directly (without inversion) for this event
	1 Output is inverted for this event
7	Output of AND Gate 4 Control.
OutAND4	0 Output is used directly (without inversion) for this event
	1 Output is inverted for this event
8	Output OR GATE Control.
OutOR	0 Output is used directly (without inversion) for this event
	1 Output is inverted for this event
9:31	Reserved. Read returns 0.

69.6.3.2 Sequence control (SCTRL)

This configuration register is provided to enable the sequence and to select the states that are part of a particular sequence. There can be at most four active sequences. Each sequence can have one state or a maximum of eight states. However, each state can only be used in one unique sequence. In other words, if one state has been selected for a sequence, then this state cannot be used for any other defined sequence. Each sequence has a start state and end state. The following table shows the format of the SCTRL register.

Offset 0x3F Access: User read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

Table continues on the next page...

R	0	ST2			ST1			ST0	0	ST2			ST1			ST0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	ST2			ST1			ST0	0	ST2			ST1			ST0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The SCTRL register fields are described in the following table.

Table 69-12. SCTRL register field descriptions

Field	Description
0	Sequence1 Enable/Disable[1, 1]. Once enabled, this bit is disabled at the end of the Sequence 1.
ST0	0 Sequence 1 is disabled
	1 Sequence 1 is enabled
3:1	Sequence 1 Start State.
ST1	000 State 0 (Always)
6:4	Sequence 1 End State.
ST2	000 State 0 is the Stop state for sequence 1
	001 State 1 is the Stop state for sequence 1
	010 State 2 is the Stop state for sequence 1
	011 State 3 is the Stop state for sequence 1
	100 State 4 is the Stop state for sequence 1
	101 State 5 is the Stop state for sequence 1
	110 State 6 is the Stop state for sequence 1
	111 State 7 is the Stop state for sequence 1
7	Reserved. Read returns 0.
8	Sequence2 Enable/Disable[1]. Once enabled, this bit is disabled at the end of the Sequence 2.
ST0	0 Sequence 2 is disabled
	1 Sequence 2 is enabled
11:9	Sequence 2 Start State.[2, 2]
ST1	000 State 0 is the Start state for sequence 2
	001 State 1 is the Start state for sequence 2
	010 State 2 is the Start state for sequence 2
	011 State 3 is the Start state for sequence 2
	100 State 4 is the Start state for sequence 2
	101 State 5 is the Start state for sequence 2
	110 State 6 is the Start state for sequence 2
	111 State 7 is the Start state for sequence 2
14:12	Sequence 2 End State.

Table continues on the next page...

Table 69-12. SCTRL register field descriptions (continued)

Field	Description	
ST2	000	State 0 is the Stop state for sequence 2
	001	State 1 is the Stop state for sequence 2
	010	State 2 is the Stop state for sequence 2
	011	State 3 is the Stop state for sequence 2
	100	State 4 is the Stop state for sequence 2
	101	State 5 is the Stop state for sequence 2
	110	State 6 is the Stop state for sequence 2
	111	State 7 is the Stop state for sequence 2
15	Reserved. Read returns 0.	
16	Sequence 3 Enable/Disable[1]. Once enabled, this bit is disabled at the end of the Sequence 3.	
ST0	0	Sequence 3 is disabled
	1	Sequence 3 is enabled
19:17	Sequence 3 Start State.[2]	
ST1	000	State 0 is the Start state for sequence 3
	001	State 1 is the Start state for sequence 3
	010	State 2 is the Start state for sequence 3
	011	State 3 is the Start state for sequence 3
	100	State 4 is the Start state for sequence 3
	101	State 5 is the Start state for sequence 3
	110	State 6 is the Start state for sequence 3
	111	State 7 is the Start state for sequence 3
22:20	Sequence 3 End State.	
ST2	000	State 0 is the Stop state for sequence 3
	001	State 1 is the Stop state for sequence 3
	010	State 2 is the Stop state for sequence 3
	011	State 3 is the Stop state for sequence 3
	100	State 4 is the Stop state for sequence 3
	101	State 5 is the Stop state for sequence 3
	110	State 6 is the Stop state for sequence 3
	111	State 7 is the Stop state for sequence 3
23	Reserved. Read returns 0.	
24	Sequence 4 Enable/Disable[1]. Once enabled, this bit is disabled at the end of the Sequence 4.	
ST0	0	Sequence 4 is disabled
	1	Sequence 4 is enabled
27:25	Sequence 4 Start State.[2]	

Table continues on the next page...

Table 69-12. SCTRL register field descriptions (continued)

Field	Description	
ST1	000	State 0 is the Start state for sequence 4
	001	State 1 is the Start state for sequence 4
	010	State 2 is the Start state for sequence 4
	011	State 3 is the Start state for sequence 4
	100	State 4 is the Start state for sequence 4
	101	State 5 is the Start state for sequence 4
	110	State 6 is the Start state for sequence 4
	111	State 7 is the Start state for sequence 4
30:28	Sequence 4 End State.	
ST2	000	State 0 is the Stop state for sequence 4
	001	State 1 is the Stop state for sequence 4
	010	State 2 is the Stop state for sequence 4
	011	State 3 is the Stop state for sequence 4
	100	State 4 is the Stop state for sequence 4
	101	State 5 is the Stop state for sequence 4
	110	State 6 is the Stop state for sequence 4
	111	State 7 is the Stop state for sequence 4
31	Reserved. Read returns 0.	

1. Multiple active sequences such as Sequence 2, Sequence 3 and Sequence 4 must be programmed with Div4 option only.
2. Ideally, the start state for a sequence should be the state immediately after the stop state for the previous sequence. For example, if sequence 1 starts at state 0 and ends at state3, sequence 2 should start at state4.

69.6.3.3 State n goto control (SnGC)

SLU states are linked to each other based on if-then-else conditions. There are two conditions: true and false. A SLU state can remain in the same state or jump to any other state. The state goto control register is programmed to specify the GoTo a particular state from current state.

The following table shows the format of the SnGC register, where n, the state number, is equal to 0–7.

Offset: 0x40 (State0) Access: User read/write

0x41 (State1)

0x42 (State2)

0x43 (State3)

0x44 (State4)

0x45 (State5)

0x46 (State6)

Table continues on the next page...

0x47 (State7)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NF			FE	NT			TE
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The SnGC register fields are described in the following table.

Table 69-13. SnGC register field descriptions

Field	Description	
0	GoTo Control of State Under True Condition. Defines the Goto control of a state under the true condition. If this bit is enabled, then the State moves to the next state based on bits[3:1]. If disabled, then this is the end of a sequence when the sequence comes to this state. Actions associated with states under true condition are executed. Disabling this bit ends the sequence under a true condition.	
TE	0	Disable Goto control
	1	Enable Goto control
3:1	Next State on True Condition. Defines the Goto next state in case of true condition from the present state. States may remain in the same state on a true condition if the same state is chosen as the next state.	
NT	000	Goto State 0
	001	Goto State 1
	010	Goto State 2
	011	Goto State 3
	100	Goto State 4
	101	Goto State 5
	110	Goto State 6
	111	Goto State 7
4	GOTO Control of State Under False Condition. Defines the Goto control of a state under the false condition. If this bit is enabled, then the state moves to the next state based on bits[7:5] . If disabled, then this is the is end of a sequence when the sequence comes to this state. Actions associated with states under false condition are executed. Disabling this bit end the sequence under a false condition.	
FE	0	Disable Goto control
	1	Enable Goto control
7:5	Next State on False Condition. Defines the Goto next state in case of false condition from the present state. States may remain in the same state on a false condition if the same state is chosen as the next state.	
NF	000	Goto State 0
	001	Goto State 1
	010	Goto State 2
	011	Goto State 3
	100	Goto State 4
	101	Goto State 5
	110	Goto State 6
	111	Goto State 7
31:8	Reserved. Read returns 0.	

69.6.4 SLU status (SS) register

The SLU status register indicates the states that have been hit/encountered in any of the active sequences. The following table shows the format of the SS register.

Offset 0x48 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	IM1	IM0	0	EM1	EM0	SAS4			SAS3
W								w1c	w1c		w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SAS3		SAS2		SAS1				SH7	SH6	SH5	SH4	SH3	SH2	SH1	SH0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The SS register fields are described in the following table.

Table 69-14. SS register field descriptions

Field	Description
0	State Hit 0. This bit is set when State0 is hit/encountered in any of the active sequences
SH0	0 State not hit
	1 State hit
1	State Hit 1. This bit is set when State1 is hit/encountered in any of the active sequences
SH1	0 State not hit
	1 State hit
2	State Hit 2. This bit is set when State2 is hit/encountered in any of the active sequences
SH2	0 State not hit
	1 State hit
3	State Hit 3. This bit is set when State3 is hit/encountered in any of the active sequences
SH3	0 State not hit
	1 State hit
4	State Hit 4. This bit is set when State4 is hit/encountered in any of the active sequences
SH4	0 State not hit
	1 State hit
5	State Hit 5. This bit is set when State5 is hit/encountered in any of the active sequences
SH5	0 State not hit
	1 State hit
6	State Hit 6. This bit is set when State6 is hit/encountered in any of the active sequences
SH6	0 State not hit
	1 State hit

Table continues on the next page...

Table 69-14. SS register field descriptions (continued)

Field	Description	
7	State Hit 7. This bit is set when State7 is hit/encountered in any of the active sequences	
SH7	0	State not hit
	1	State hit
10 :8	Sequence 1 Active State. Two consecutive read cycles are required to know the active state. These bits capture the active state of sequence 1 at the clock when these bits are read. The active state might be different in the previous clock and the next clock.	
SAS1	000	State0 is the active state
	001	State1 is the active state
	010	State2 is the active state
	011	State3 is the active state
	100	State4 is the active state
	101	State5 is the active state
	110	State6 is the active state
	111	State7 is the active state
13 :11	Sequence 2 Active State. Two consecutive read cycles are required to know the active state. These bits capture the active state of sequence 2 at the clock when these bits are read. The active state might be different in the previous clock and the next clock.	
SAS2	000	State0 is the active state
	001	State1 is the active state
	010	State2 is the active state
	011	State3 is the active state
	100	State4 is the active state
	101	State5 is the active state
	110	State6 is the active state
	111	State7 is the active state
16:14	Sequence 3 Active State. Two consecutive read cycles are required to know the active state. These bits capture the active state of sequence 3 at the clock when these bits are read. The active state might be different in the previous clock and the next clock.	
SAS3	000	State0 is the active state
	001	State1 is the active state
	010	State2 is the active state
	011	State3 is the active state
	100	State4 is the active state
	101	State5 is the active state
	110	State6 is the active state
	111	State7 is the active state
19 :17	Sequence 4 Active State. Two consecutive read cycles are required to know the active state. These bits capture the active state of sequence 4 at the clock when these bits are read. The active state might be different in the previous clock and the next clock.	

Table continues on the next page...

Table 69-14. SS register field descriptions (continued)

Field	Description	
SAS4	000	State0 is the active state
	001	State1 is the active state
	010	State2 is the active state
	011	State3 is the active state
	100	State4 is the active state
	101	State5 is the active state
	110	State6 is the active state
	111	State7 is the active state
20	Exception Match 0. This bit indicates the status of the processor exception vector match value of CPU0.	
EM0	0	No match
	1	Match occurred
21	Exception Match 1. This bit indicates the status of the processor exception vector match value of CPU1.	
EM1	0	No match
	1	Match occured
22	Reserved	
23	Interrupt Match 0. This bit indicates the status of the match value on interrupt priority register of CPU0.	
IM0	0	No match
	1	Match occurred
24	Interrupt Match 1. This bit indicates the status of the match value on interrupt priority register of CPU1.	
IM1	0	No match
	1	Match occurred
25	Reserved	
31:26	Reserved. Read returns 0	

69.6.5 SPU enable (SE) register

The SPU enable/disable register is used to enable and disable the SPU. If disabled, all inputs are masked. The following table shows the format of the SE register.

Offset: 0x49 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DIV	SED
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The SE register fields are described in the following table.

Table 69-15. SE register field descriptions

Field	Description
0	SPU Enable/Disable. This bit should be set after programming all of the SPU configuration registers.
SED	0 SPU is disabled
	1 SPU is enabled
1	Div Option. Select Div4 and Div1 programmable options based on number of active sequences.
DIV	0 Use Div4 option, when multiple up to four sequences are active or only one sequence is active which does not require the highest resolution.
	1 Use a Div1 option, when only one sequence (Sequence1) is active and highest resolution is required.
31:2	Reserved

NOTE

Div option mapping with active sequences must be correctly programmed by the user. No hardware checks are implemented.

69.6.6 Sample and hold mechanism

SPU supports Div4/Div1 sample and state processing frequency (both are effectively the same). For the Div1 option, the sample and state processing frequency is the SPU clock frequency. Likewise, for the Div4 option, the sample and state processing frequency is 1/4th the SPU clock frequency.

The Div1 or Div4 is related to hold circuitry (among other circuits as well), but the sample circuitry is always Div1. Therefore, if Div4 is selected for hold circuitry, then four samples are taken and then latched into a hold circuitry (for the next four Div1 clocks, synchronized for all inputs).

This sample and hold mechanism is applicable for external signals only, but counters logic process differently. These external signals may be considered as selected conditions for any sequence, and for any of the states of any sequence. There is no difference based on state or sequence, as all inputs are treated the same regardless.

69.6.6.1 Functional limitations

If there are multiple true conditions in a Div4 period, only one condition is registered. Never are there any missed true conditions, except for extra true conditions in a Div4 period. Therefore, the user should select the resolution of SPU based on number of active sequences.

- If the signal changes faster than the sample and state frequency, then the SPU would probably only detect one event per the sample period (for example: If there are multiple true conditions in a Div4 period, only one condition is registered).
- If the signal changes faster than the sample and state frequency, then the SPU would only be able to process one event per state processing frequency.

69.6.6.2 Assumption

When using Div1, SPU counter operations are not available for state evaluation in the immediately next clock cycle. Therefore, special handling must be done in state evaluation during this delayed effect to assure that it does not create undesirable results. Refer to counter workaround description in SPU actions.

When using Div4, however, there are no delayed effects when using SPU counter operations. This is because subsequent state evaluations are done four clocks apart, instead of every clock for Div1 selection. Therefore, counter workaround description in SPU actions, becomes redundant with Div4 option.

69.6.7 SLU action unit registers

The action unit of the SPU is responsible for generating the action triggers supported by various actions. List of the actions is given in SPU actions. The action triggers are mapped to the events by programming the action registers. Each event can be programmed to trigger interrupts, watchpoints, trace signals, counter stop/reset/increment and many other general actions. The following table shows the bit mapping for all the possible actions. All possible actions are divided in three groups: client based, counter based, system related.

Table 69-16. SPU action decoder

Group bit [11:10]		Client/counter bit [9:5]		Action bit [4:1]			Enable bit [0]
ID	Name	ID	Name	ID	Name	Comment	
00	Client group	00000	CPU0	0000	No action	—	0=Stop 1=Start
		00001	CPU1	0001	DTM	Valid for Cores, AHB, and trace group actions.	
		00010	Reserved	0010	OTM		
		00011–00111	Reserved	0011	PTM		
		01000	Reserved	0100	WTM		
		01001–01111	Reserved	0101	PMC1/NAR_supp_trig1	Valid for the cores as PMC; Valid for NAR as the suppress trigger actions	For PMC1/2/3/4 0= Stop 1= Start; For NAR suppress trigger = Don't care
		10000	NXMC0	0110	PMC2/NAR_supp_trig2		
		10001	NXMC1	0111	PMC3/NAR_supp_trig3		
		10010–10111	Reserved	1000	PMC4/NAR_supp_trig4		
		11000	NAR	1001	Reserved	—	0=Stop 1=Start
		11001–11110	Reserved	1010	Reserved		
				1011	Reserved		
				1100	Reserved		
		11111	Trace group[1]	1101–1111	Reserved	Valid for CPU cores and AHB	
01	Counter group	00000	Counter0	0000	No action	—	Don't care
		00001	Counter1	0001	Start timer/Incr counter	Valid for all counters/timers	
		00010	Counter2	0010	Stop timer		
		…	…	0011	Reset counter		
		01111	Counter15	0100	Capture		
		10000–11111	Reserved	0101–1111	Reserved		
10	System group	00000	System action	0000	No action	—	Don't care
		00001–11111	Reserved	0001	Halt1_dci_evti	—	
				0010	Halt2_dci_evti	—	
				0011	Timestamp0_NXMC0	Valid for the NXMC clients	
				0100	Timestamp1_NXMC0		
				0101	Timestamp0_NXMC1		
				0110	Timestamp1_NXMC1		
				0111	Evto1	—	
				1000	Evto2	—	
				1001	NAR_sync_trigg	Valid for NAR	
				1010	Core INTR	—	
				1011–1111	Reserved	—	—
11	Reserved						

1. The valid actions for Trace group are OTM trace, PTM trace, DTM trace or WTM trace. No other actions can be selected with Trace group. The group can be defined using the Trace group configuration registers.

69.6.7.1 State n true action 0 (SnTA0)

The SPU can be programmed to enable up to four different actions against a True condition of every state. All the possible actions and their encoding are listed in Table 69-16. A total of 12 bits are needed to code every action. The first two actions for true condition are defined in SnTA0 and the next two actions are defined in SnTA1.

The following table shows the format of the SnTA0 register, where n, the state number, is equal to 0–7.

Offset: 0x01 (State0), 0x03 (State1), 0x05 (State2), 0x07 (State3), 0x09 (State4), 0x0B (State5), 0x0D (State6), 0x0F (State7) — Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	ACTION2												0	0	0	0	ACTION1											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The SnTA0 register fields are described in the following table.

Table 69-17. SnTA0 register field descriptions

Field	Description
0:11 ACTION1	Action 1 to be performed for State n true condition. The decoding of these bits is defined in Table 69-16
12:15	Reserved
16:27 ACTION2	Action 2 to be performed for State n true condition. The decoding of these bits is defined in Table 69-16
28:31	Reserved

69.6.7.2 State n true action 1 (SnTA1)

The SPU can be programmed to enable up to four different actions against a True condition of every state. All the possible actions and their encoding are listed in Table 69-16. A total of 12 bits are needed to code every action. The first two actions for true

condition are defined in SnTA0 and the next two actions are defined in SnTA1. The following table shows the format of the SnTA1 register, where n, the state number, is equal to 0–7.

Offset: 0x02 (State0), 0x04 (State1), 0x06 (State2), 0x08 (State3), 0x0A (State4), 0x0C(State5), 0x0E (State6), 0x10 (State7) — Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	ACTION4												0	0	0	0	ACTION3											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The SnTA1 register fields are described in the following table.

Table 69-18. SnTA1 register field descriptions

Field	Description
0:11 ACTION3	Action 3 to be performed for State n true condition. The decoding of these bits is defined in Table 69-16
12:15	Reserved
16:27 ACTION4	Action 4 to be performed for State n true condition. The decoding of these bits is defined in Table 69-16
28:31	Reserved

69.6.7.3 State n false action 0 (SnFA0)

The SPU can be programmed to enable up to four different actions against a False condition of every state. All the possible actions and their encoding are listed in Table 69-16. A total of 12 bits are needed to code every action. The first two actions for false condition are defined in SnFA0 and the next two actions are defined in SnFA1. The following table shows the format of the SnFA0 register, where n, the state number, is equal to 0–7.

Offset: 0x11 (State0), 0x13 (State1) — Access: User read/write

Table continues on the next page...

0x15 (State2)
0x17 (State3)
0x19 (State4)
0x1B (State5)
0x1D (State6)
0x1F (State7)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	ACTION2												0	0	0	0	ACTION1											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The SnFA0 register fields are described in the following table.

Offset: 0x12 (State0) Access: User read/write
0x14 (State1)
0x16 (State2)
0x18 (State3)
0x1A (State4)
0x1C(State5)
0x1E (State6)
0x20 (State7)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	ACTION4												0	0	0	0	ACTION3											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The SnFA0 register fields are described in the following table.

Table 69-19. SnFA0 register field descriptions

Field	Description
0:11 ACTION1	Action 1 to be performed for State n false condition. The decoding of these bits is defined in Table 69-16
12:15	Reserved
16:27 ACTION2	Action 2 to be performed for State n false condition. The decoding of these bits is defined in Table 69-16
28:31	Reserved

69.6.7.4 State n false action 1 (SnFA1)

The SPU can be programmed to enable up to four different actions against a False condition of every state. All the possible actions and their encoding are listed in Table 69-16. A total of 12 bits are needed to code every action. The first two actions for false condition are defined in SnFA0 and the next two actions are defined in SnFA1. The following table shows the format of the SnFA1 register, where n, the state number, is equal to 0–7.

Offset: 0x12 (State0), 0x14 (State1), 0x16 (State2), 0x18 (State3), 0x1A (State4), 0x1C(State5), 0x1E (State6), 0x20 (State7) — Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	ACTION4												0	0	0	0	ACTION3											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The SnFA1 register fields are described in the following table.

Table 69-20. SnFA1 register field descriptions

Field	Description
0:11 ACTION3	Action 3 to be performed for State n false condition. The decoding of these bits is defined in Table 69-16
12:15	Reserved
16:27 ACTION4	Action 4 to be performed for State n false condition. The decoding of these bits is defined in Table 69-16
28:31	Reserved

69.6.7.5 Trace group configuration registers (DTMGC, PTMGC, OTMGC and WTMGC)

There are four action trace groups and a configuration register for each trace group:

- DTM trace—clients are configured with the DTM trace group configuration (DTMGC) register

- PTM trace—clients are configured with the PTM trace group configuration (PTMGC) register
- OTM trace—clients are configured with the OTM trace group configuration (OTMGC) register
- WTM trace—clients are configured with the WTM trace group configuration (WTMGC) register

Each of these groups are specific to a trace. Any number of clients can be enabled for one trace using the action trace group.

The following table shows the format of the trace group configuration registers.

Offset 0x21 (DTMGC) 0x22 (PTMGC) 0x23 (OTMGC) 0x24 (WTMGC) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AHB2	AHB1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CPU1	CPU0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The trace group configuration register fields are described in the following table.

Table 69-21. Trace group configuration register field descriptions

Field	Description
0 CPU0	0 Disable trace for CPU0 1 Enable trace for CPU0
1 CPU1	0 Disable trace for CPU1 1 Enable trace for CPU1
15:2	Reserved
16 AHB1	0 Disable trace for AHB1 1 Enable trace for AHB1
17 AHB2	0 Disable trace for AHB2 1 Enable trace for AHB2
31:18	Reserved

69.6.7.6 Interrupt status (INTS)

The status of the external interrupt request to the processor core through the JTAGM is registered in the Interrupt status (INTS) register. This status bit is cleared when the acknowledge is received from the JTAGM. The following table shows the format of the INTS register.

Offset: 0x25 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	INT
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The INTS register fields are described in the following table.

Table 69-22. INTS register field descriptions

Field	Description
0 INT	1 Interrupt status bit is cleared from JTAGM
31:1	Reserved

69.6.8 Counters control n (CCTRLn) registers

There are 16 configurable counters/timers in the SPU. Configuration control of these counters is defined in the counter control registers. The following table shows the format of the CCTRLn registers, where n, the counter number, is equal to 0–15.

Offset 0x26–0x35 (CCTRL0–15) Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	SW_RESET	PRESCALING			0	COUNT_INCR	MODE	ENABLE
W								0								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The CCTRLn register fields are described in the following table.

Table 69-23. CCTRLn register field descriptions

Field	Description	
0	Enable.	
ENABLE	0	Counter disabled
	1	Counter enabled
1	Mode Select. Select counter or timer operation.	
MODE	0	Counter
	1	Timer
2	Count Increment.	
COUNT_INCR	0	Increment the counter based on the actions generated from the sequences
	1	Increment the counter based on the PMC input. The mapping of the PMC inputs with the counters is shown in Figure 69-11.
3	Reserved. Read returns 0.	
6:4	Prescaling. Divides the timer clock by a defined value.	
PRESCALING	000	No scaling, divide by 1
	001	Divide by 4
	010	Divide by 16
	011	Divide by 32
	100	Reserved
	101	Reserved
	110	Reserved
	111	Reserved
7	Reset. Counter can be reset by software	
SW_RESET	0	Software reset is disabled
	1	Software reset
8	This bit is for internal use only and must be set to 0.	
Reserved		
31:9	Reserved. Read returns 0.	

69.6.9 Counters compare n (CCMPn) registers

A 32-bit comparator value can be programmed for each of the counters. This value is compared with the corresponding counter/timer to generate an event for the SPU. The following table shows the format of the CCMPn registers, where n, the counter number, is equal to 0–15.

Offset: 0x36–0x45 (CCMP0–15) Access: User read/write

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table continues on the next page...

R	COMPARE_VAL																													
W																														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The CCMPn register fields are described in the following table.

Table 69-24. CCMPn register field descriptions

Field	Description
31:0 COMPARE_VAL	Compare value. This value is compared with the corresponding counter/timer value to generate an event for the SPU.

69.6.10 Status registers

69.6.10.1 Counter compare status (CCOMS)

The counter compare status register shows the status of the comparisons for all the counters. The bits in this register can be cleared by writing a 1 to them. The following table shows the format of the CCOMS register. All of the CC fields in the CCOMS register have the same description as shown in Table 69-25. Counter compare values should not be changed dynamically. Thus, these fields should be cleared (by writing 1) only during these conditions:

- Counter is in disabled state (ENABLE bit of CCTRLn set to 0).
- Software reset bit (SW_RESET) is set.

Offset 0x46 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CC15	CC14	CC13	CC12	CC11	CC10	CC9	CC8	CC7	CC6	CC5	CC4	CC3	CC2	CC1	CC0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The CCOMS register fields are described in the following table, where n is the counter number.

Table 69-25. CCOMS register field descriptions

Field	Description	
CCn	Counter n Compare.	
	0	Counter status is not true
	1	Counter status is true

69.6.10.2 Counter overflow status (COS)

Whenever a counter overflow occurs, the corresponding counter overflow status register bit is set. The bits in this register can be cleared by writing a 1 to them. The following table shows the format of the COS register. All of the CO fields in the COS register have the same description as shown in Table 69-26.

Offset 0x47 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CO15	CO14	CO13	CO12	CO11	CO10	CO9	CO8	CO7	CO6	CO5	CO4	CO3	CO2	CO1	CO0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The COS register fields are described in the following table, where n is the counter number.

Table 69-26. COS register field descriptions

Field	Description	
COn	Counter n Overflow.	
	0	Counter did not overflow
	1	Counter overflow occurred

69.6.10.3 Counter capture status (CCAPS)

Whenever a counter value is captured, the corresponding capture status register bit for that counter is set. If the counter capture value causes a FIFO overflow, and thus the captured value is lost, the status bit is not set. The bits in this register can be cleared by writing a 1 to them. The following table shows the format of the CCAPS register.

All of the CC fields in the CCAPS register have the same description as shown in Table 69-27.

Offset 0x48 Access: User read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CC15	CC14	CC13	CC12	CC11	CC10	CC9	CC8	CC7	CC6	CC5	CC4	CC3	CC2	CC1	CC0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The CCAPS register fields are described in the following table, where n is the counter number.

Table 69-27. CCAPS register field descriptions

Field	Description	
CCn	Counter n Captured.	
	0	Counter not captured
	1	Counter captured

69.7 Functional description

This section discusses the functional description of the SPU.

69.7.1 Input Mux unit

The SPU receives numerous inputs in the form of complex triggers from various clients to create a predefined sequence. One state logic can accommodate 16 inputs at one point of time. Therefore, an input Muxing unit becomes essential for reducing the debug and

performance events to a manageable set of 16 inputs for each SLU. In addition to client input signals, there are certain performance events which are fed directly to the different counters inside the SPU.

The selection of 16 inputs for each state logic is done at two level of muxing both of which are shown in Figure 69-4.

- Level 1 Muxing—At this level, there are 167 inputs. Out of these possible inputs, 63 inputs are selected. This selection is done using an 8:1 Mux. Therefore, sixty-four 8×1 muxes are required. The set of 63 watchpoints are passed to the second level of muxing. The selection mechanism is shown in Figure 69-4.
- Level 2 Muxing—At this level, 16 inputs are selected out of the 63 possible inputs for each SLU. These 63 inputs consist of multiple set of 8 watchpoints from different clients, processor exception signals, interrupt from clients, and event IN/OUT signals. In addition, certain events from counters are used as feedback in the present state logic of each state.

  At level two there are 16 banks of 64:1 muxes for the selection of 16 inputs for one state logic. The inputs to these Muxes are expected to be somewhat unique. In general, care should be take to choose signal assignments that allow for key use cases to be achieved without encountering concurrency restrictions. Since there are eight such states, 16 × 8 banks of 64:1 mux are required. The selection mechanism is shown in the following figure.

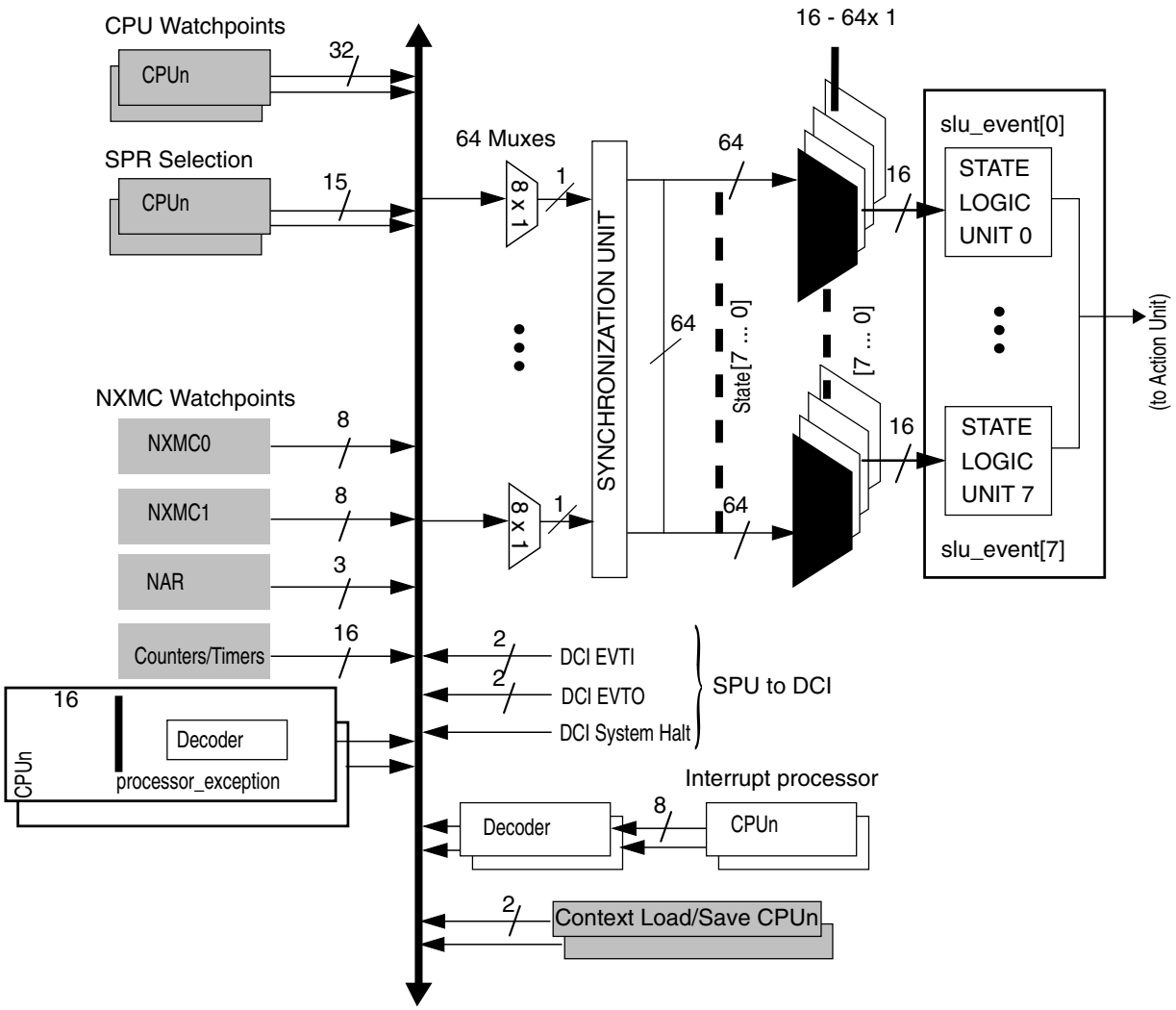


Figure 69-4. Input Mux unit block diagram

69.7.2 Processor exception vector encoding

Active-high input signals from the CPU indicate the particular exception being processed when the CPU exception enable signal is asserted. These signals can be used for debug triggering mechanisms. The SPU captures the CPU exception vector whenever the CPU exception enable signal is asserted and the SPU uses the exception vector as an input condition as shown in the following table. The CPU exception vector values from the CPUs must be byte swapped before they are programmed to the CPUn field of the CnPEVP registers as shown in the following table.

Table 69-28. Processor exception processing conditions

CPU exception vector	CnPEVP register CPUn field value	Exception condition
0x0000	0x0000	Critical Input—autovectored
0x0010	0x1000	Machine check
0x0020	0x2000	Data storage
0x0030	0x3000	Instruction storage
0x0040	0x4000	External input—autovectored
0x0050	0x5000	Alignment
0x0060	0x6000	Program trap
0x0061	0x6100	Program illegal
0x0062	0x6200	Program privileged
0x0070	0x7000	Performance monitor
0x0080	0x8000	System call
0x0090	0x9000	Debug
0x00A0	0xA000	EFPU data exception
0x00B0	0xB000	EFPU round exception
0xVVVV3, low two bits = 01	0xVVVV3, high two bits = 01	External input—vectored
0xVVVV3, low two bits = 01	0xVVVV3, high two bits = 01	Critical input—vectored

69.7.3 State logic unit (SLU)

Complex triggers and system performance monitor functions are implemented in the SLU. The SPU creates debug events based upon states in a sequence. A sequence can consist of a single state, or any number of states up to 8. Each state consists of combinational logic allowing AND/OR operations on inputs from the trigger source unit. Single or multiple (up to 8) actions can be triggered by a state machine.

A logic diagram of one SLU state is shown in the following figure.

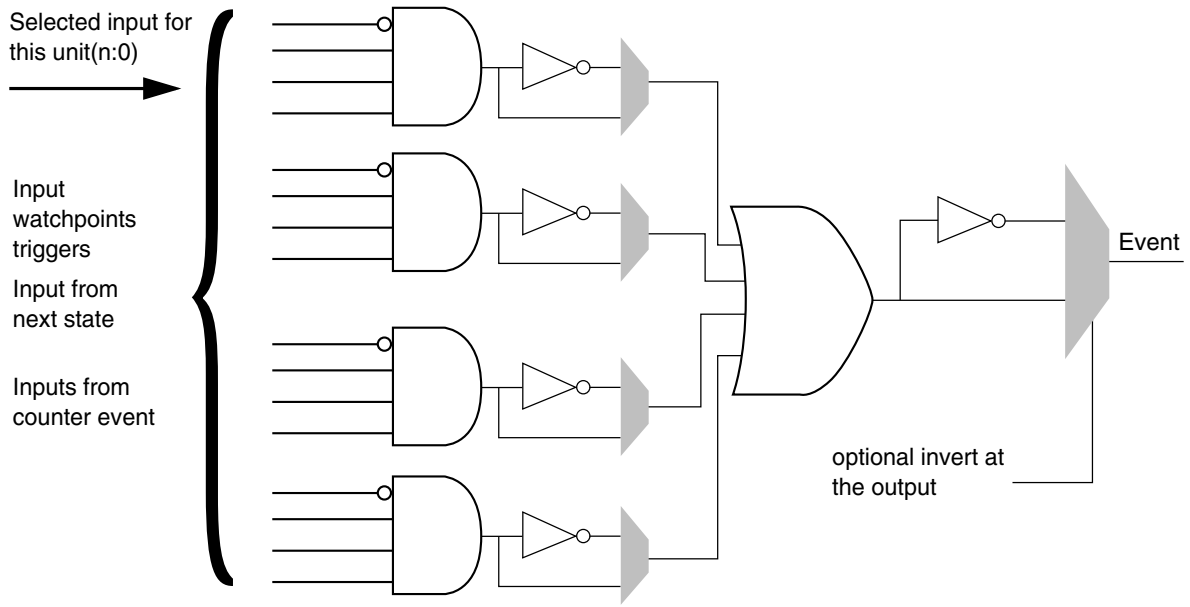


Figure 69-5. SLU block diagram

Each state consists of AND gates, the outputs of which are ORed together.

- Each AND gate can have up to 4 inputs, selected from the trigger source unit by the user.
- The user has an option to invert one input signal into the AND gate.
- The OR gate has a user selectable option to invert the output.
- The output of the OR gate is used as the trigger to move to the next state in the sequence.

The SLUs are responsible for performing operations on the selected input events to produce derived events. Each SLU has a control register that manages how the various inputs are combined to produce a derived event.

An event is triggered from each state based on all the possible selected inputs for each state. This event is used to trigger an Action.

Note

With only one optional inversion at the input of each AND gate and optional inversion at the output of the OR gate, many combinations are not possible.

69.7.4 Sequence formation

Event sequencing can be accomplished with the if-then-else operations. The next event can be fed back as a necessary condition for the current event. For example, event5 can be designated as a necessary condition for event4 in case test condition fails in event5. Event5 can be in turn be designated as necessary for event6 in case test condition passes, and so on. In this way, events can be chained together to create a sequence.

To create complex triggers, a configurable state machine is implemented. This allows the user to join states together with if-then-else operations to create a sequence. The SPU supports up to four simultaneous sequences, however only eight states are supported regardless of the number of active sequences, and each state can only be used in one unique sequence.

For example, the following state configurations are possible:

- 1 active sequence, with 8 states
- 2 active sequences, each with 4 states
- 2 active sequences, the first with 3 states and the second with 5 states
- 4 active sequences, each with 2 states
- 4 active sequences, each with 1 state

Each sequence has the following options which are shown in the following figure:

- The ability to optionally trigger one or more actions based on a true condition from any state in the sequence.
- The ability to optionally trigger one or more actions based on a false condition from any state in the sequence.
- Each state in a sequence has the ability to route to another state on a true condition from the state logic, and the ability to route to a different state based on a false condition from the state logic

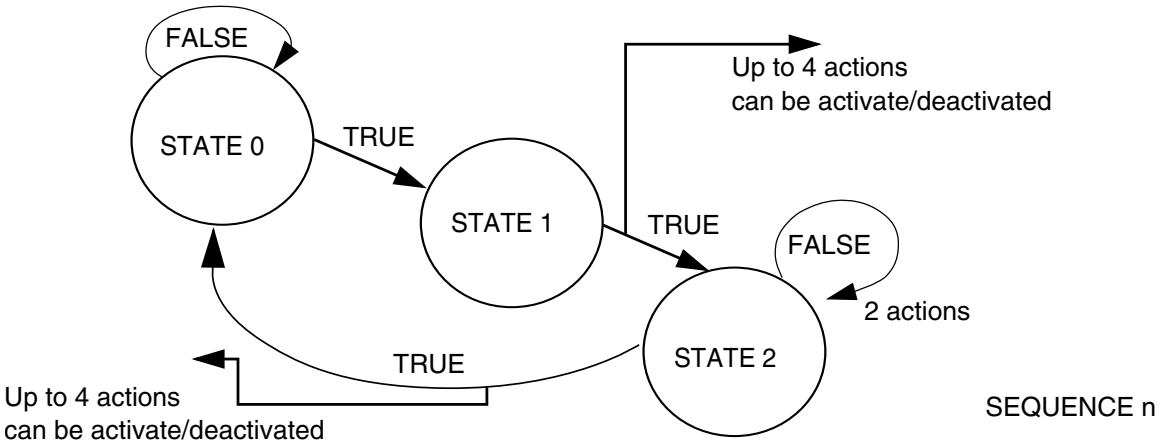


Figure 69-6. Example of a simple configurable if-then-else sequence

69.7.4.1 Complex sequence example

A user can create a complex sequence using five states as shown in the state transition diagram in the following figure.

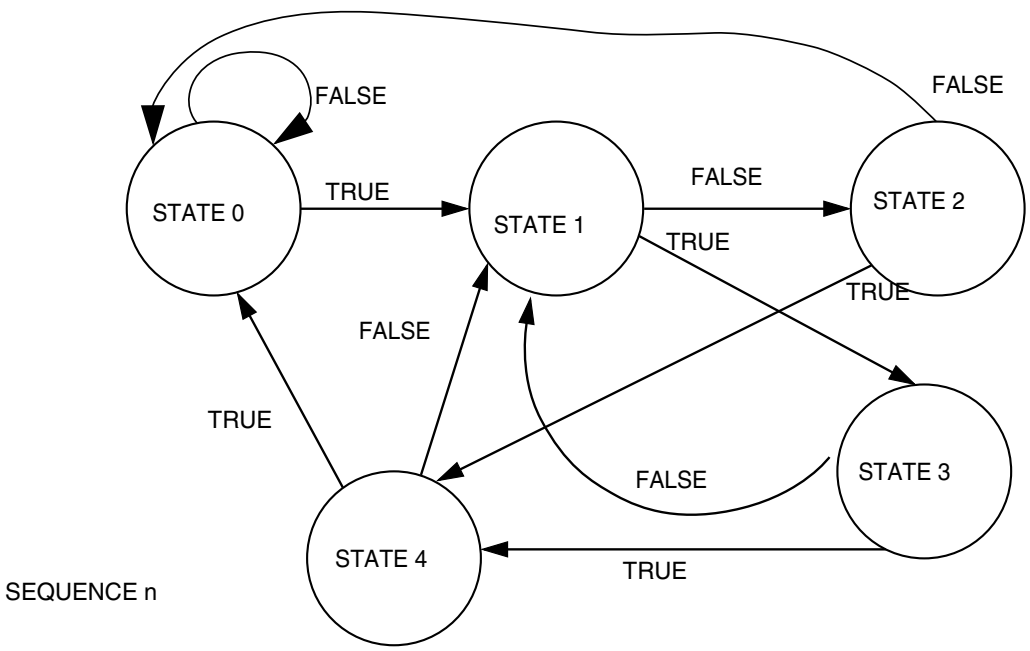


Figure 69-7. Complex sequence example

69.7.4.2 Typical sequence example

For this example, the user would like to create a timestamp in the trace stream each time a function is entered, and upon the 12th time of entering the function, stop the trace signal. This use case can be implemented as follows:

- State 0 takes an instruction address compare as its input. On a match of the instruction address to a specific function, called function_x, the sequence moves to State 1. Upon this action, State0 is configured to insert a timestamp into the trace stream and also increment an SPU counter, say counter[0].
- The sequence is now in State 1. State 1 takes an input from counter[0]. Counter[0] has been pre-configured to create a match event when the counter value is equal to 12.
  - If the value of counter[0] is less than 12, then the sequence moves back to State0.
  - If the value of couner[0] is equal to 12, then the sequence completes and an action from State1 disables the trace signal.

69.7.4.3 Sequence status register description

A status register providing feedback on each sequence is provided to allow debug tools to identify the source of a debug event from the SPU. The status register consists of two field types:

- States—each state has a sticky bit that is set upon a true match of the state logic. This sticky bit is cleared upon a write to this register.
- Sequence status—each sequence has a field indicating the active state of the state machine. This is a read only field.

69.7.5 Performance counters/timers unit

The SPU counters are 32-bit counters that may be used to delay events, count distances, count latencies or count events as shown in the following figure. Therefore, these counter are used as either counters or timers. The selection between counter or timer for each of the counters is done via configuration register. The basic counting operation is straightforward. Once enabled (default is disabled mode), whenever the selected input is driven to a logic 1, the counter increments using the platform clock.

Each of the counter/timer registers are accessible (via the JTAG interface) for the configuration of timer/counter, the compare registers and global SPU timer/counter status registers. These register are accessible via the JTAG interface even when the SPU is in disabled mode.

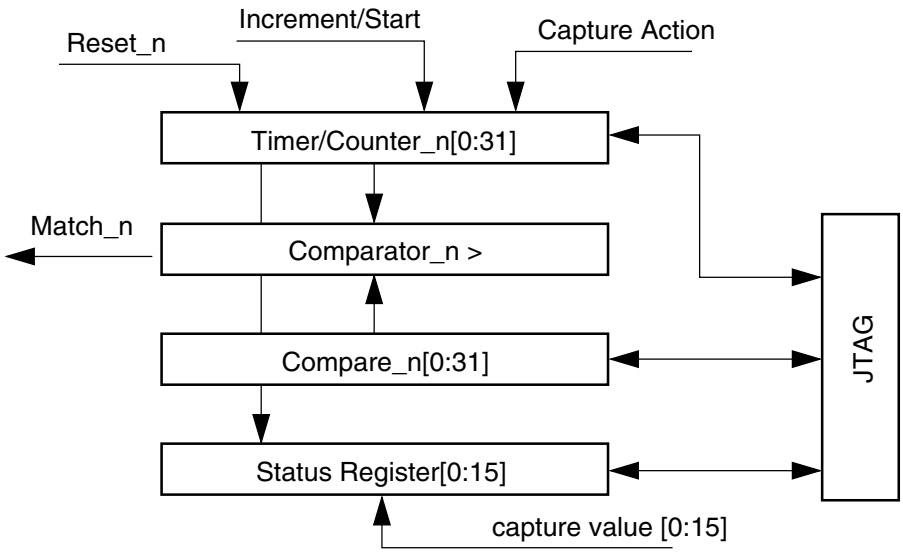


Figure 69-8. Counters/timers overview

69.7.5.1 Counter events and actions

Each counter is capable of detecting one event condition, such as compare value match. The compare value match is used for triggering actions in the SPU. Each counter can be reset in response to a reset action generated from the SPU. Capture registers are available for capturing the counter state in response to a SPU trigger condition. This capture information is updated in the status register which is accessible to users via JTAG interface. These status registers are if type write 1 to clear (W1C).

There are two classes of signals, actions and events:

- Actions—each timer/counter has an interface from the action unit to start/increment and reset as appropriate. The start/increment signal is a single signal, with the operation selected in the SPU configuration registers. The generation of the appropriate count/reset signal is generated in the SPU.

  On an action to insert the timer/counter value into the trace stream, the value stored in the timer/counter is buffered inside the SPU and routed to the appropriate SoC block to insert the value into the trace stream one by one.

- Events—on a match event, whereby the a timer/counter value matches the value of the respective compare register, a signal is sent to the trigger source unit to indicate this.

69.7.5.2 Counter input selection at SPU level

Counter increment inputs derive from the SPU action or direct from the CPU performance measurement counters (CPU_PMC), as shown in the following figure. These inputs include signals from the primary SPU muxes or SPU events of counters.

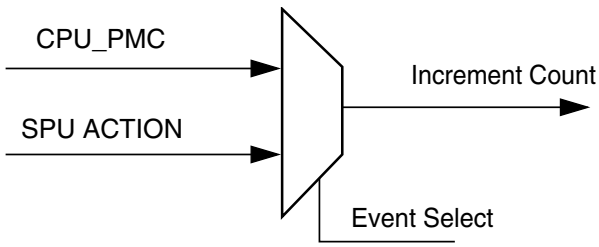


Figure 69-9. Input selection for count increment

69.7.5.3 Counters increment assignment

When configured as a counter, the counter has an increment signal. This can optionally be routed from the SPU action unit, or directly from a CPU performance counter signal. The assignment of which signal is used to increment a counter is handled in the SPU, and is configurable by user. The assignment of increment signals to counters is shown in the following figure.

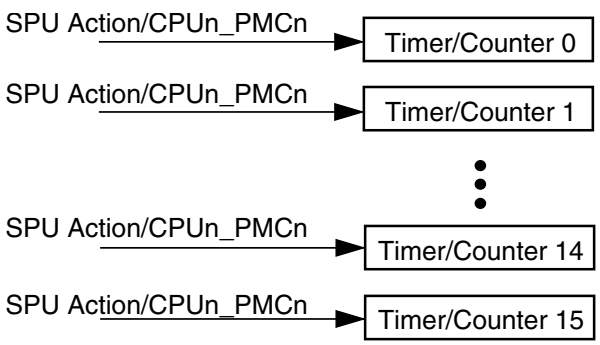


Figure 69-10. SPU counters increment assignment

69.7.5.4 CPU PMC mapping to counters

Each of the CPUs in the system has four performance measurement counters (PMC). The outputs (overflow signals) from these counters (of CPUn) are routed to the SPU debug unit. The counters inside the SPU can be configured to count the pulses from the PMCs. There is a one-to-one mapping between the PMC inputs and the counters that they trigger. Their mapping is shown in the following figure.

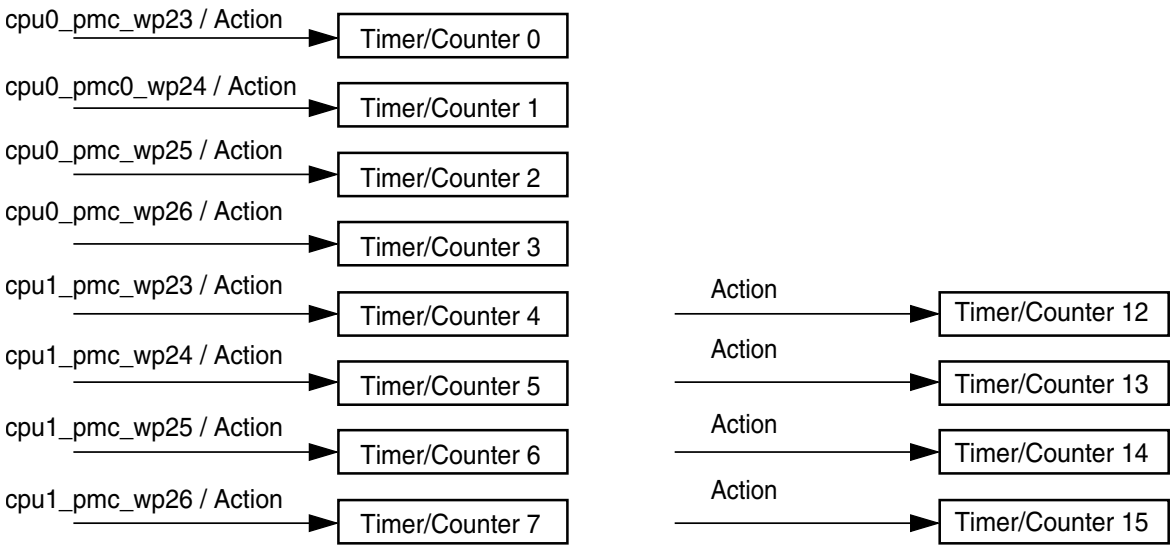


Figure 69-11. CPU PMC mapping inside the SPU unit

69.7.6 Action unit

The action unit block of the SPU is responsible for the generating the action triggers supported by various actions. A state has true and false (if-else-than) condition. Four simultaneous actions can be generated under true/false condition of a particular state. The list of the actions is given in the SPU actions. The actions triggers are mapped to the events by programming the set of SPU group action control for events registers. Each event can be programmed to trigger interrupts, timestamp watch points, trace signals, PMCs, start/reset/increment counters and many other general actions.

69.7.7 Optional trace group

User defined trace group can be defined through the configuration register. Configuration registers registers, defined in Table 69-21, are available where the user can group the different traces from same or different clients.

69.7.8 CKSRC and CKDATA format

The format of the CKSRC is defined for the SPU peripheral (as in-circuit trace client) as shown below:

ICT clients	CKSRC value
SPU counter trace client	b00_0000

Table continues on the next page...

ICT clients	CKSRC value
Reserved	b00_0001 — b11_1111

The format for the CKDATA field for the SPU counter trace client is as shown below:

CKDATA	
CKDATA[CTR_index]	CKDATA[CTR_value]
4 bits	32 bits

Chapter 70
JTAG Master (JTAGM)

70.1 Introduction

The JTAG Master (JTAGM) is a module that is able to act as JTAG master inside the device. The module has a parallel interface that can exchange data with another serial communication module or via customer software.

The data transferred to this module is transformed to produce TCK, TMS, TDI and TRST outputs and to accept TDO inputs. The JTAGM is connected in the device to allow these five signals to connect to the JTAGC as if the JTAG data is coming from an outside tool. The JTAGM generates all required JTAG scan chains to allow software and high speed serial communication access to all JTAG mapped resources.

70.1.1 Overview

The JTAGM is the master to drive JTAG signals from within the device. It has options to receive parallel data from software through the IPS interface . It then sends this data on TDI, TMS and TCK to the DCI. The JTAGM also receives serial data through TDO from the DCI. The JTAGM has the following registers:

- Configuration register
- Status register
- Four data output registers
- RxCRC receive register
- Two data input registers

All these registers are memory mapped and can be accessed by software.

The clocks for the JTAGM are derived from the system clock. The JTAGM also samples the ready signal from the Nexus, to provide a more efficient handshake to the external tool to read and write any memory mapped address location within the device.

This figure shows the block diagram of the JTAGM.

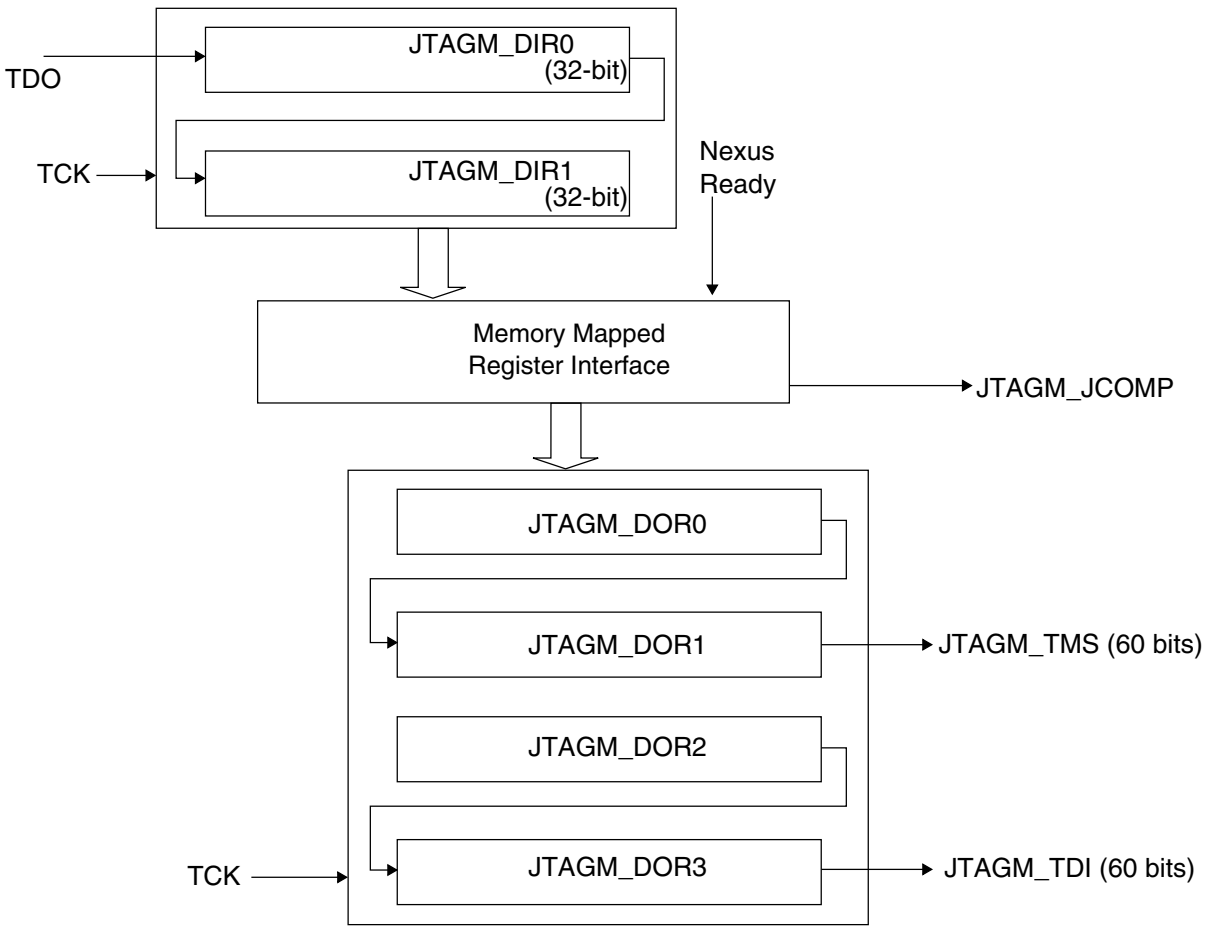


Figure 70-1. JTAGM block diagram

70.1.2 Feature description

The main features of the JTAGM are the following:

- Provides efficient handshake to the external tool to read any memory mapped address location within the device
- Provides software the option to write data for driving JTAG

70.2 Functional description

The JTAGM is simple in concept. TCK is generated from the device system clock. Data is pushed to the modules via a 32-bit wide parallel interface. The data is in the form of 60 bits of TMS data and 60 bits of TDI data. This data is the logical state to be driven onto the TMS and TDI output pins on each of the 60 TCK cycles. During these 60 TCK cycles, the TDO pin is sampled and the 60 bits of sampled data is transferred to the 32-bit wide module interface. This concept allows complete flexible generation on TMS and TDI data and capture of TDO data. The only limitation is that JTAG signals can only be generated in 60 TCK cycles packets. Extra cycles are wasted in idle cycles at the end of a scan chain.

70.2.1 JTAGM Ready signal

The Nexus Read/Write Access (RWA) module allows an external tool to read and write any memory mapped address location within the device via JTAG commands. Read accesses to memory take a finite amount of time and a JTAG read of the data register too soon results in erroneous data being read back. There is a bit in a JTAG register to show the read data is available.

70.2.1.1 Status register ready bit

A bit is provided in the JTAGM status register to indicate the status of the ready signal. To provide this ready signal functionality, the ready signal is monitored. If the ready signal has toggled between the start of the previous 60 TCK shift period and start of the current 60 TCK shift period, the ready bit in the JTAGM status register is set. Otherwise the ready bit in the status register is cleared.

The entire content of the 32-bit status register is appended to the 60 bits of JTAG TDO data captured along with 4 bits of 0-padding and sent as output on the parallel output port as a 96-bit payload on channel A. This provides back to the tool, an indication that the data read in the current transaction is valid.

70.2.1.2 Inter-JTAG frame gap timer

To operate in conjunction with the status register ready bit, a configurable timer is provided to force a programmable gap between the end of one 60-bit JTAG frame and the start of the next. This timer is configured via the JTAGM configuration register from 1 to 64 TCKs in 1 TCK intervals. Furthermore, if the ready signal asserts during this inter-frame gap period, the gap timer is aborted and the next JTAG frame starts immediately.

70.2.2 JTAGM to DCI serial interface

This figure shows the data transfer timings between JTAGM and DCI.

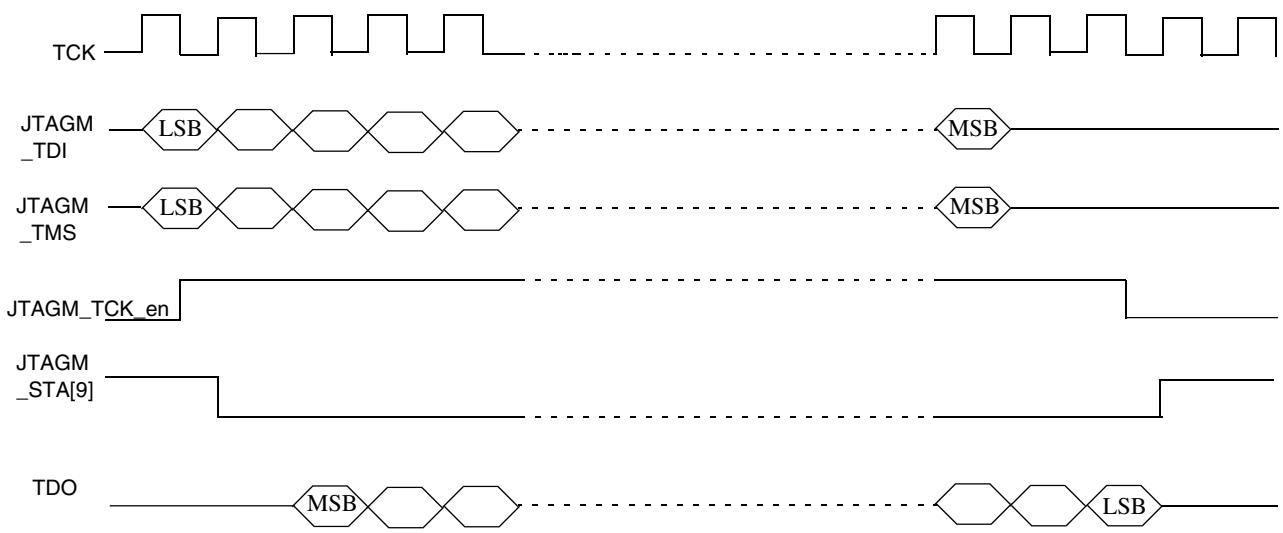


Figure 70-2. JTAGM to DCI serial interface diagram

70.2.3 Software interface

It is possible for software to write the 120 bits of JTAG data to be output and to read the 60 bits of data. The JTAGM module has an IPS interface.

70.3 Modes of operation

There are no special modes of operation. The JTAGM works normally in the debug mode.

70.4 Memory mapped registers

JTAGM supports only 32-bit write accesses to its registers. A byte write access will be converted into a 32-bit write; all other byte values in the register will become 0000_0000b.

JTAGM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (JTAGM_MCR)	32	R/W	0300_0002h	70.4.1/3027
4	Status Register (JTAGM_SR)	32	R/W	0020_0200h	70.4.2/3029
8	Data Out Register 0 (JTAGM_DOR0)	32	R/W	0000_0000h	70.4.3/3030
C	Data Out Register 1 (JTAGM_DOR1)	32	R/W	0000_0000h	70.4.4/3031
10	Data Out Register 2 (JTAGM_DOR2)	32	R/W	0000_0000h	70.4.5/3031
14	Data Out Register 3 (JTAGM_DOR3)	32	R/W	0000_0000h	70.4.6/3032
1C	Data Input Register 0 (JTAGM_DIR0)	32	R	0000_0000h	70.4.7/3032
20	Data Input Register 1 (JTAGM_DIR1)	32	R	0000_0000h	70.4.8/3033

70.4.1 Module Configuration Register (JTAGM_MCR)

This register contains the status bits for the current transfer.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SWRESET	Reserved														
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

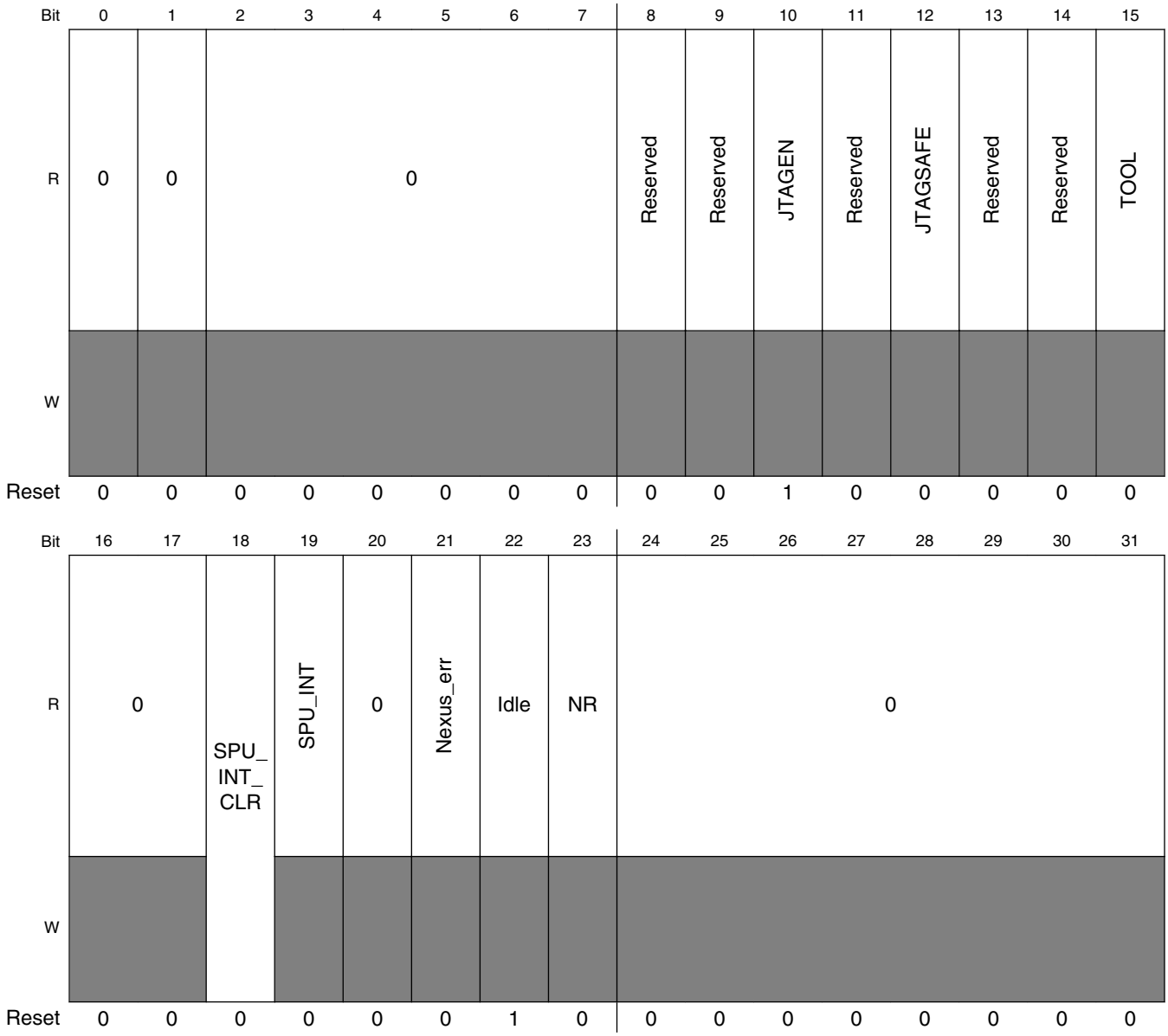
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved		inter_jtag_frame_timer						Reserved	SIE	IIE	TCKSEL			jtagm_JCOMP	DTM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

JTAGM_MCR field descriptions

Field	Description
0 SWRESET	Software Reset. Writing a 1 resets the state machine and counters inside JTAGM. It is a self clearing bit; after being written with a 1, this bit is auto-cleared within 3 TCK cycles. Writing a 0 has no effect.
1–17 Reserved	This field is reserved.
18–23 inter_jtag_frame_timer	TCK delay. 000000 0 TCK delay 000001 1 TCK delay ... ... 111111 63 TCK delay
24 Reserved	This field is reserved.
25 SIE	SPU Interrupt Enable. 0 JTAGM does not generate an interrupt to the CPU 1 JTAGM generates an interrupt to the CPU if the SPU interrupt request is asserted
26 IIE	Idle Interrupt Enable. 0 JTAGM does not generate an interrupt to the CPU upon completion of a 60-bit JTAG transfer 1 JTAGM generates an interrupt to the CPU upon completion of a 60-bit JTAG transfer
27–29 TCKSEL	TCK clock frequency selection. When the JTAGM is enabled, this bit should not be programmed with 000. 000 Reserved 001 TCK is system clock ÷ 2 010 TCK is system clock ÷ 3 011 TCK is system clock ÷ 4 100 TCK is system clock ÷ 5 101 TCK is system clock ÷ 6 110 TCK is system clock ÷ 7 111 TCK is system clock ÷ 8
30 jtagm_JCOMP	JTAG reset. 0 JCOMP low/asserted 1 JCOMP high/not asserted (default)
31 DTM	Data Transfer Mode. 0 Data transferred by software disabled. 1 Data for JTAG transferred by software.

70.4.2 Status Register (JTAGM_SR)

Address: 0h base + 4h offset = 4h



JTAGM_SR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

JTAGM_SR field descriptions (continued)

Field	Description
8 Reserved	This field is reserved. Reserved to 0
9 Reserved	This field is reserved. Reserved to 0
10 JTAGEN	DCI status: DCI JTAG port enable
11 Reserved	This field is reserved. Reserved to 0
12 JTAGSAFE	DCI status: DCI JTAG safe mode
13 Reserved	This field is reserved. Reserved to 0
14 Reserved	This field is reserved. Reserved to 0
15 TOOL	DCI status: DCI tool present, set when a debug tool is connected
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 SPU_INT_CLR	Write 1 clears the SPU_INT. This is a self clearing bit. If SPU_INT is not set then SPU_INT_CLR remains set until SPU_INT is set, and both are cleared.
19 SPU_INT	This bit is set when the SPU interrupt request input rising edge is detected. Writing a 1 to SPU_INT_CLR resets this bit.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 Nexus_err	Nexus error bit. It is set when there is an error, else it is 0.
22 Idle	Idle bit. Idle bit is cleared while a JTAG frame is in progress. This bit is set to 1 when a JTAG frame ends. Writing a 1 to the Idle bit clears the Idle_interrupt in SW Mode. A read always gives the JTAG frame status but does not give the write value.
23 NR	Status of Ready bit from Nexus RWA. This bit is automatically cleared when a new JTAG message is started. 0 RDY has not asserted indicating Nexus RWA is not ready 1 RDY has asserted indicating a Nexus RWA has completed bus access and is ready for the data register to be read
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

70.4.3 Data Out Register 0 (JTAGM_DOR0)

Address: 0h base + 8h offset = 8h

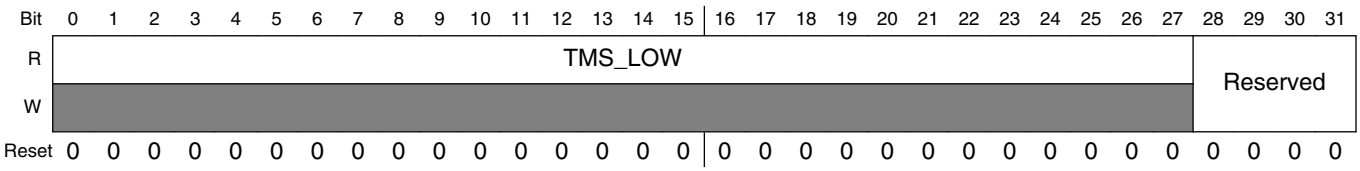
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TMS_HIGH																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

JTAGM_DOR0 field descriptions

Field	Description
0–31 TMS_HIGH	Higher word of data for TMS, bits 59-28. Writable by software only when DTM = 1

70.4.4 Data Out Register 1 (JTAGM_DOR1)

Address: 0h base + Ch offset = Ch

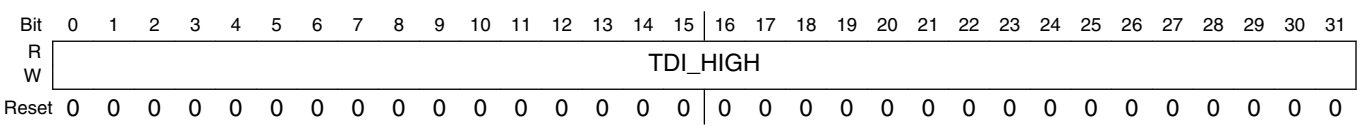


JTAGM_DOR1 field descriptions

Field	Description
0–27 TMS_LOW	Lower word of data for TMS, bits 27-0. Writable by software only when DTM = 1. TMS_LOW[0] is shifted out first
28–31 Reserved	This field is reserved.

70.4.5 Data Out Register 2 (JTAGM_DOR2)

Address: 0h base + 10h offset = 10h

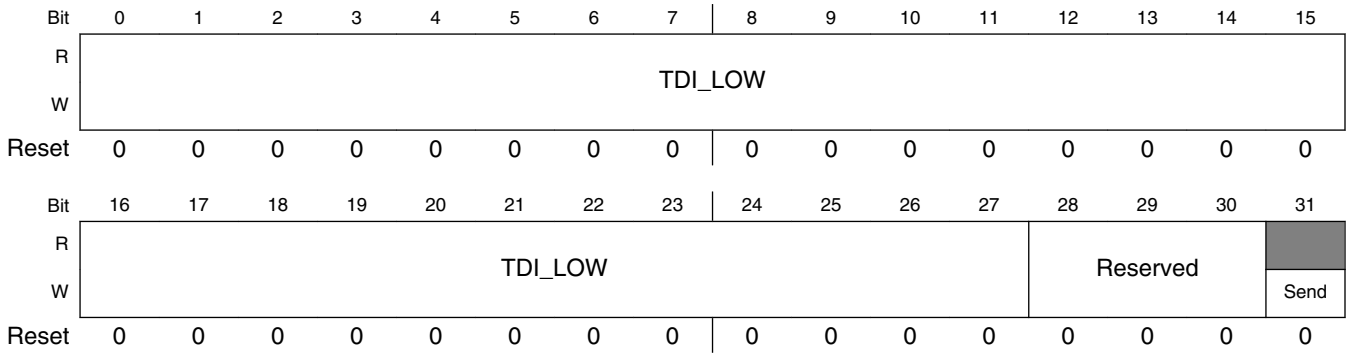


JTAGM_DOR2 field descriptions

Field	Description
0–31 TDI_HIGH	Higher word of data for TDI, bits 59-28. Writable by software only when DTM = 1

70.4.6 Data Out Register 3 (JTAGM_DOR3)

Address: 0h base + 14h offset = 14h

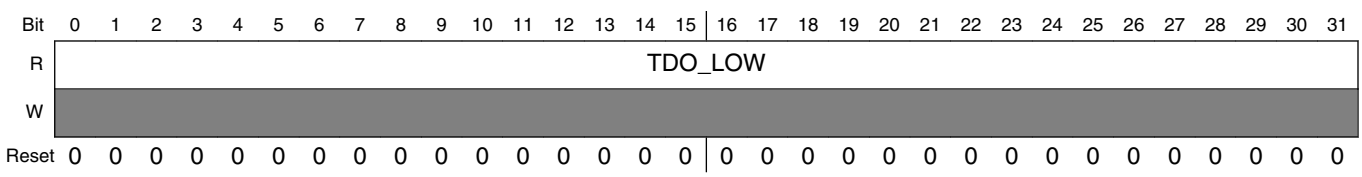


JTAGM_DOR3 field descriptions

Field	Description
0–27 TDI_LOW	Lower word of data for TDI, bits 27-0. Writable by software only when DTM = 1. TDI_LOW[0] is shifted out first
28–30 Reserved	This field is reserved.
31 Send	Send bit. When this bit is set, data is sent to the DCI. It is a self-clearing bit and is always read as 0. Writable by software only when DTM = 1

70.4.7 Data Input Register 0 (JTAGM_DIR0)

Address: 0h base + 1Ch offset = 1Ch

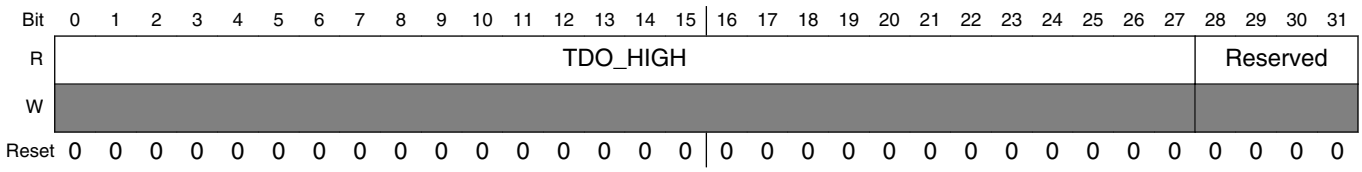


JTAGM_DIR0 field descriptions

Field	Description
0–31 TDO_LOW	Lower word of data received on TDO, bits 0-31

70.4.8 Data Input Register 1 (JTAGM_DIR1)

Address: 0h base + 20h offset = 20h



JTAGM_DIR1 field descriptions

Field	Description
0–27 TDO_HIGH	Higher word of data received on TDO, bits 32-59
28–31 Reserved	This field is reserved.

Chapter 71
Development Trigger Semaphore (DTS)

71.1 Introduction

The Development Trigger Semaphore (DTS) module enables software to signal an external tool by driving a persistent (affected only by reset or an external tool) signal on an external device pin. There are a variety of ways this module can be used, including as a component of an external real-time data acquisition system.

Note

When used as a component of a triggered data acquisition system, Nexus read/write access is different than the data acquisition protocol defined in the IEEE-ISTO 5001-2003 or IEEE-ISTO 5001-2012 Nexus standards, which use the Nexus Auxiliary port.

71.2 Overview

The Development Trigger Semaphore (DTS) module consists of registers and a small amount of combinational logic to generate an output signal—DTS Trigger Output (DTO). The registers are as follows:

- DTS_SEMAPHORE register—Any bit in this 32-bit register, when set to a value of logic '1', causes the DTS module output signal to be asserted if the corresponding DTS_EN bit is set in the DTS_ENABLE register. This enables an external tool to detect up to 32 signals from the application software. In an application, each bit is generally associated with a specific data set. Only the processor core and DMA module can set bits in this register. The bits can only be cleared by a tool access via Nexus Read/Write Access.

- DTS_SEMAPHORE_B register—Any bit in this 32-bit register, when set to a value of logic '1', causes the DTS module output signal to be asserted if the corresponding DTS_EN_B bit is set in the DTS_ENABLE register. This enables an external tool to detect an additional 32 signals from the application software. In an application, each bit is generally associated with a specific data set. Only the processor core and DMA module can set bits in this register. The bits can only be cleared by a tool access via Nexus Read/Write Access.
- DTS_STARTUP register—This register provides a mechanism for the external tool to notify software running on the CPU that the tool is connected and can provide information about either the type of tool or options that can be used by the software.
- DTS_ENABLE register—This register provides an enable/disable capability for the DTS feature.

The architecture is shown in the following figure.

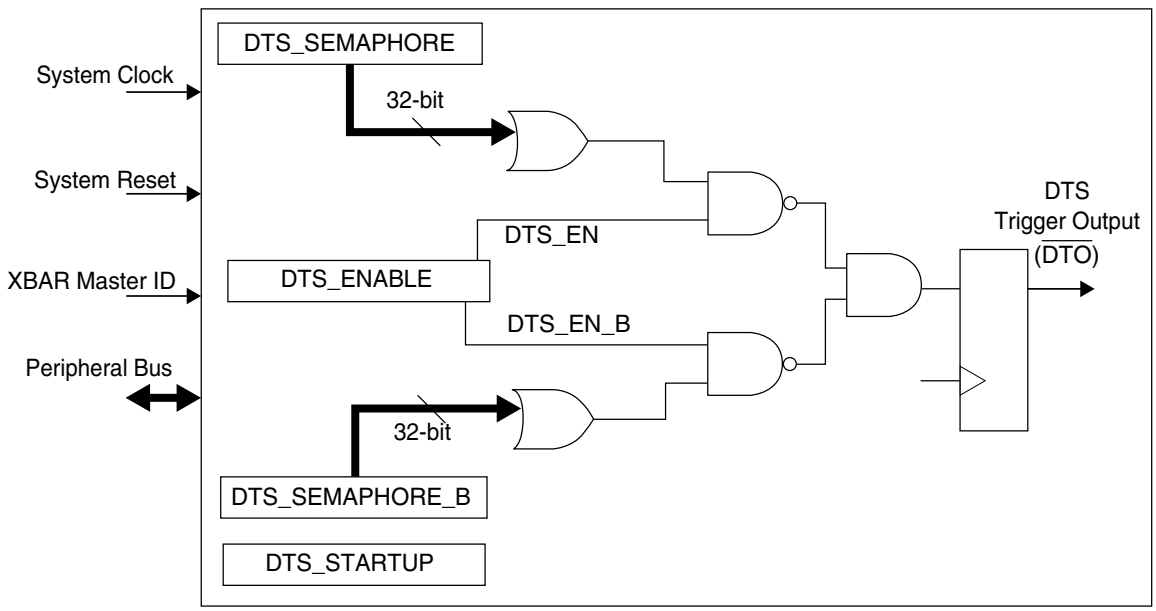


Figure 71-1. DTS block diagram

The DTS Trigger Output (DTO) signal is connected to one of the EVTO inputs of the Debug and Calibration Interface (DCI). The other EVTO inputs are connected to the other Nexus modules in the device. DTO is asserted when any bit in a semaphore register is set and the DTS function of that register is enabled.

Note

When the DTS module is enabled (DTS_ENABLE[DTS_EN] = 0b1 or DTS_ENABLE[DTS_EN_B] = 0b1), all other Nexus EVTO functions associated should be disabled by the tool and

EVTO becomes the DTO. Unlike the EVTO function that only asserts for one clock, the DTO function remains asserted until the tool reads the semaphore register that is the source of the assertion, clearing the register's contents.

The following figure shows an example of the chain of events that begins with setting of a bit in the DTS_SEMPAHORE register and the clearing of the register caused by a Nexus read.

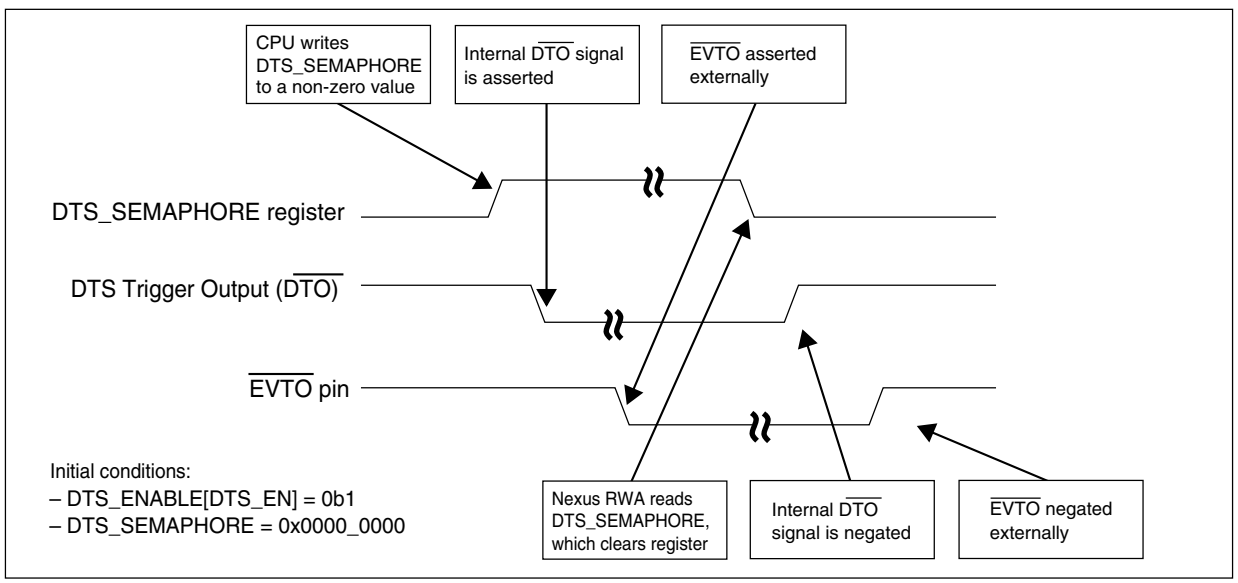


Figure 71-2. DTO event sequence example

71.3 DTS device connections

The following figure shows the DTS device connections. The DTS module connects to the Peripheral Bridge (PBRIDGE) for access to the registers. The PBRIDGE is connected to the device modules through a slave port of the Crossbar bus interface (XBAR).

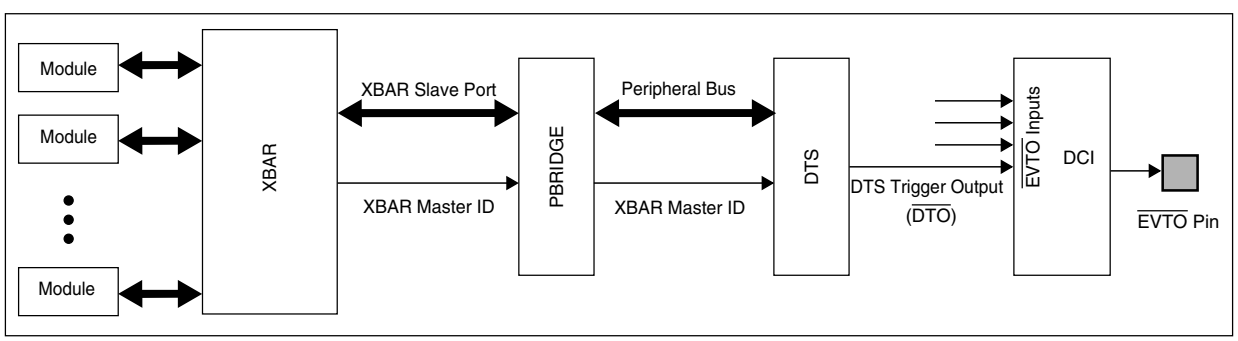


Figure 71-3. DTS device connections

The registers have limited access as described in DTS register access. Access is based on the XBAR Master ID of the accessing module. Writing to the semaphore registers is limited to the cores and the eDMA module and is restricted to only setting bits. Only an access via a Nexus Read/Write Access from an external tool can clear bits in the semaphore registers (semaphore register bits are cleared automatically when read). Similarly, the DTS_ENABLE and DTS_STARTUP registers can only be written via a Nexus Read/Write Access.

Note

Nexus Read/Write Accesses use the core load/store bus to perform accesses, but Nexus accesses have a different Master ID than normal core load/stores.

71.3.1 DTS register access

A summary of accesses to all DTS registers by bus masters is provided in the following table. Only proper 32-bit accesses are valid. The effects of write accesses that are not 32 bits are not defined.

Table 71-1. DTS register access effects

Register	32-bit read				32-bit write			
	RWA[1, 1]	Core	eDMA	Other crossbar masters	RWA [1]	Core	eDMA	Other crossbar masters
DTS_ENABLE	Data	Data	Data	Data	Data	No effect	No effect	No effect
DTS_STARTUP	Data	Data	Data	Data	Data	No effect	No effect	No effect
DTS_SEMAPHORE	Data and Clear[2, 2]	Data	Data	Data	No effect	Bit OR	Bit OR	No effect
DTS_SEMAPHORE_B	Data and Clear[2]	Data	Data	Data	No effect	Bit OR	Bit OR	No effect

1. Nexus Read/Write access via an external tool
2. A read of the semaphore registers by Nexus Read/Write Access module is destructive and clears all bits in the register

Access to DTS module registers is controlled based on the XBAR Master ID of the accessing module.

Note

The XBAR Master ID should not be confused with the Master Port number of the XBAR. See the Crossbar Switch (XBAR) chapter for details.

Tools must access the DTS registers through the Nexus Read/Write Access mechanism of a core. External tool accesses through either core appear as if the access is via the core and therefore do not have the same level of access as a Nexus Read/Write Access.

71.4 Memory mapped registers

Only certain types of accesses are allowed to the DTS registers. See the description for each register to determine the accesses that are allowed.

DTS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Output Enable Register (DTS_ENABLE)	32	R/W	0000_0000h	71.4.1/3039
4	Startup Register (DTS_STARTUP)	32	R/W	0000_0000h	71.4.2/3040
8	Semaphore Register (DTS_SEMAPHORE)	32	R/W	FFFF_FFFFh	71.4.3/3041
C	Semaphore Extension (DTS_SEMAPHORE_B)	32	R/W	FFFF_FFFFh	71.4.4/3042

71.4.1 Output Enable Register (DTS_ENABLE)

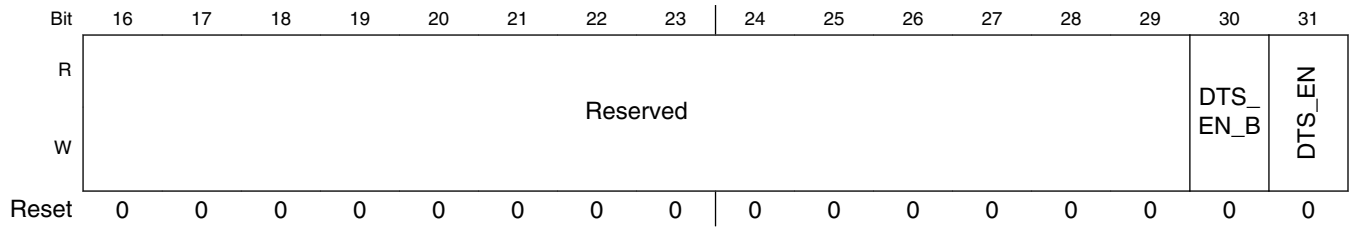
The DTS_ENABLE register controls the DTS Trigger Output ($\overline{\text{DTO}}$) and whether $\overline{\text{DTO}}$ is active on the $\overline{\text{EVTO}}$ output pin of the device. The DTS_ENABLE register can be read by the core, but can only be written by a Nexus Read Write Access (RWA).

NOTE

Access to the DTS_SEMAPHORE, DTS_SEMAPHORE_B, and DTS_STARTUP registers are unaffected by the state of this register.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



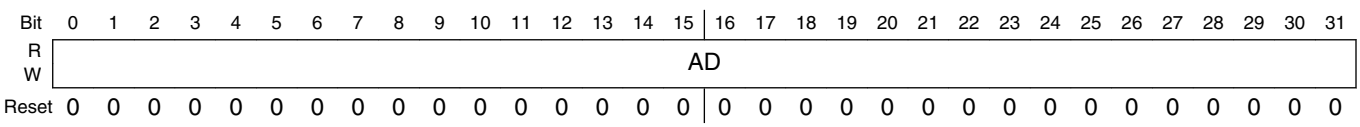
DTS_ENABLE field descriptions

Field	Description
0–29 Reserved	This field is reserved.
30 DTS_EN_B	DTS Enable B. Controls whether the $\overline{\text{DTO}}$ signal is routed to the $\overline{\text{EVTO}}$ pin. The DTS Enable bit is cleared by a device reset (either the assertion of the external $\overline{\text{RESET}}$ or by an internally generated reset). A Nexus reset does not change the state of this register. 0 DTS output disabled. Any bit set in the DTS_SEMAPHORE_B register does not assert the DTS trigger output signal. 1 DTS output enabled. Any bit set in the DTS_SEMAPHORE_B register asserts the DTS trigger output signal ($\overline{\text{DTO}}$).
31 DTS_EN	DTS Enable. Controls whether the $\overline{\text{DTO}}$ signal is routed to the $\overline{\text{EVTO}}$ pin. The DTS Enable bit is cleared by a device reset (either the assertion of the external $\overline{\text{RESET}}$ or by an internally generated reset). A Nexus reset does not change the state of this register. 0 DTS output disabled. Any bit set in the DTS_SEMAPHORE register does not assert the DTS trigger output signal. 1 DTS output enabled. Any bit set in the DTS_SEMAPHORE register asserts the DTS trigger output signal ($\overline{\text{DTO}}$).

71.4.2 Startup Register (DTS_STARTUP)

The DTS_STARTUP register is used for tool detection and startup information exchange between external data acquisition tool and the embedded controller. DTS_STARTUP register can be read by the core, the eDMA module and Nexus but can only be updated by a Nexus Read Write Access (RWA).

Address: 0h base + 4h offset = 4h



DTS_STARTUP field descriptions

Field	Description
0–31 AD	Application Dependent register bits. The bits have no defined meaning to the microcontroller. They are used to by an external tool to pass information (for example, application options and status) to application

DTS_STARTUP field descriptions (continued)

Field	Description
	software running on target microcontroller at startup time. Use a Nexus RWA 32-bit write access to update the contents of this register. A device reset (either from the $\overline{\text{RESET}}$ pin or an internally generated reset) clears all bits in the register. A Nexus reset does not change the contents of the register.

71.4.3 Semaphore Register (DTS_SEMAPHORE)

The DTS_SEMAPHORE register is used by software to assert the $\overline{\text{DTO}}$ signal on the device $\overline{\text{EVTO}}$ pin. A one in any bit of this register causes the $\overline{\text{DTO}}$ signal on the $\overline{\text{EVTO}}$ pin to be driven low. The intended use of this register is for the $\overline{\text{DTO}}$ signal to notify tools that data is available. Individual bits are used to identify the specific data.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ST																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

DTS_SEMAPHORE field descriptions

Field	Description
0–31 ST	Semaphore Trigger. When a core or eDMA writes a logical '1' to a bit, the bit is set. A write of '0' by the core or DMA does not change the state of the bit. • All register bits are set to '1' by a device reset. • A Nexus reset does not change the state of this register. • The register can be accessed, with restrictions, by any core, DMA or any Nexus RWA. • For the core or DMA, only 32-bit write or read accesses are valid. • A core or DMA valid read access returns the current value of the register and leaves the register unchanged. 0 No flag 1 Flag is set

71.4.4 Semaphore Extension (DTS_SEMAPHORE_B)

The DTS_SEMAPHORE_B register is used by software to assert the $\overline{\text{DTO}}$ signal on the device $\overline{\text{EVTO}}$ pin. A one in any bit of this register causes the $\overline{\text{DTO}}$ signal on the $\overline{\text{EVTO}}$ pin to be driven low. The intended use of this register is for the $\overline{\text{DTO}}$ signal to notify tools that data is available. Individual bits are used to identify the specific data.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ST_B																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

DTS_SEMAPHORE_B field descriptions

Field	Description
0–31 ST_B	Semaphore Trigger Extension. When a core or eDMA writes a logical '1' to a bit, the bit is set. A write of '0' by the core or DMA does not change the state of the bit. • All register bits are set to '1' by a device reset. • A Nexus reset does not change the state of this register. • The register can be accessed, with restrictions, by any core, DMA or any Nexus RWA. • For the core or DMA, only 32-bit write or read accesses are valid. • A core or DMA valid read access returns the current value of the register and leaves the register unchanged. 0 No flag 1 Flag is set

71.5 Example application

The calibration process of a new engine requires real-time access to calibration tables and the ability to update the tables in real-time The DTS module enables this capability by enabling software to assert a signal to an external device pin to notify an external tool that data is available. The tool can then retrieve the data.

In this type of application the DTS_SEMAPHORE register and DTS Trigger Output (DTO) signal provide a mechanism to notify the calibration tool that the calibration variable or variables (or sets of measurements), up to 32, have been updated with new values and are available for the tool to access.

Note

It is the user's responsibility to ensure that the tool has time to retrieve the data prior to that particular trigger being set a second time. It is also permissible to have multiple triggers active at the same time or for a second trigger to be set before a previous trigger has been serviced, as long as it is not the same trigger (unless it is acceptable to the tool to not receive every data set).

The following table shows an example DTS startup sequence for an external real-time data acquisition system. The startup and synchronization sequence can be as simple or as complicated as the need requires. However, a typical startup sequence is as follows:

1. The DTS_STARTUP register is cleared by a power on reset or any CPU reset.
2. The tool writes a non-zero value to the DTS_STARTUP register.
3. The CPU (user application software) then reads the value of the DTS_STARTUP register. Based on this value, different initialization options can be selected. The bits can be used for any application specific definitions.
4. Since the DTS_SEMAPHORE register is cleared when the tool reads the current value. The tool should perform all necessary initialization before reading this register. The application software can then check that the DTS_SEMAPHORE register was cleared by the tool, to determine that it is safe to start using it for its intended raster trigger semaphore function.
5. An optional hand shake from the CPU can be used to inform the tool that the user software has detected that the tool is attached and the CPU has performed the proper initialization for the tool by writing a predefined value to the DTS_SEMAPHORE register (the example shown in the figure above uses 0xAAAA_AAAA—all A's was used since it is unrealistic that 16 channels could be enabled very quickly after start up after a reset).

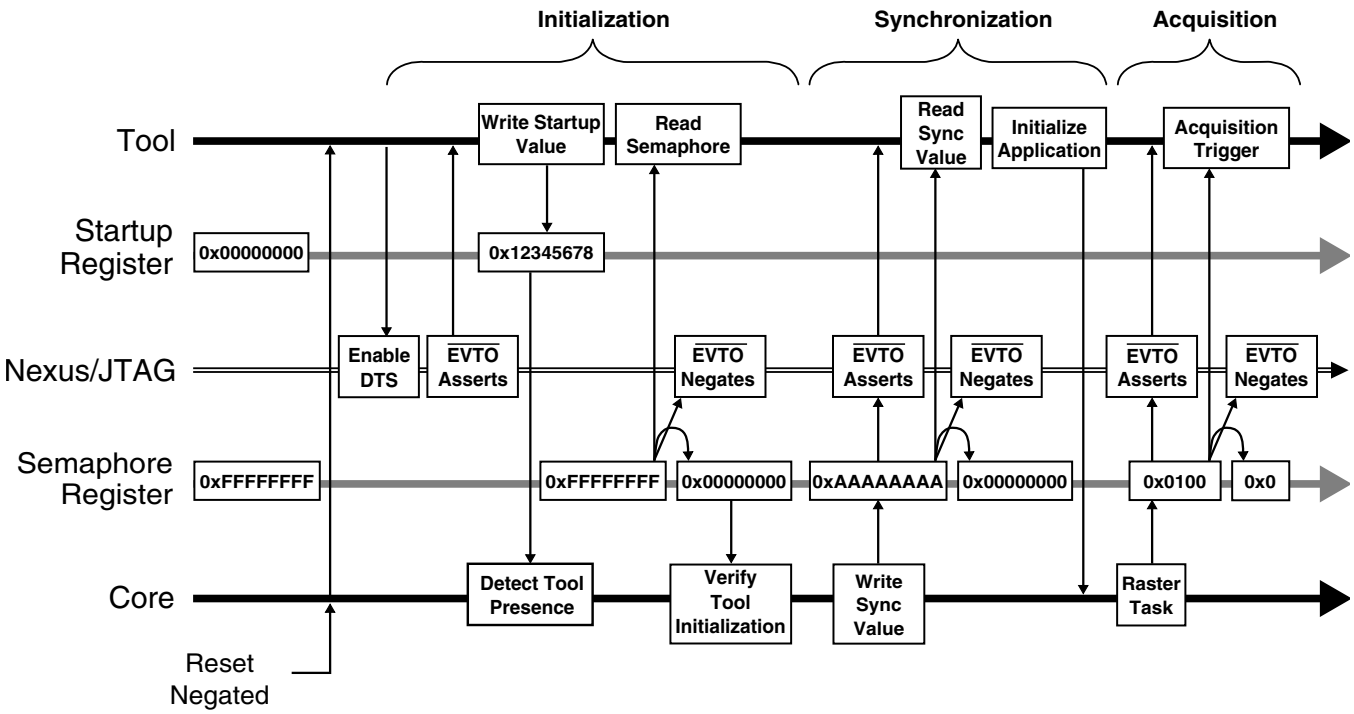


Figure 71-8. DTS startup sequence example

Chapter 72
Nexus Aurora Router (NAR)

72.1 Nexus Aurora Router (NAR) chip-specific information

72.1.1 NAR DID value

The NAR Device ID (DID) value for Cut 2.0B is 0x1834_601D.

72.2 Introduction

On the device, there are multiple blocks that require development interface support. The blocks that are Nexus-compliant (based on the IEEE-ISTO 5001 standard) are expected to interface with the development tool through a dedicated high-bandwidth debug port. The Nexus Aurora Router (NAR) controls the usage of the output port in a manner that allows all the individual development interface blocks to share the port, and appear to the development tool to be a single block.

Top-level architecture is shown in the following figure. Multiple clients send messages by sharing the high-bandwidth port through the NAR. For a list of the specific clients that are connected to the NAR, see the Calibration and Debug configuration chapter which describes how the modules are configured.

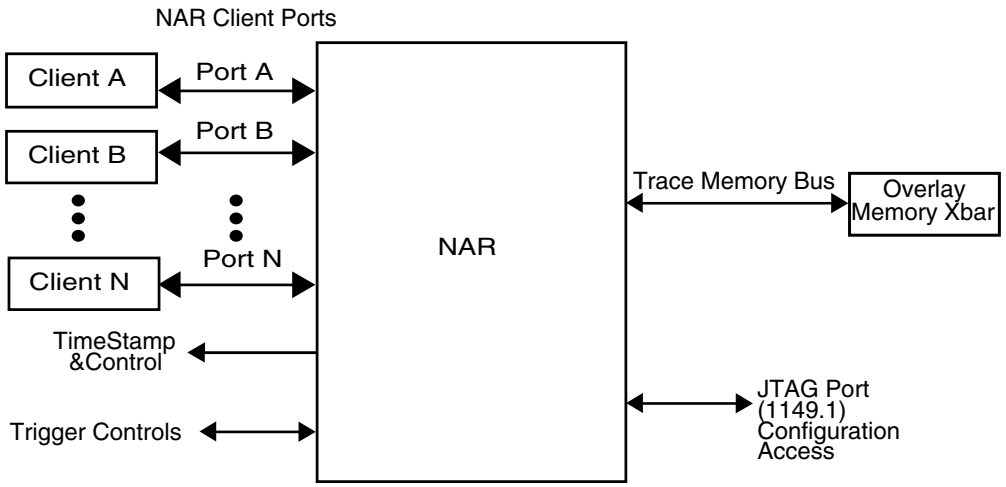


Figure 72-1. NAR top level block diagram

The trace memory bus port provides added flexibility to the channel trace data to memory on the Nexus Aurora crossbar. Setup and configuration of the NAR is performed by accessing the configuration registers via a JTAG (1149.1) port.

Each Nexus client has a dedicated receive queue to allow seamless switching between sources at Nexus message boundaries. Incoming Nexus data is parsed into the receive queues so that end-of-message can be determined. Messages are then transferred into the main queue. An arbitration block determines the order in which clients are serviced, using an optional client request level bus to prioritize the data (lacking request level data, a simple round-robin algorithm is used). The NAR has two output ports: the high bandwidth debug port (Aurora out) and the trace memory bus. If both output ports are enabled at the same time, the main queue is logically partitioned inside NAR. The Production Device (PD) does not support partitioning of the main queue; therefore, only one of the ports can be active at a time. Data going through the Aurora out port or the Trace port goes through an additional translation block to put the data in the correct format. The actual Aurora formatting is done in the NAL. Additional blocks for message generation and timestamp generation are described in more detail in the following sections.

The NAR supports the following client interfaces:

- Legacy interface to support existing client interface protocol through a conversion gasket logic with clock domain transfer
- New client interface with clock domain transfer

72.3 Overview

The NAR serves as an arbiter for the high-bandwidth debug port (HBDP). The NAR pulls Nexus messages from multiple clients, queues the data, and forwards it (based on the user-controlled configuration) to the high-speed Aurora Out port and/or the Trace port. The trace memory bus controller allows the NAR to transmit data via the normal system interface. NAR control and configuration registers are accessed via JTAG (IEEE 1149.1) port.

72.4 Features

The NAR supports the following features:

- Connects 7 client(s) with dedicated receive queue(s). Clients can transmit data as long as there is space in its associated receive queue, but only the active port receive queue empties into the main queue.
- Legacy client interface for backward compatibility
- Aurora output data format for dedicated high-bandwidth serial transmission
- Aurora output format support at client interface
- Trace Memory port (64-bit address and data) can be used to write Nexus messages directly to external overlay memory. Block transfer mode provides highest-bandwidth option.
- Configurable Start Address and Range Selection for Trace Memory port write operation provides flexibility to choose Overlay memory Section and range to store Trace data.
- Time-stamping provides a central synchronization resource for message ordering. Gray coded timer output for clients working in different clock domians.
- Re-allocatable receive queues allow queue-space of an inactive client to be reassigned to an active client
- Stall detection and handling allows NAR to continue when a client stops sending data in the middle of a message

- Event generation on External Trace Memory (overlay) programmable full
- Suppress mode can be used to shut down message delivery and relieve back-pressure to clients without turning off Nexus messaging in the clients.

72.5 Register definition

The NAR configuration registers provide configuration and control for all NAR operations. They are accessible through the JTAG (IEEE 1149.1) port. Addressing (within the local NAR address space) for the NAR registers is shown in the following table. If the NAR clock is turned off, the NAR registers cannot be accessed.

Table 72-1. NAR registers

Register name	Read/Write	JTAG index (NAR relative)
NAR_CR—NAR Control Register	R/W	0
NAR_ST—NAR Status Register	R	1
NAR_SFR—Source Filter Register	R/W	2
NAR_TFR—Type Filter Register	R/W	3
NAR_TBALO—Trace memory port base address, low	R/W	4
NAR_TBAHI—Trace memory port base address, high	R/W	5
NAR_TCR—Trace Memory bus control register	R/W	6
NAR_STCR—Suppress Trigger Control Registers 1–4	R/W	7 (Trigger 1) 8 (Trigger 2) 9 (Trigger 3) A (Trigger 4)
NAR_CSSR–Client Suppress Status Register	R	B
NAR_CDR—Client Disable Register	R/W	C
NAR_AHFPAR—Trace memory fill level and partition configuration	R/W	E

72.5.1 NAR control register (NAR_CR)

The NAR control register provides all the main control and enable bits for the NAR and its output ports. The following table shows the format of the NAR_CR.

Register index: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NEN	NCD	TEN	TRS	TSG		SED	SRD	TBW	AQP		0	0	0	0	0

Table continues on the next page...

W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WPE			0	0	0	0	0	ALT		0	AFA	0	0	PUS	NSR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_CR is described in the following table.

Table 72-2. NAR_CR field descriptions

Field	Description
31 NEN	NAR Enable. 0 NAR is disabled unless HBDP is independently enabled. See Disabled for more information. 1 NAR is enabled.
30 NCD	NAR Communication Disable. 0 NAR monitors clients for activity 1 NAR communication with clients is disabled
29 TEN	Timestamp Enable. 0 Timestamp packet is not included in messages 1 Timestamp packet is included in messages from all client including NAR generated messages
28 TRS	Timestamp Reset. 0 to 1 transition: reset timestamp counter in NAR
27–26 TSG	Timestamp Granularity (Mode). **Note:** Some clients may only support timestamps on every message. 00 Add timestamp to every message 01 Add timestamp to every 4th message 10 Add timestamp to every 16th message 11 Add timestamp to every 32nd message
25 SED	Stall Error Disable. 0 NAR injects dummy complete message when active client stalls 1 NAR does not inject dummy complete message
24 SRD	Stall Recovery Disable. 0 Stalled receive queue resumes operation when its client comes back on line 1 Stall error recovery is disabled
23 TBW	Tracebuffer Wrap. 0 NAR does not capture new data when main queue is full 1 NAR begins overwriting main queue from the beginning upon queue overflow

Table continues on the next page...

Table 72-2. NAR_CR field descriptions (continued)

Field	Description	
22–21 AQP	Alternate Queue Partition.	
	00	Queue partitioning is disabled
	01	Reserved
	10	Reserved
	11	Reserved
20	Reserved and must be written with 0	
19–16	Reserved and must be written with 0s	
15–13 WPE	Watchpoint Enable	
	000	NAR watchpoint messaging is disabled
	xx1	NAR generates a watchpoint message upon a sync event
	x1x	NAR generates a watchpoint message upon a trace memory bus or tracebuffer block full/wrap event
	1xx	NAR generates a watchpoint message upon a suppress-mode event
12	Reserved	
11	Reserved. This bit must be written with 0.	
10–9	Reserved and must be written with 0s.	
8	Reserved	
7–6 ALT	Alternate Port. Defines which port is the target for non-filtered messages; main port in single-port applications.	
	00	No alternate port. Running in trace-buffer mode.
	01	Aurora out is the ALT
	10	Trace memory bus is the ALT
	11	Reserved
5	Reserved	
4 AFA	Automatic Flush Accumulators.	
	0	Output queue accumulators do not flush stranded beats
	1	Accumulators automatically flush stranded beats
3	Reserved. This bit must be written with 0.	
2	Reserved and must be written with 0.	
1 PUS	Power-Up Suppressed.	
	0	NAR powers up active
	1	NAR powers up in global suppressed mode
0 NSR	NAR Soft Reset.	
	1	Resets all configuration registers and sets queue address pointers to the top of their respective queues. Also resets itself after a single-cycle delay. (This does not affect other blocks)

72.5.2 NAR status register (NAR_ST)

The NAR_ST register provides information on NAR activity and queue status, as well as HBDP status and errors. It is a read-only register. The following table shows the format of the NAR_ST.

Register index: 1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NEN	NUM			NCE				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HPR	HPE	MPE	MOV	0	TOV	QWA									
W		w1c	w1c	w1c		w1c										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

The NAR_ST register is described in the following table.

Table 72-3. NAR_ST field descriptions

Field	Description	
31 NEN	NAR Enabled.	
	0	NAR is disabled and inactive
	1	NAR is enabled with clocks on
30–28 NUM	NAR Mode.	
	000	NAR is in disabled mode
	001	NAR is in global suppressed mode
	010	NAR is in communication-disabled mode
	011	NAR is in tracebuffer mode
	100	NAR is in active transmit mode
	101	Reserved
	110	Reserved
	111	NAR is in run-control mode
27–24 NCE	NAR Config Error. See Table 72-17 for coding	
23–16	Reserved	
15 HPR	HBDP Ready.	
	0	HBDP (Aurora Out port) is not ready
	1	HBDP is ready and accepts data for transmission
14 HPE	HBDP Error.	
	0	No HBDP error
	1	Error condition exists on HBDP

Table continues on the next page...

Table 72-3. NAR_ST field descriptions (continued)

Field	Description	
13 MPE	Trace memory bus Error.	
	0	No Trace memory bus error
	1	Error condition detected on trace transaction
12 MOV	Trace memory bus block overflow.	
	0	No Trace memory bus overflow
	1	Targeted external memory block is full. Trace memory bus writes are halted (or wrapped).
11	Reserved	
10 TOV	Tracebuffer Overflow.	
	0	No tracebuffer overflow
	1	The tracebuffer has overflowed/wrapped at least once
9–0 QWA	Queue Write Address. Current value of the output queue write pointer (last valid write + 1)	

72.5.3 Message filtering registers

The two output ports (HBDP and trace memory bus) are independently controllable. In the BD NAR, two ports can be active simultaneously. When multiple ports are active, user-programmable filters are determine which port receives a message, based on the message source or type. Filters apply to the designated Message Select Port (MSP), and are used to select messages for that port. All other messages go to the other (alternate) port. If multiple ports are used, an MSP must be designated; otherwise a configuration error results.

72.5.3.1 Source filter register (NAR_SFR)

The MSP can be configured to accept or exclude messages from up to four sources. The sources which are to be filtered are programmed in the source filter register (NAR_SFR) as shown in the following table. This register should only be set in the BD NAR because partitioning is not supported in the PD NAR.

Register index: 2

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFI	FE0	SF0				0	0	0	FE1	SF1				0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table continues on the next page...



The NAR_SFR is described in the following table. The source trace IDs, that are compared to the source filter fields (SFn), are device specific. For a list of the clients that are connected to the NAR and their corresponding Nexus source trace IDs, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 72-4. NAR_SFR field descriptions

Field	Description	
31 SFI	Source Filter Invert.	
	0	Match if message source equals any of the enabled filters
	1	Match if message source equals none of the enabled filters
30 FE0	Enable Filter 0.	
	0	Filter 0 disabled. SF0 ignored
	1	Filter 0 enabled and contained in SF0
29–26 SF0	Source Filter 0. The message source trace ID is compared with this field.	
25–23	Reserved.	
22 FE1	Enable Filter 1.	
	0	Filter 1 disabled. SF1 ignored
	1	Filter 1 enabled and contained in SF1
21–18 SF1	Source Filter 1. The message source trace ID is compared with this field.	
17–15	Reserved	
14 FE2	Enable Filter 2.	
	0	Filter 2 disabled. SF2 ignored
	1	Filter 2 enabled and contained in SF2
13–10 SF2	Source Filter 2. The message source trace ID is compared with this field.	
9–7	Reserved	
6 FE3	Enable Filter 3.	
	0	Filter 3 disabled. SF3 ignored
	1	Filter 3 enabled and contained in SF3
5–2 SF3	Source Filter 3. The message source trace ID is compared with this field.	
1	Reserved	
0 ACC	OR source and type filters.	
	0	Match if source AND type filters match
	1	Match if source OR type filters match

72.5.3.2 Message type filter register (NAR_TFR)

The MSP can be configured to accept or exclude messages of up to four types, as encoded in the message TCODE field. The TCODEs which are to be filtered are programmed in the message type filter register (NAR_TFR) as shown in the following table. This register should only be set in the BD NAR because partitioning is not supported in the PD NAR.

Register index: 3

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TFI	FE0	TF0						0	FE1	TF1					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	FE2	TF2						0	FE3	TF3					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_TFR is described in the following table.

Table 72-5. NAR_TFR field descriptions

Field	Description
31 TFI	Type Filter Invert. 0 Match if message TCODE equals any of the enabled filters 1 Match if message TCODE equals none of the enabled filters
30 FE0	Enable Filter 0. 0 Filter 0 disabled. TF0 ignored 1 Filter 0 enabled and contained in TF0
29–24 TF0	Type Filter 0. The message TCODE is compared with this field.
23	Reserved
22 FE1	Enable Filter 1. 0 Filter 1 disabled. TF1 ignored 1 Filter 1 enabled and contained in TF1
21–16 TF1	Type Filter 1. The message TCODE is compared with this field.
15	Reserved

Table continues on the next page...

Table 72-5. NAR_TFR field descriptions (continued)

Field	Description	
16 FE2	Enable Filter 2.	
	0	Filter 2 disabled. TF2 ignored
	1	Filter 2 enabled and contained in TF2
13–8 TF2	Type Filter 2. The message TCODE is compared with this field.	
7	Reserved	
6 FE3	Enable Filter 3.	
	0	Filter 3 disabled. TF3 ignored
	1	Filter 3 enabled and contained in TF3
5–0 SF3	Type Filter 3. The message TCODE is compared with this field.	

72.5.4 Trace memory bus configuration registers

The trace memory bus can be used to transfer debug messages out to a dedicated block of external memory space. The trace memory bus configuration registers are used to specify the location and size of the external debug space.

72.5.4.1 Trace memory bus base address registers (NAR_TBALO and NAR_TBAHI)

The trace memory bus base address registers determine where Nexus data sent out through the trace memory bus goes. The NAR sends out blocks of data on the trace memory bus, writing to an address block defined by the specified base address. Data can transmitted either on an as-available, word-wide basis, or in larger blocks. In block transfer mode, the size of the data block is determined by the BTM field of the Control Register; the default is 64 bytes. Even if block transfer mode is not used, the 5, 6, 7, or 8 (depending on the BTS field in NAR_TCR register) LSBs of the base address are not accessible. This forces the debug memory space to be aligned to a block boundary.

When read, the address registers return the current address, allowing the user to monitor the progress of a trace memory transfer. The following tables show the format of the trace memory base address registers (NAR_TBALO and NAR_TBAHI). For a description of the allowed address for these registers, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Register index: 5

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBAHI																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Register index: 4

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBALO																						ADRLO[9]	ADRLO[8]	ADRLO[7]	ADRLO[6]	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_TBALO and NAR_TBAHI registers are described in the following tables.

Table 72-6. NAR_TBAHI field descriptions

Field	Description
31–0 TBAHI	Trace Debug Memory Base Address on AHB-EW port, upper part. For 32-bit address applications this field is all zeros.

Table 72-7. NAR_TBALO field descriptions

Field	Description
31–10 TBALO	Trace Debug Memory Base Address on AHB-EW port, lower part
9 ADRLO[9]	Base Address LSB for BTS = 11 (384-byte block)
8 ADRLO[8]	Base Address LSB for BTS = 10 (256-byte block)
7 ADRLO[7]	Base Address LSB for BTS = 01 (128-byte block)
6 ADRLO[6]	Base Address LSB for BTS = 00 (64-byte block)
5–0	Reserved

72.5.4.2 Trace memory bus control register (NAR_TCR)

The trace memory bus control register is used to control the attributes of the trace memory bus port, such as block transfer size and message priority. Since the trace memory bus transfers are expected to go to a static block of external memory, a maximum transfer size must be specified to prevent the NAR from overwriting memory it is not authorized to access. This is done via the MXFR max transfer size register field. The memory allocation is specified in units of the trace memory bus block transfer size (64, 128, 256, or 384 bytes), so the actual size of the reserved memory block depends on the BTS setting. The following table shows the format of the NAR_TCR.

Register index: 6

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BTM	BTS		BTH		0	0	0	TBW	MXFR						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MXFR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_TCR is described in the following table.

Table 72-8. NAR_TCR field description

Field	Description	
31 BTM	Block Transfer Mode.	
	0	Data for the Trace Memory bus port is transmitted one word at a time, upon first availability
	1	Data to the Trace Memory bus port is bundled into blocks before transmitting
30–29 BTS	Block Transfer Size.	
	00	Block transfer size = 64 bytes
	01	Block transfer size = 128 bytes
	10	Block transfer size = 256 bytes
	11	Block transfer size = 384 bytes
28–27 BTH	Block Transfer Threshold.	
	00	Full threshold. Number of entries in partition must equal block transfer size before block transfer starts
	01	Begin transfer when block is 25% full
	10	Begin transfer when block is 50% full
	11	Begin transfer when block is 75% full
26–24	Reserved. These bits must be written with 0.	

Table continues on the next page...

Table 72-8. NAR_TCR field description (continued)

Field	Description	
23 TBW	Trace Memory Bus Block Wrap.	
	0	NAR stops writing to the Trace Memory bus debug memory space once Max Size has been reached
	1	NAR overwrites Trace Memory bus debug memory from the beginning upon Trace Memory bus block overflow.
22–0 MXFR	Trace Memory Bus Max Transfer Size. Total size of the dedicated debug memory space, in Trace Memory bus data transfer blocks.	

72.5.5 Suppress mode triggers registers (NAR_STCR)

Four suppress mode triggers are provided for precise control of the NAR suppress mode feature described in Suppress mode. Each trigger can be programmed to suppress or wake clients individually or the entire NAR. All receive queues can be suppressed on the occurrence of a selected trigger, or the full NAR can be brought out of suppress mode upon the occurrence of the appropriate trigger. A trigger can also be programmed to toggle one or more clients between the suppressed and active states. Each Suppress Trigger Control Register, shown in the following table, is associated with one of the triggers, and are used to determine the action taken, if any, upon the assertion of that trigger. The suppress triggers are connected to the Sequence Processing Unit (SPU) action unit.

Register index: 7-A

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LST															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_STCR are described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the local suppress trigger, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 72-9. NAR_STCR field descriptions

Field	Description	
31 GST	Global Suppress Trigger.	
	0	The associated trigger cannot cause NAR to enter global suppress mode
	1	The associated trigger is enabled for global suppress mode

Table continues on the next page...

Table 72-9. NAR_STCR field descriptions (continued)

Field	Description	
29–16	Reserved	
15–0	Local Suppress Trigger.	
LST	0000000000000000	The associated trigger is disabled for local suppress mode for all clients
	xxxxxxxxxxxxxxx1	Trigger is enabled for Client 1 local suppress
	xxxxxxxxxxxxxx1x	Trigger is enabled for Client 2 local suppress
	xxxxxxxxxxxxx1xx	Trigger is enabled for Client 3 local suppress
	xxxxxxxxxxxx1xxx	Trigger is enabled for Client 4 local suppress
	xxxxxxxxxxx1xxxx	Trigger is enabled for Client 5 local suppress
	xxxxxxxxxx1xxxxx	Trigger is enabled for Client 6 local suppress
	xxxxxxxxx1xxxxxx	Trigger is enabled for Client 7 local suppress
	xxxxxxxx1xxxxxxx	Trigger is enabled for Client 8 local suppress
	xxxxxxx1xxxxxxxx	Trigger is enabled for Client 9 local suppress
	xxxxxx1xxxxxxxxx	Trigger is enabled for Client 10 local suppress
	xxxxx1xxxxxxxxxx	Trigger is enabled for Client 11 local suppress
	xxxx1xxxxxxxxxxx	Trigger is enabled for Client 12 local suppress
	xxx1xxxxxxxxxxxx	Trigger is enabled for Client 13 local suppress
	xx1xxxxxxxxxxxxx	Trigger is enabled for Client 14 local suppress
	x1xxxxxxxxxxxxxx	Trigger is enabled for Client 15 local suppress
	1xxxxxxxxxxxxxxx	Trigger is enabled for Client 16 local suppress
	All other options	Reserved

72.5.6 Client suppress status register (NAR_CSSR)

Clients can be individually suppressed, leaving the rest of the NAR unaffected. When enabled for local suppress, assertion of one of the suppress triggers causes one or more client receive queues to stop queueing messages. A subsequent pulse of the same trigger causes the client(s) to awaken. Local suppress mode is enabled by setting the appropriate bit in one of the NAR_STCRs. An additional status register, Client Suppress Status (NAR_CSSR), shown in the following table, can then be read to determine which clients are suppressed, and which are active.

Register index: B

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LSS																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_CSSR is described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the local suppress status, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 72-10. NAR_CSSR field descriptions

Field	Description	
31–16	Local Suppress Status.	
LSS	0000000000000000	No client is in local suppress mode
	xxxxxxxxxxxxxxx1	Client 1 is suppressed
	xxxxxxxxxxxxxx1x	Client 2 is suppressed
	xxxxxxxxxxxxx1xx	Client 3 is suppressed
	xxxxxxxxxxxx1xxx	Client 4 is suppressed
	xxxxxxxxxxx1xxxx	Client 5 is suppressed
	xxxxxxxxxx1xxxxx	Client 6 is suppressed
	xxxxxxxxx1xxxxxx	Client 7 is suppressed
	xxxxxxxx1xxxxxxx	Client 8 is suppressed
	xxxxxxx1xxxxxxxx	Client 9 is suppressed
	xxxxxx1xxxxxxxxx	Client 10 is suppressed
	xxxxx1xxxxxxxxxx	Client 11 is suppressed
	xxxx1xxxxxxxxxxx	Client 12 is suppressed
	xxx1xxxxxxxxxxxx	Client 13 is suppressed
	xx1xxxxxxxxxxxxx	Client 14 is suppressed
	x1xxxxxxxxxxxxxx	Client 15 is suppressed
	1xxxxxxxxxxxxxxx	Client 16 is suppressed
	All other options	Reserved
15–0	Reserved	

72.5.7 Receive queue client disable register (NAR_CDR)

Disabling a client reallocates its receive queue to the next active client, allowing that memory space to be put to use. The Client Disable Register, shown in the following table, allows the user to individually disable or enable clients as needed. The status bits also inform the user when the client has new data available.

Register index: C

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAS																RQD															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_CDR is described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the client activity status and receive queue disable, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 72-11. NAR_CDR field descriptions

Field	Description	
31–16	Client Activity Status.	
CAS	xxxxxxxxxxxxxxx1	Client 1 is active with new data
	xxxxxxxxxxxxxx1x	Client 2 is active with new data
	xxxxxxxxxxxxx1xx	Client 3 is active with new data
	xxxxxxxxxxxx1xxx	Client 4 is active with new data
	xxxxxxxxxxx1xxxx	Client 5 is active with new data
	xxxxxxxxxx1xxxxx	Client 6 is active with new data
	xxxxxxxxx1xxxxxx	Client 7 is active with new data
	xxxxxxxx1xxxxxxx	Client 8 is active with new data
	xxxxxxx1xxxxxxxx	Client 9 is active with new data
	xxxxxx1xxxxxxxxx	Client 10 is active with new data
	xxxxx1xxxxxxxxxx	Client 11 is active with new data
	xxxx1xxxxxxxxxxx	Client 12 is active with new data
	xxx1xxxxxxxxxxxx	Client 13 is active with new data
	xx1xxxxxxxxxxxxx	Client 14 is active with new data
	x1xxxxxxxxxxxxxx	Client 15 is active with new data
	1xxxxxxxxxxxxxxx	Client 16 is active with new data
	All other options	Reserved

Table continues on the next page...

Table 72-11. NAR_CDR field descriptions (continued)

Field	Description	
15–0	Receive Queue Disable.	
RQD	xxxxxxxxxxxxxxx1	Disable receive queue for Client 1
	xxxxxxxxxxxxxx1x	Disable receive queue for Client 2
	xxxxxxxxxxxxx1xx	Disable receive queue for Client 3
	xxxxxxxxxxxx1xxx	Disable receive queue for Client 4
	xxxxxxxxxxx1xxxx	Disable receive queue for Client 5
	xxxxxxxxxx1xxxxx	Disable receive queue for Client 6
	xxxxxxxxx1xxxxxx	Disable receive queue for Client 7
	xxxxxxxx1xxxxxxx	Disable receive queue for Client 8
	xxxxxxx1xxxxxxxx	Disable receive queue for Client 9
	xxxxxx1xxxxxxxxx	Disable receive queue for Client 10
	xxxxx1xxxxxxxxxx	Disable receive queue for Client 11
	xxxx1xxxxxxxxxxx	Disable receive queue for Client 12
	xxx1xxxxxxxxxxxx	Disable receive queue for Client 13
	xx1xxxxxxxxxxxxx	Disable receive queue for Client 14
	x1xxxxxxxxxxxxxx	Disable receive queue for Client 15
	1xxxxxxxxxxxxxxx	Disable receive queue for Client 16
	All other options	Reserved

72.5.8 Trace Memory fill level and partition configuration register (NAR_AHFPAR)

This register provides the configuration to set the fill level marker for debug trace memory, connected at the trace memory bus, to generate the partial full event indication and the partition related configuration for main queue (internal central queue). The following table shows the format of the NAR_AHFPAR.

Register index: E

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0			0		

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	AHB01kEN	0	0	0	0	0	0	AHB_01K_Marker			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_AHFPAR is described in the following table.

Table 72-12. NAR_AHFPAR field description

FIELD	Description
31-11	Reserved and must be written with 0s
10 AHB01kEN	Enable 1K Marker Event
9-4	Reserved and must be written with 0s
3–0 AHB_01K_Marker	Value at this field is used to compare a 4-bit slice of internal Trace Memory address bus which works at boundary of 1K. A match with this value generates an event to SPU to indicate the fill status of External Overlay Trace memory. See Overlay and internal trace memory full status generation **Note:** memory ranges from 1K to 15K

72.6 Functional description

High-level descriptions of the operation of the NAR and its main functional blocks are given in the following sections.

72.6.1 Reset/Startup

Assertion of global reset causes the NAR operation to immediately cease. All queues and configuration registers are reset. Upon exiting reset, the Nexus Port Controller (NPC) block inside the NAR is disabled. Holding TMS high for five consecutive rising edges of TCK puts TAP controller into the Test-Logic-Reset state regardless of the current state. This applies reset to only TAP controller and JTAG related logic. Applying soft reset through the NAR_CR resets the entire NAR including the TAP controller and configuration registers.

The typical startup procedure for running a trace through the NPC is as follows:

1. Reset the block
2. Upon exiting reset, the Aurora Phy configuration is determined by sampling the configuration pin values from the Nexus Aurora Phy (NAP). If the NAP is enabled, then the NAR is enabled. The Aurora communication is started only when the NAP is ready to accept data. (If any debug lanes are enabled, the Aurora Link begins training at this time.)

3. Debugger/external user sets the NAR enable (NEN) bit in the NAR_CR and configures the NAR. The configuration includes setting up message selection/ filtering when two output ports are enabled, enabling timestamp and so on.
4. User programs additional device level debug facilities, such as event-processing, performance-monitoring, and so on.
5. User sets up trace filtering.
6. Enable Nexus transmission in clients.

To enable NAR operation, either the Aurora Link must be enabled at reset, or the user must set the NEN bit in the NAR control register.

Note

The HBDP may already be active by the time the NAR comes up. The NAR only tries to initialize the HBDP if it is currently inactive.

The NAR can be enabled to begin immediate operation, or to start in suppress mode, see Suppress mode. Once the NAR and desired ports are enabled, the user should wait until the NAR-ready and HBDP port-ready status bits are set in the NAR status register before beginning Nexus messaging in the clients.

72.6.2 Operation modes

The NAR operating mode is determined by the state of the main control signals: reset and Phy enable. In its enabled state, operation is controlled by its configuration registers. A truth table for all the NAR states is given in the following table.

Table 72-13. NAR operational modes truth table

Reset	NAR enable	NAR comm disable	Global suppress	HBDP enable	ALT partition	MSP partition	Mode
0	x	x	x	x	x	x	RESET
1	0	x	x	0	x	x	Disabled
1	1	1	x	0	x	x	Trace-Disable
1	x	1	x	1	x	x	Trace-Disable
1	0	x	x	1	x	x	Trace-Disable
1	1	0	1	x	x	x	Suppressed
1	1	0	0	x	Trace Memory bus	x	NAR Transmit
1	1	0	0	1	AUR	x	NAR Transmit

Table continues on the next page...

Table 72-13. NAR operational modes truth table (continued)

Reset	NAR enable	NAR comm disable	Global suppress	HBDP enable	ALT partition	MSP partition	Mode
1	1	0	0	1	Trace Memory bus	AUR	NAR Transmit
1	1	0	0	1	AUR	Trace Memory bus	NAR Transmit

There can be overlap between modes as modes are not mutually exclusive.

72.6.2.1 Reset

During reset all internal resources and queues are reset and all output ports are disabled. All inputs are ignored. Upon exiting reset mode, the NAR is in a fully disabled state.

72.6.2.2 Disabled

In the disabled state, all message traffic is ignored, and all output ports are disabled. The NAR is disabled if and only if the NAR enable bit of the NAR control register is not set and the Aurora Link is inactive.

72.6.2.3 Suppressed

The suppressed state differs from the disabled state in that the NAR is on. The NAR pulls out any messages that come up at the clients, keeping the Message Start-End Out (MSEO) bits (so that it can keep track of message boundaries) and discarding the rest.

In the suppressed state, the NAR is waiting for a trigger to begin normal operation. Once that trigger is received, the NAR begins queueing messages that started after the trigger arrived. Incomplete messages that were in transit to the NAR when the trigger arrived are discarded. The NAR can be put into suppressed mode by enabling one of the global suppress states.

72.6.2.4 Trace disable

In trace disable mode, the NAR is powered up but all client-trace is disabled, and no messages are pulled. This mode is also useful for port initialization (to prevent any activity from occurring before the outputs are ready) or for debug/trace-buffer mode, to ensure that, during data retrieval, the NARs queues are stable.

The NAR goes into trace disable (communication-disable) mode when the NAR-enable and NAR communication-disable bits are set or when the HBDP channel is active while the NAR-enable bit is not set.

72.6.2.5 NAR transmit

This is the standard operational mode of the NAR when it is transferring Nexus messages from multiple clients to one or two enabled output port(s). The NAR performs four main operations as described in the following table on the incoming messages before transmitting to one of the output ports.

Table 72-14. NAR transmit operations

Operation	Description
Data reception	NAR accepts data from each client based on the status of the client receive queue status. The maximum data rate is 8 × 32-bit data words (where 8 is the number of Nexus message beats in a full entry and 32 is the number of bits in the Nexus word) per cycle per client. MSEO bits of incoming data are monitored to identify end-of-message boundaries.
Input Arbitration	Active receive queue unloads into main queue(s), 8 × 32 bits per cycle (where 8 is the number of Nexus message beats in a full entry and 32 is the number of bits in the Nexus word). If end-of-message detected, new active receive queue is determined based on client request level or round-robin algorithm.
Output Arbitration	If multiple output ports are enabled, route messages from top of queue to appropriate output port based on message type or source. If split queue is used, outputs can run simultaneously, with each sub-queue communicating directly with a port.
Translation	Data going out the Aurora or trace memory bus ports must be translated into the correct format prior to transmission.

Enabling the NAR and one or more output ports without setting the communication-disable bit puts the NAR into transmit mode.

72.6.3 Data reception and arbitration

The NAR supports 7 clients. Each client has a dedicated input port on the NAR and communicates with its associated input port via a dedicated data bus. A data transmission must consist of an integer number of 32-bit Nexus words (32-bit word is called a beat).

The number of beats varies from client to client. The number of beats (Bn) for each client are described in the Calibration and Debug configuration chapter which describes how the modules are configured.

There are two types of clients supported at NAR client interface. The client types have their own protocols to send Nexus messages. The only common parameter is the Nexus word size (beat). One Nexus word consists of 30 Message Data Out (MDO) bits plus two Message Start/End Out (MSEO) control bits. Internally, the NAR converts all client interfaces to a common interface. To achieve this, the NAR instantiates interface level gaskets that handle the protocol conversion and clock domain transfer.

When new data is available in the core, it is transferred over to the asynchronous gasket in the internal NAR clock domain. The gasket then asserts enough data valids to fill internal receive queue, using a receive queue available signal to track available space in the associated queue, under the assumption that the receive queue available signal is initialized with the size of the queue. If the internal arbiter consumes any of its queued entries, it responds with a data accept signal, which increments the receive queue available signal and allows the gasket to send the next piece of data. Under certain circumstances, like receive queue reallocation, it is possible for the internal arbiter to send out surplus data accept signals such that the receive queue available signal can become larger than the size of the queue. If the receive queues are sized correctly, when NAR is actively servicing a particular client, then asserted data accept signals allow the client to continually stream data until the NAR switches to a new client. The data tracking at the gasket assures that it never sends more data than the receive queue can accept and inactive queues are always full and ready for when they are switched to.

The client gaskets must always be in sync with the internal logic of NAR in terms of the number of available entries in each receive queue. For this reason, the internal NPC and its client gaskets need to be reset by the same signal. Any soft-reset of the NAR (via the SRESET configuration bit) is communicated to all gaskets and used to reset every thing inside it.

Each input port has a dedicated receive queue with a minimum of two entries, where an entry is wide enough to store an entire Bn × 32 data word (where 32 is the number of bits in the Nexus word or the beat). The receive queue can be made larger to allow the NAR to continue transmitting during switches from one client to another in a pipelined environment. Every cycle, if there is space in a particular receive queue, it latches in one full AUX word from its client. Because the client may be running at a much higher frequency than the NAR, several cycles worth of Nexus messages are contained in a single transaction, as shown in the following figure for a 3-beat per word transaction.

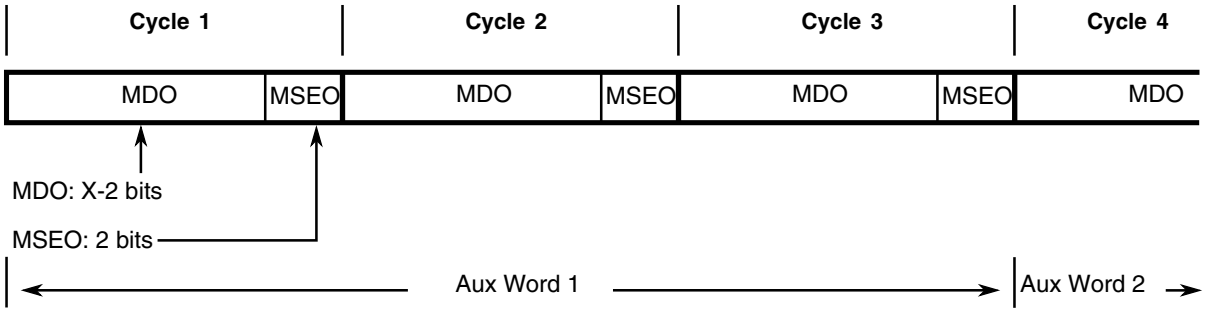


Figure 72-2. AUX bus parsing

Each AUX word consists of Bn cycles of Nexus data, along with an optional request-level sideband signal consisting of a priority field (bit 0) and a weight field (bits 1 and 2). Request level priority indicates to the NAR the clients queue status, allowing it to give preferential treatment to clients that are closer to queue overflow. Clients with a set priority bit are always chosen ahead of clients without the bit. For this reason, priority should be used sparingly, only when a client is in danger of overflowing. Request-level weighting also gives a client preferential treatment, but without entirely shutting out the other clients. Normally, the NAR attempts to switch to a new client every time it sees an end-of-message boundary.

As the AUX word is pulled into the receive queue, the MSEO bits are monitored to determine the current Nexus state of the data flow; since Nexus is a variable-length protocol, messages can end in any cycle. If the start message or end message state is detected on any data beat, the beat is marked as containing an end-of-message boundary.

Any received beat which is in the idle state, indicating null data content, is discarded (see the IEEE-ISTO 5001-2003 standard for details of the Nexus protocol). At the same time, if two outputs are enabled, then the source and TCODE fields are extracted from each new message and compared with the user-programmed filters. Messages that generate a match are tagged with an MSP bit, which allows the NAR to route it to the correct port.

Every cycle, the NAR transfers 8 beats of data from the receive queues into the main output queue. Only one port (the active port) is allowed to empty into the main queue at a time: the active port unloads its receive queue into the main queue until the end of the current Nexus message is detected. At that point, a new active port is selected based on the latest request level received at each port. In the case where multiple ports have the same request level (or when request level is not used), then the active port is rotated amongst those ports. If the end of message occurs in the middle of an AUX word, then all the beats up to the end of message are transferred from the old active port, while the rest of the beats come from the new active port. For example, if the message ended at beat two of a three-beat AUX word, then the first two beats into the main queue would come from the original active port, while the last beat would come from the new active port.

72.6.4 Suppress mode

The suppress mode feature enables the user to quickly turn Nexus messaging through the NAR on or off, based on the four dedicated suppress-mode triggers. A suppress can either be global, meaning messaging is turned off for all clients, or local, which only applies to selected clients. In suppress mode, the NAR pulls messages from the suppressed client(s), but does not queue or deliver them. After extracting the MSEO bits to keep track of message boundaries, the rest of the message is discarded. This allows the NAR to better keep up with fast-running Nexus clients without having to turn off messaging in the client.

Typically, a client would be suppressed during less interesting periods, and would be awakened when some preset condition is detected. This is communicated to the NAR via the one of the suppress triggers, which can be sourced by a client or an external Event Processing block. When an enter suppress trigger is received, the NAR output queue finishes servicing the current active ports until the end of the message. If the end of the message occurs in a middle beat of a Nexus word, then the remainder of the word is padded with null messages. Once the last message has been loaded into the output queue, the receive queues are all emptied. New messages are pulled from clients as they appear, go through their respective receive queues and then are discarded. In this way, the NAR can naturally monitor the MSEO bits and keep track of each client's Nexus state. In suppress mode, the output ports remain active until the output queue is emptied, meaning there is a delay between the assertion of the suppress trigger and the end of messaging.

When the NAR is in suppress mode and receives a wake-up trigger, it continues discarding any partial messages still in the receive queues. Any message that begins after the trigger is queued normally, insuring that only complete messages are sent to the output queue.

Global suppress mode applies to all clients. Any of the suppress triggers can be programmed to cause the NAR to toggle between suppressed and awake by setting the GST bit of the associated Suppress Trigger Control Register (NAR_STCR). Additionally, the NAR starts up in global suppress mode if the PUS bit of the Control Register is set when the NAR is enabled. All clients are then suppressed until a wake-up trigger is received. If the PUS bit is set without enabling any of the triggers for global suppress, a configuration error results and the NAR powers up in active mode.

Clients can go also into local suppress mode without putting the entire NAR to sleep. Each NAR_STCR has a 16-bit LST field. Setting one or more of these bits causes the trigger assigned to that NAR_STCR to toggle the associated clients between suppressed and awake. All other clients are unaffected. Thus, suppress trigger one can be assigned to one set of clients, while suppress trigger two can be assigned to another set of clients, and

different parts of the device can be ignored at different times, all based on different, programmable events. Global suppress always has higher priority than local suppress. That is, if the NAR is in global-suppressed mode, any local suppress wake-up triggers are ignored. Also, if some clients are in local suppress when a global suppress trigger hits, then all go into suppress mode; however, upon the subsequent global wake-up, the clients that were previously in local suppress remain so.

72.6.5 Stall detection

Since the NAR can only switch between clients on an end-of-message boundary, every message that goes into the NAR must complete. If a client happens to die or stall in the middle of transmitting a message, this uncompleted message could cause the NAR to stall, at a time when other clients might still be able to deliver useful information. To prevent this from occurring, the NAR monitors each client for a stall condition, which is defined as occurring when data is not valid in the middle of a Nexus message for a set number of cycles.

When a stall condition is detected, the NAR can do one thing. If the client stalls, then the NAR must complete the current stalled message to continue operating. This scenario is termed a stall error. On a stall error, the NAR injects a dummy message beat into the message stream after the last valid beat of data received from the stalled client. The dummy message is a single beat injected at the output of the receive queue into the main queue. The MSEO bits of the data payload are set to 2b11 regardless of the previous Nexus state. This forces the dummy message into the End Message state. When the NAR injects a dummy message, it also generates an error message to notify the user what has happened.

Injecting a dummy message is an intrusive act, which alters the data stream from the client. If a stalled client is thought likely to recover cleanly, it might be advisable to suppress stall errors to prevent the data stream from being corrupted. This can be done by setting the SED bit of the configuration register, which carries the risk of a truly dead client causing the NAR to hang. Otherwise, the default operation is for the NAR to inject the dummy message and then monitor the client for signs of recovery. If the client does come back online, the NAR attempts to wait until it sees an end-of-message boundary before capturing new data. At best, if the client recovers smoothly, only the interrupted message is lost; if the client does not recover smoothly, it might not be possible for the client and NAR to get back in sync in terms of their Nexus states, which would lead to nothing but garbled data coming out. For this reason, it is also possible to prevent the NAR from trying to recover from a stall error, by setting the SRD bit of the NAR_CR. If the SRD bit is set, a stalled client is not allowed to recover until the SRD bit is unset or the NAR is reset.

72.6.6 Message translation and formatting

The NAR performs the formatting on Nexus messages before sending them to an output port. The exact translation performed depends on the destination port.

72.6.6.1 Trace memory bus format

The trace memory bus works on AHB-EW protocol. Data destined for the trace memory bus port is split back out into 32-bit beats (Nexus word). Depending on the relative sizes of the beat and the width of the trace memory bus port, one or more Nexus words are packed into a trace memory bus data word, with any remaining bits padded to 0.

For example, if the size of the beat = 32 and the width of the trace memory bus = 64, then each trace memory bus data packet contains two beats of Nexus data. Since the data is stored in a regular format (MSEO bits in same place at every location), this makes data interpretation straightforward, regardless of where the trace memory bus port is sending the data. The trace memory bus translation block performs the necessary accumulation and resizing in the likely case where the trace memory bus width does not equal the queue entry size.

72.6.7 Trace memory bus output port usage

The NAR can communicate directly to external memory by sending Nexus data out onto the trace memory bus port, using the AHB-EW (Early Write) protocol. This provides a convenient, secondary area for data in applications or in situations where the HBDP is not available. The trace memory bus access are generated internally by trace memory bus controller.

Because in many systems it is desirable to have the trace memory bus interface run at a divided clock rate relative to the main platform clock, the trace memory bus port operates off of a separate AHB clock. This clock is assumed to be free-running (i.e. unaffected by NAR-enable and the status of the Aurora Link), synchronous to the platform clock used by the rest of the NAR, and either running at the same rate as the platform clock or an integer divide of that rate.

The trace memory bus interface protocol is a slightly modified version of AMBA AHB specification. The write data comes in the address phase instead of data phase and it is termed as Early Write (EW). When NAR uses the trace memory bus port as a destination for Nexus data, it performs a series of writes to a predefined block of memory space.

This memory space is specified through the trace memory bus base address and trace memory bus max transfer configuration registers. The size of the block is set by the max transaction count (MXFR field in NAR_TCR) times the size of a transaction (trace memory bus-width, 64-byte, 128-byte, 256-byte, or 384-byte, depending on block transfer mode and block transfer size). For example, if:

- Base address = 0x0D00_0000 (i.e. TBAHI = 0x0, TBALO = 0x0D00_0000)
- NAR_TCR[BTM] = 1 (Block-transfer mode enabled)
- NAR_TCR[BTS] = 00 (Block-transfer size = 64-byte)
- NAR_TCR[MXFR] = 0x100 (max transaction count)

then the total block size would be 0x100 × 0x40 = 0x4000 bytes. The dedicated debug address range would then be 0x0D00_0000 – 0x0D00_3FFF.

The NAR can either write to the debug space one (trace memory bus-width) word at a time, or for improved performance, it can use a block transfer. If the BTM bit of the trace memory bus Control Register (NAR_TCR) is set, then the NAR uses block transactions to send data through the trace memory bus port, where the size of the block is determined by the setting of the BTS field of the NAR_TCR.

In block transfer mode, the NAR waits until the main queue partition dedicated to the trace memory bus port reaches a specified threshold point, determined by the BTH field of the NAR_TCR, before initiating a block transfer. The default threshold is the size of the block transfer. That is, a block transfer does not start until there is enough data in the associated partition to fill the block. But it might be desirable, for bandwidth reasons or because the associated partition is smaller than the block transfer size, to begin the transfer sooner. Hence, based on BTH, a transfer can begin when the partition is at 25%, 50%, or 75% of the block transfer size. This capability should be used with caution. If the Nexus client does not generate enough new data during the transfer to fill the block, then a block transfer error results.

The NAR generates Nexus trace write transactions for the trace memory bus port using the address from the base address register for the first transaction and then incrementing through the memory space. Once the programmed max number of transactions have been sent, the NAR sets the trace memory bus overflow status bit and, if desired, asserts the NAR event-out. At this point, the NAR can either shut down the trace memory bus port for trace, or, if the TBW bit of the NAR_TCR is set, reset the address to the beginning of the debug space and continue sending trace data. This allows the debugger to view the most recent data, in cases where the buffer overflows.

When TBW bit of NAR_TCR is not set then NAR shuts down the trace memory bus port. To restart the trace memory write again from trace memory base address, the user should write either the NPC trace memory bus base address (TBALO/TBAHI) or the transaction count (MXFR). The user does not have to actually change the value, but can write the same target base address again. The NAR sees any write to one of these registers as a reconfiguration of the trace memory bus port, and resets its internal address pointer to the base address along with clearing the target-space-full flag.

72.6.8 Internal message generation

Generally, the NAR simply conveys Nexus messages from client to port, and does not generate messages of its own. There are, however, certain situations when the NAR needs to communicate directly with the debugger. Internally generated messages are added to the queue the same as external messages. If the queue is partitioned, then internal messages are always channeled to whichever partition is dedicated to the HBDP.

The NAR generates the following message types:

- Device_ID—The first message transmitted when the NAR comes up. It consists of a 6-bit TCODE (TCODE = 1) and the 32-bit device ID. The first beat of the message is always an IDLE (MSEO = 2b11, MDO = all 0s) to allow the debugger Nexus state machine to correctly initialize (by moving to end message or idle state). Device_ID is also generated if the queue is repartitioned (by changing either the ALT or MSP setting in the NAR_CR) while it is not empty. This serves the dual purpose of insuring that the user always gets at least one device_ID message and providing a message boundary in the case where there might have been corruption due to the lost data.
- Watchpoint—The NAR generates its own watchpoint match message under the circumstances shown in Table 72-18.
- NAR_Error (TCODE = 8)—If an error condition is detected inside the NAR, it generates its own error message. An NAR_Error message consists of the 6-bit TCODE, the 6-bit NAR source ID, and a 32-bit data field which corresponds to the NAR status register. An NAR error can be triggered by:
  - HBDP error
  - Illegal configuration
  - Stall Error
  - Trace memory bus read/write access error

This table describes the NAR message formats.

Table 72-15. NAR message formats

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
Device ID	6	6	fixed	TCODE	TCODE value = 1
	32	32	fixed	DID	DID register value
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp
Error Message[1]	6	6	fixed	TCODE	TCODE value = 8
	6	6	fixed	SRC [2, 2]	NAR source ID
	4	4	fixed	ETYPE	See Table 72-16
	12	12	fixed	ECODE	See Table 72-17
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp
Watchpoint Hit Message[3]	6	6	fixed	TCODE	TCODE value = 15
	6	6	fixed	SRC [2]	NAR source ID
	6	6	fixed	WPHIT	See Table 72-18
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp

1. The Error message conforms to the IEEE-ISTO 5001-2012 standard.
2. Tools should treat the NAR SRC field as a 4-bit SRC field and a 2-bit unused field. The extra 2-bit field (the two most significant bits of the 6-bit TCODE) is always 0b00. This operation is not compliant to the IEEE-ISTO 5001 Nexus standard.
3. The Watchpoint Hit message does not conform to the IEEE-ISTO 5001-2012 standard, it conforms to the IEEE-ISTO 5001-2003 standard. The WPHIT should be a variable length packet, but is implemented as a fixed length packet. In most cases, tools will not see a difference, however, a difference will be seen if a timestamp field is appended to the message.

72.6.8.1 Error conditions

NAR errors can occur upon events such as illegal memory accesses, block-transfer overloads, communication errors, or when the NAR is configured incorrectly. If the configuration registers are programmed incorrectly, it is possible to put the NAR into an illegal state. The NARs response varies depending on the error. For a minor error, the NAR reverts to a default mode and continues operation, while major errors result in complete shutdown.

In either case, a Nexus error message is sent to the Aurora out port, if its enabled, otherwise to the trace memory bus port. The Nexus error message contains both ETYPE and ECODE fields:

- ETYPE is used to denote the basic type of error that has occurred
- ECODE carries specific additional information about the error

All NAR error types are defined in the following table.

Table 72-16. NAR error types

ETYPE	Description	Classification	NAR action	Comment
0000 (h'0)	Receive queue overrun	Minor	Dropped message	If a client sends a data word that the NAR cannot accept (due to a full receive queue), the new data word is dropped.
0001 (h'1)	Internal message contention	Minor	Dropped message	Internal errors/watchpoints occurred faster than NAR could generate messages for them, causing lower-priority message(s) to be dropped. The type of message dropped is conveyed in the ECODE field, see Table 72-17.
0011 (h'3)	Trace memory bus port read/ write access error	Minor	Access fails	Unable to send an Aurora-to-trace memory bus request. Typical causes would be writes larger than the data buffer or sending in a new request before the previous one completes.[1]
0101 (h'5)	Invalid NRR	Minor	Access fails	Tried to access an un-implemented NRR.
1001 (h'9)	Config error	Major	See Table 72-17	The specific configuration error is conveyed in bits 0–3 of the ECODE field (Table 72-17), as well as the NCE field of Status Register.
1100 (h'C)	HBDP error	Minor	N/A	A hard error is automatically reset the Aurora Link and causes it to re-initialize. Although the Link is down, the error message is queued and released as soon as the Link comes back up.
1101 (h'D)	Receive Queue Overrun	Minor	Drop message	If a client sends a data word that the NPC cannot accept (due to a full receive queue), the new data word will be dropped2.
1110 (h'E)	Block transfer error	Minor	Dummy fill	If the Trace memory bus port becomes data-starved during a block transfer, it fills the rest of the block with IDLE messages.
1111 (h'F)	Stall error	Minor	—	—

1. In the case where a request was successfully translated to a Trace memory bus transaction which itself returned an error, that is conveyed in the STATUS field of the Nexus Target Response message subsequently returned to the debugger.

Typically, the ECODE field is used to communicate which type of message was dropped in queue overflow or message contention situation, but for the NAR it is also used to denote the type of configuration error that occurred. If a configuration error is detected, the NAR generates a Config Error Nexus message and transmits it, with the actual type of error encoded in the four MSBs of ECODE, as well as the NCE field of the Status Register. The ECODE field encodings are listed in the following table.

Table 72-17. Configuration error codes

ECODE	ETYPE	Description	NAR action	Comment
0000xxxxxxx1	0001	Watchpoint message dropped	N/A	Occurs with internal message overflow error.
0000xxxxxx1x	0001	Data trace message dropped	N/A	Occurs with internal message overflow error; MMA response message was dropped.

Table continues on the next page...

Table 72-17. Configuration error codes (continued)

ECODE	ETYPE	Description	NAR action	Comment
00001xxxxxxx	0001	Response message dropped	N/A	Occurs with internal message overflow error; target response (memory access or NRR) message was dropped.
0000xxx1xxxx	0001	Status message dropped	N/A	Occurs with internal message overflow error.
000100000000	1001	Config error: multiply defined port	Communication disable mode	Both MSP and ALT were assigned to the same port. NPC waits until the configuration is fixed before proceeding.
001000000000	1001	Config error: no ALT defined	Tracebuffer	An MSP was defined without defining the ALT. The NPC reverts to tracebuffer mode.
001100000000	1001	Config error: cannot enable port	Communication disable mode	User has attempted to enable a port that does not exist in this implementation of the NPC.
010000000000	1001	Config error: cannot support message filtering	Message filtering and/or queue partitioning disabled	User has attempted to enable message filtering or queue partitioning with only one output enabled. Filtering/partitioning is turned off.
100100000000	1001	Config error: no wakeup trigger	Power up in active mode	Occurs if none of the defined clients in the system can cause a wakeup (PUS is set in NAR_CR, but no NAR_STCRs have WUT set).

72.6.8.2 Watchpoint messages

The NAR generates its own watchpoint match message under the circumstances shown in the following table. The value of the Watchpoint Hit field of the Watchpoint Message is also shown in this table.

Table 72-18. Watchpoint hit values

WPHIT	Description	Comment
b010000	Entering Suppress Mode	If (local or global) suppress mode is enabled and the appropriate trigger is received, the NAR generates a watchpoint message. The type of event (in this case, a suppress event) is encoded in the WPHIT field of the watchpoint message (see the Nexus standard, sec. 5.3.21).
b100000	Client Sync	If sync messaging is enabled, the NAR generates a sync watchpoint message whenever a client sync event occurs.
b000100	Trace Memory bus Block Overflow (TBO)	If the trace memory bus block is filled, causing it to either disable trace memory bus data trace or wrap back to the beginning of the dedicated debug memory block and if trace memory bus overflow messaging is enabled, then an TBO message is generated.
b001000	Tracebuffer Wrap	If the NAR is being used in tracebuffer mode with the wrap-around feature, then a watchpoint message is generated every time the tracebuffer wraps back to address 0. This would be useful in informing the user that the buffer had wrapped and that older data had been overwritten.

72.6.9 Timestamp function

The NAR and its clients may transmit timestamps to allow precise ordering of events. Synchronous or asynchronous clients use the NAR timer value broadcasted over a bus to ensure that all timestamps are in sync as shown in the following figure. The NAR uses enable signal to centrally control the enabling of time stamping, which informs the clients to start adding the timestamp field to their message traffic.

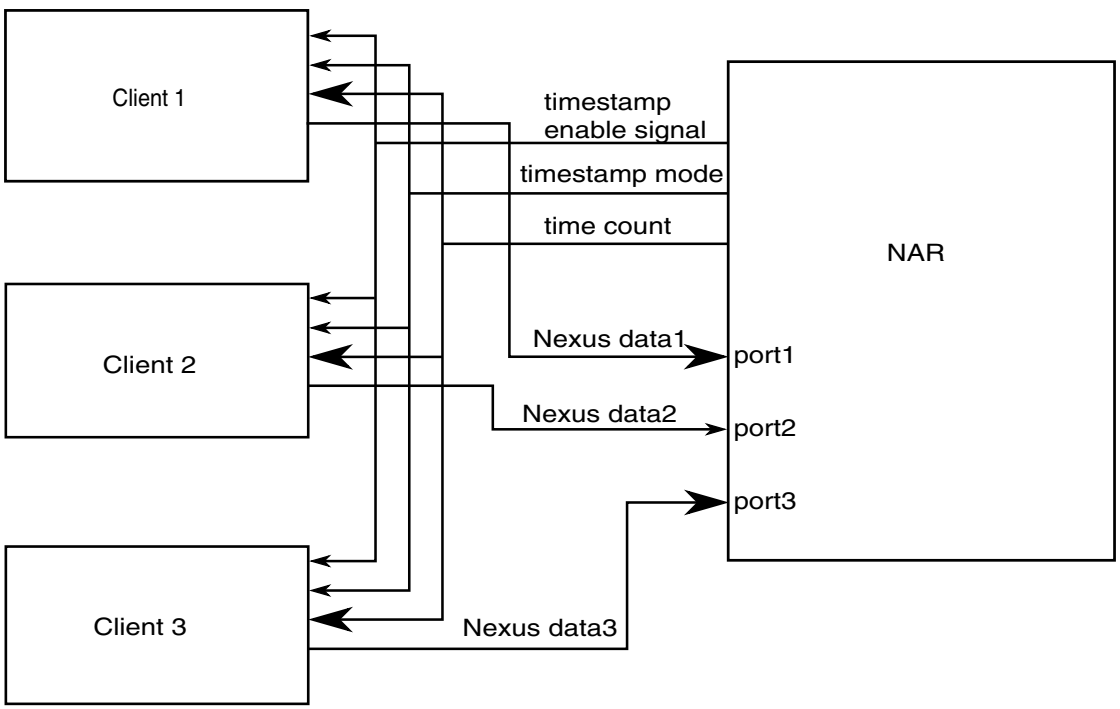


Figure 72-3. Timestamp architecture

Timestamping is enabled by setting the TEN bit of the NAR_CR. This causes the NAR to assert the time stamp enable signal to clients. Typically, the user would also set the TRS bit at this time to reset the NAR Time stamp counter. Internally the NAR has a 30-bit counter external to the NPC block. This 30-bit counter is broadcast to all clients after encoding its value in Gray code. This minimizes the error in synchronization by the clients. The NPC uses an additional internal 24-bit counter to send the timestamp inside its messages. The client converts the gray coded timestamp to binary to append to the Nexus messages.

The timestamp mode, a 2-bit field, is broadcasted to all client which tells the granularity of sending time stamps into message. This is controlled by the TSG bits of the NAR_CR.

Asynchronous clients, those on a separate clock domain from the NAR, synchronize the enable, mode, and timestamp value in their domain. The error introduced by this is assumed to be within tolerable range by application.

A timestamp counter reset can be forced by causing a 0 to 1 transition in the TRS bits of the NAR_CR. The NAR deasserts the enable signal to clients while the reset occurs, and then the counter is restarted (assuming the TEN bit is set). Data throughput can be increased (at the loss of timestamp granularity) by controlling the TSG bits of the NAR_CR. Rather than adding a timestamp to every message, clients can be told to only stamp every 4th, 16th, or 32nd message, easing the bandwidth requirements on the output ports.

The timestamp block maintains the timestamp counter for NAR. It is the central time source for all synchronous/asynchronous clients.

72.6.10 Overlay and internal trace memory full status generation

In certain events, such as main memory (internal trace memory) full or external overlay trace memory full, it is required to control the clients to stop sending data, to avoid any loss of messages from the clients.

The NAR provides three outputs (triggers) to indicate three different situations as described below:

- Internal trace memory (the main queue) is full
- External debug memory on trace memory bus is full
- External debug memory on trace memory bus is full up to a programmable watermark set by the configuration register

The last two triggers are used by SPU to generate stall and suppress triggers. Typically the stall request should be generated whenever space in the external memory connected to trace memory bus has been exhausted. The NAR starts writing from the first location (wrap) as specified in trace memory bus address register in this condition if overwrite trace memory bus debug memory (TBW bit) is set in the NAR_TCR.

The user should configure the partial full watermark register such that it can give around 90% full indication of external debug memory on the trace memory bus. The emulator tool memory full indication is typically generated from the DCI block which actually communicates with the tool.

For example, the following steps can be used to generate event at around 90% fill level for trace memory connected in AHB-EW and to set the watermark in the configuration register.

- From the configuration, the programmer knows the following:

  - AHB start (base) address—Base address = 0x0D00_0000 (i.e.TBAHI = 0x0, TBALO = 0x0D00_0000
  - Block transfer size—NAR_TCR[BTM] = 1 (Block-transfer mode enabled)
  - Max transfer count—NAR_TCR[MXFR] = 0xC0 (maximum transaction count)
- Thus, the total block size is (0xC0 × 0x40) = 0x3000 bytes. The dedicated debug address range is then 0x0D00_0000 – 0x0D00_2FFF.
- To generate an event at nearly 90% fill level of the trace memory at 1 KB boundary, the value to be programmed in the register NAR_AHFPAR[3:0] is calculated as:
  - Number of 1K blocks in 0x3000 bytes = (3000 right shift by 10 bits) = C = 12
  - So the value to be programmed should be 11 as 90% of 12 is almost 11.
- NAR_AHFPAR[3:0] = (NAR_TBALO[13:10] + 0xB) = (0x7+0xB) 0x2 (upper bits truncated due to width) and enable this with NAR_AHFPAR[10] = 1.

This assumes that NAR starts writing from base address and stops at maximum address and no reading is done in overlay memory trace data area during this operation. Whenever the address bus bits [13:10] match with this value, the partial event is generated (toggled).

72.6.11 Configuration/debug interface JTAG

The configuration and status registers of the NAR are accessed through the standard IEEE 1149.1 JTAG interface. The JTAG frequency (TCK) must be 1/2 of the NAR operating clock. The access mechanism through JTAG is described in this section.

The NAR block uses the IEEE 1149.1-2001 TAP for accessing registers. TAP signals include TCK, TDI, TMS, and TDO. The NAR implements a TAP controller state machine that transitions based on the state of the IEEE 1149.1-2001 16-state state machine. It also implements Nexus controller state machine as defined by the IEEE-ISTO 5001-2001 standard.

The instructions implemented by the NAR TAP controller are listed in Table 72-19. Each unimplemented instruction acts like the BYPASS instruction. The size of the NAR instruction register is 4 bits.

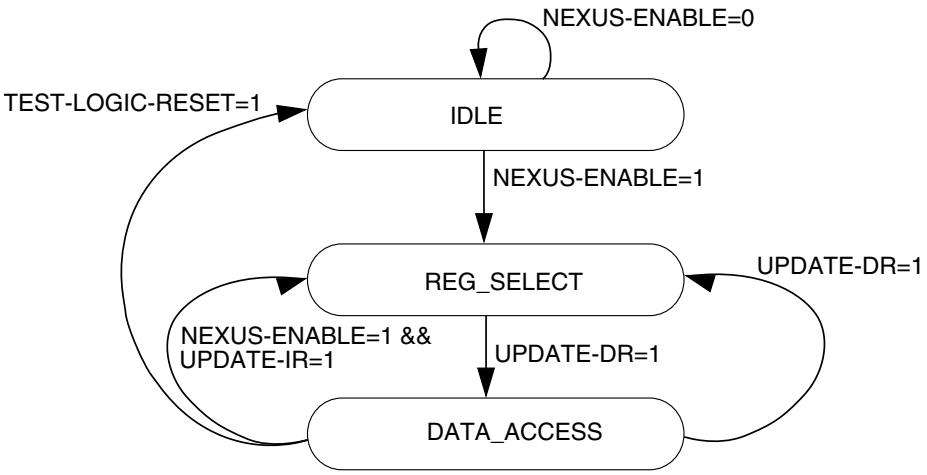


Figure 72-4. Nexus controller state machine

Table 72-19. Implemented instructions

Instruction Name	Private/ Public	Opcode	Description
NEXUS_ENABLE	Public	0x0	Activate Nexus controller state machine to read and write NAR registers.
BYPASS	Private	0x0F	NAR BYPASS instruction. Also the value loaded into the NAR IR upon exit of reset.

Data is shifted between TDI and TDO starting with the least significant bit as illustrated in the following figure. This applies for the instruction register and all Nexus tool-mapped registers.

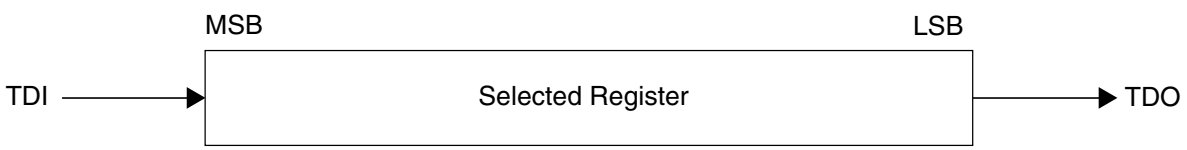


Figure 72-5. Shifting data into register

72.6.11.1 Enabling the NAR TAP controller

Assertion of the power-on reset signal resets the NAR TAP controller. When not in power-on reset the NAR TAP controller is enabled by driving JCOMP with the NAR enable value and exiting the Test-Logic-Reset state. Loading the NEXUS-ENABLE instruction then grants access to Nexus debug.

72.6.11.2 Loading NEXUS-ENABLE instruction

Access to the NAR registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. This instruction is shifted in via the SELECT-IR-SCAN path and loaded in the UPDATE-IR state. At this point, the Nexus controller state machine, shown in Figure 72-4, transitions to the REG_SELECT state. The Nexus controller has three states: idle, register select, and data access. The following table illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

Table 72-20. Loading NEXUS-ENABLE instruction

Clock	TMS	IEEE 1149.1 state	Nexus state	Description
0	0	RUN-TEST/IDLE	IDLE	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	IDLE	Transitional state
2	1	SELECT-IR-SCAN	IDLE	Transitional state
3	0	CAPTURE-IR	IDLE	Internal shifter loaded with current instruction
4	0	SHIFT-IR	IDLE	TDO becomes active, and the IEEE 1149.1-2001 shifter is ready. Shift in all but the last bit of the NEXUS-ENABLE instruction.
5 TCKS				
12	1	EXIT1-IR	IDLE	Last bit of instruction shifted in
13	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
14	0	RUN-TEST/IDLE	REG_SELECT	Ready to be read/write Nexus registers

72.6.11.3 Selecting a JTAG register to create a bus (memory or peripheral) access

When the NEXUS-ENABLE instruction is decoded by the TAP controller, the input port allows the development tool access to all NAR registers. Each register has a 7-bit address index.

All register access is performed via the SELECT-DR-SCAN path. The Nexus controller defaults to the REG_SELECT state when enabled. Accessing a register requires two passes through the SELECT-DR-SCAN path: one pass to select the register and the second pass to read/write the register.

The first pass through the SELECT-DR-SCAN path is used to enter an 8-bit Nexus command consisting of a read/write control bit in the LSB followed by a 7-bit register address index, as illustrated in the following figure. The read/write control bit is set to 1 for writes and 0 for reads. The index for registers are shown in Table 72-1.

Table 72-21. Nexus command format for register selection

MSB	LSB
7-bit register index	R/W

The second pass through the SELECT-DR-SCAN path is used to read or write the register data by shifting in the data (LSB first) during the SHIFT-DR state.When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the CAPTURE-DR state.When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the UPDATE-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

The following table illustrates a sequence which writes a 32-bit value to a register.

Table 72-22. Writing a 32-bit value to a register

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	REG_SELECT	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	REG_SELECT	First pass through SELECT-DR-SCAN path
2	0	CAPTURE-DR	REG_SELECT	Internal shifter loaded with current value of controller command input.
3	0	SHIFT-DR	REG_SELECT	TDO becomes active, and write bit and 4 bits of register index shifted in.
7 TCKs				
11	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI
12	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register
13	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through SELECT-DR-SCAN path
14	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register
15	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				
47	1	EXIT1-DR	DATA_ACCESS	Last bit of current value shifted out TDO. Last bit of new value shifted in TDI.
48	1	UPDATE-DR	DATA_ACCESS	Value written to register
49	0	RUN-TEST/IDLE	REG_SELECT	Controller returned to idle state. It could also return to SELECT-DR-SCAN to write another register.

72.6.12 Trace memory bus controller

The purpose of the trace memory bus is to transfer trace data from main queue to external overlay memory. This port works on a modified AHB protocol that supports early write. It has AHB interface signals as defined in AMBA specification (Rev 2.0). In addition, there are side-band signals for added functionality.

Key features of the module include:

- AHB-EW Interface

- Early Write (for higher BW in special module)
- 32-bit address bus width
- 64-bit data bus width
- Transfer Types
- IDLE, NONSEQ, SEQ
- Single Transfers
- Burst Transfers
- Configurable support for INCR
- Configurable support for INCR4, INCR8, INCR16, WRAP4
- Response Type
- OKAY, ERROR
- Transfer Size
- Byte, Half Word, Word, Double-Word
- Protection Control
- Cacheable/ Non Cacheable Access Control
- User/Privilege Access

- Endianess Support
  - Configurable Little Endian/Big Endian support on AMBA-AHB interface
- Pipeline Support
  - Total of two transactions (One plus one ongoing)

72.6.12.1 Operation

The trace memory bus controller allows all read/write operations to be cut through. The controller has a pipeline depth of two. While the data phase of one transaction is ongoing, it can accept one more read/write request from the internal bus master and initiate the address phase on the AHB interface during the last data cycle of the ongoing transaction.

72.6.12.2 Write operation

When the trace memory bus initiator asserts a request for a write operation, it can find the acknowledge signal parked asserted if the controller is ready to accept the transaction. If the pipeline is full or in case of an ongoing read transaction, the controller inserts wait states by keeping the acknowledge low. The trace memory bus controller initiates the address phase on AHB interface once it accepts the transaction by asserting the acknowledge on the trace memory bus interface. The write data from the trace memory bus initiator is accepted by the gasket by asserting the acknowledge only on completion of address phase at AHB interface, which is signified by receiving the ready signal asserted. Simultaneously, the data is sent to the data bus. Once the requisite number of bytes have been accepted from the trace memory bus initiator, the gasket generates end of data and transfer according to the Trace memory bus protocol.

The AHB protocol supports data transactions of byte, half-word, word and double-word (when both trace memory bus and AHB data bus width is 64 bits). A write transaction from the trace memory bus initiator take place as single/multiple transactions on the AHB side.

72.6.12.3 Read operation

When the trace memory bus initiator asserts a request signal to request for a read operation, it can find the acknowledge signal parked asserted if the gasket is ready to accept the transaction. If the pipeline is full, the controller inserts wait states by keeping the acknowledge low. The trace memory bus controller initiates the address phase on AHB-EW interface once it accepts the transaction by asserting the acknowledge on trace memory bus interface. The controller reads the data from the bus when the ready signal is sensed asserted. The data is transferred on the trace memory bus data bus with the data acknowledge asserted. The gasket asserts end of data and transfer at the end of transaction as per trace memory bus protocol.

The AHB protocol supports data transactions of byte, word, half-word and double-word (when both trace memory bus and AHB data bus width is 64-bits). A read transaction from trace memory bus initiator takes place as single/multiple transaction on the AHB interface before the read data is merged and sent on the trace memory bus read data bus.

In case of variable data bus width on trace memory bus and AHB interface (32-bit data bus on one interface and 64-bit data bus on the other interface), a maximum of 64 bits is read on each beat.

72.6.13 Legacy client interface controller

This block converts the legacy client protocol into internal client protocol that is accepted by the internal NPC block. The Nexus beat size must be same for both interfaces. The Nexus beat is composed of MDO bits along with two MSEO bits. The trace data width from the legacy client is fixed at one beat. The architecture of this block is shown in Figure 72-6.

The legacy client sends each message after arbitrating the bus. It requests the gasket with a 2-bit request signal whenever it has data to send. A 00 means no request, 01 a low-priority request, 10 a medium-priority request, and 11 a high-priority request. The legacy client can send one maximum message sized data on receiving a grant from the block. There is no backpressure mechanism during transfer of a message from the client to this block. The block asserts a grant if it has at least one max message size space in its FIFO. Seeing the grant the client asserts busy signal and holds it until it wants to send data, it may hold it up to max message size data is transferred.

The following architecture describes the legacy client to internal client protocol conversion block. There is a small FIFO of paramaterizeable depth used to store the data from client. The block asserts a grant only when at least one max message sized space is available in FIFO. The write happens in the client's clock domain. The write pointer is passed to other domain which is NPC clock domain.

The logic on other side compares read and write pointer to check the data availability in the block's FIFO and space availability in NPC's receive queue. If space and data both are available it tries to send message to the NPC.

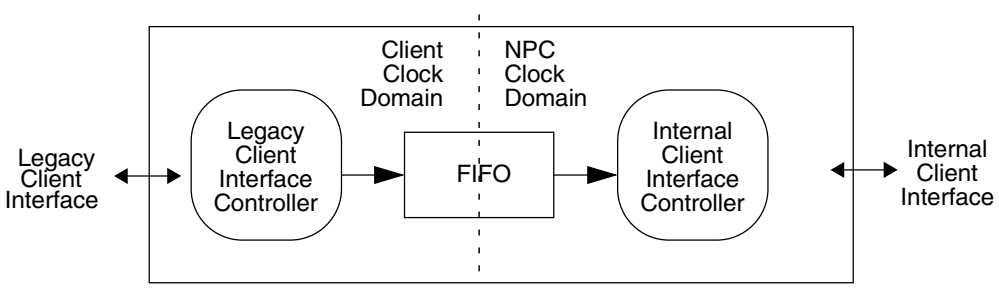


Figure 72-6. Legacy client Interface controller

The size of the receive queue of the corresponding client at NPC must be known to the block. The block drives a not empty signal whenever it has data in FIFO and looks for a ready signal. It can send as many Nexus beats as receive queue can store. It operates on decrements a counter after sending data to the NPC so that it can track how much data it can send. The counter is incremented by the value of acknowledge from NPC which tells how may spaces are freed in receive queue.

72.7 Application information

The base application for the NAR is collecting and collating Nexus trace data.

Example flow of NAR configuration for different cases.

Configuration with Two output port enabled

1. For Two port enabled one port must be Message Select (MSP) and other Alternate (ALT); Trace Memory and Aurora ports with multiple clients

   Selection of MSP port depends on user choice. One option could be based on the requirement of Filtering. If Messages from some known sources with some specific type (if any) is required to be sent to a particular port then select user can select that port as MSP. All other messages will go to the Alternate port. The filter can be used as inverted manner where unmatched messages will go to MSP. (Please see section “Output arbitration” for details)
   - Configure the Source Filter register (NAR_SFR), To enable proper filtering of messages based on Message source going to MSP or ALT port
   - Configure the Type Filter register (NAR_TFR), To enable proper filtering of messages based on Message Type going to MSP or ALT port
   - Configure Trace Memory port attributes : (This step is not required if Trace memory port is not used)
     - Start Address (base address) in NAR_TBALO & NAR_TBAHI register
     - Transfer mode, Block size, Threshold in %, Wrapping enable and Total size of memory
   - Configure Client disable register NAR_CDR (disable clients which are not required)
   - Configure NPC suppress trigger registers NAR_STCR[x] (x=0-3) (suppress the clients for which data is not to be forwarded at output) (Difference between suppressing and disabling is that suppressing can be done dynamically during NAR operation but disabling can’t be done this way which may leads to system hang)
   - Configure the Event generation register (NAR_AHFPAR) for programmable fill level event generation
   - Configure the NAR control register for other controls and Enable NAR
     - Set the MSP as Nexus Out and ALT as Trace memory
     - Set AQP for partitioning the Main Queue for ALT and MSP ports
     - Set time stamp controls
     - Enable NAR to start operation
     - Configuration for Single Output port with single partition in NAR central Message Queue

2. Only Trace Memory port is enabled with multiple client
    - Configure Trace Memory port attributes : (This step is not required if Trace memory port is not used)
        - Start Address (base address) in NAR_TBALO & NAR_TBAHI register
        - Transfer mode, Block size, Threshold in %, Wrapping enable and Total size of memory
    - Configure Client disable register NAR_CDR (disable clients which are not required)
    - Configure the Event generation register (NAR_AHFPAR) for programmable fill level event generation
    - Configure the NAR control register for other controls and Enable NAR
        - Set the Trace Memory port as ALT port (Alternate port is only the main port in single out port application)
        - Set AQP for disabling partitioning
        - Set time stamp controls
        - Set watch point controls
        - Enable NAR to start operation

Chapter 73
Nexus Module

73.1 Introduction

The Nexus 3 module provides real-time development capabilities for corresponding core processors in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

A portion of the pin interface (the JTAG port) is also shared with the OnCE / Nexus 1 unit. The IEEE-ISTO 5001 standard defines an extensible auxiliary port which is used in conjunction with the JTAG port in these processors.

73.1.1 General Description

This chapter defines the auxiliary pin functions, transfer protocols and standard development features of a Class 3 device in compliance with the IEEE-ISTO 5001 standard. The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 3 module also supports two Class 4 features: Watchpoint Triggering, and Processor Overrun Control.

73.1.2 Terms and Definitions

The following table contains a set of terms and definitions associated with the Nexus 3 module.

Table 73-1. Terms and Definitions

Term	Description
IEEE-ISTO 5001	Consortium & standard for real-time embedded system design. World wide Web documentation at http://www.ieee-isto.org/Nexus5001
Auxiliary Port	Refers to Nexus auxiliary port. Used as auxiliary port to the IEEE 1149.1 JTAG interface.
Branch Trace Messaging (BTM)	Visibility of addresses for taken branches and exceptions, and the number of sequential instructions executed between each taken branch.
Data Read Message (DRM)	External visibility of data reads to memory-mapped resources.
Data Write Message (DWM)	External visibility of data writes to memory-mapped resources.
Data Trace Messaging (DTM)	External visibility of how data flows through the embedded system. This may include DRM and/or DWM.
Data Acquisition Messaging (DQM)	Data Acquisition Messaging (DQM) allows code to be instrumented to export customized information to the Nexus Auxiliary Output Port.
JTAG Compliant	Device complying to IEEE 1149.1 JTAG standard
JTAG IR & DR Sequence	JTAG Instruction Register (IR) scan to load an opcode value for selecting a development register. The JTAG IR corresponds to the OnCE command register (OCMD). The selected development register is then accessed via a JTAG Data Register (DR) scan.
Nexus1	The (OnCE) debug module. This module integrated with each processor provides all static (core halted) debug functionality. This module is compliant with Class1 of the IEEE-ISTO 5001 standard.
Ownership Trace Message (OTM)	Visibility of process/function that is currently executing.
Public Messages	Messages on the auxiliary pins for accomplishing common visibility and controllability requirements
SoC	"System-on-a-Chip". SoC signifies all of the modules on a single die. This generally includes one or more processors with associated peripherals, interfaces & memory modules.
Standard	The phrase "according to the standard" is used to indicate according to the IEEE-ISTO 5001 standard.
Transfer Code (TCODE)	Message header that identifies the number and/or size of packets to be transferred, and how to interpret each of the packets.
Watchpoint	A Data or Instruction Breakpoint or other debug event that does not cause the processor to halt. Instead, a pin is used to signal that the condition occurred. A Watchpoint Message may also be generated.

73.1.3 Feature List

The Nexus 3 module is compliant with Class 3 of the IEEE-ISTO 5001 standard, with additional Class 4 features available. The following features are implemented:

- Program Trace via Branch Trace Messaging (BTM). Branch trace messaging displays program flow discontinuities (direct and indirect branches, exceptions, etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced.
- Data Trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to selected internal memory resources.
- Ownership Trace via Ownership Trace Messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated. An Ownership Trace Message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.
- Run-time access to embedded processor memory map via the JTAG port. This allows for enhanced download/upload capabilities.
- Watchpoint Messaging via the auxiliary pins
- Watchpoint Trigger enable of Program and/or Data Trace Messaging
- > External hardware trigger inputs to independently enable/disable Program Trace, Data Trace, Ownership Trace, and Watchpoint Trace Messaging
- Auxiliary interface for higher data
  - Message Data Out pins (**nex_mdo[29:0]**)
  - One (1) or two (2) Message Start/End Out pins (**nex_mseo_b[1:0]**)
  - One (1) Read/Write Ready pin (**nex_rdy_b**) pin
  - One (1) Read/Write Error pin (**nex_err_b**) pin
  - One (1) Watchpoint Event output pin (**nex_evto_b**)
  - Four (4) additional Watchpoint Event output pins (**nex_wevto[3:0]**) for SoC use
  - One (1) Event In pin (**nex_evti_b**)
  - One (1) MCKO (Message Clock Out) pin
- Registers for Program Trace, Data Trace, Ownership Trace and Watchpoint Trigger.
- All features controllable and configurable via the JTAG port
- Conditional software control of the module via SoC signaling input (**nex_sfwcntl_en**)

Note

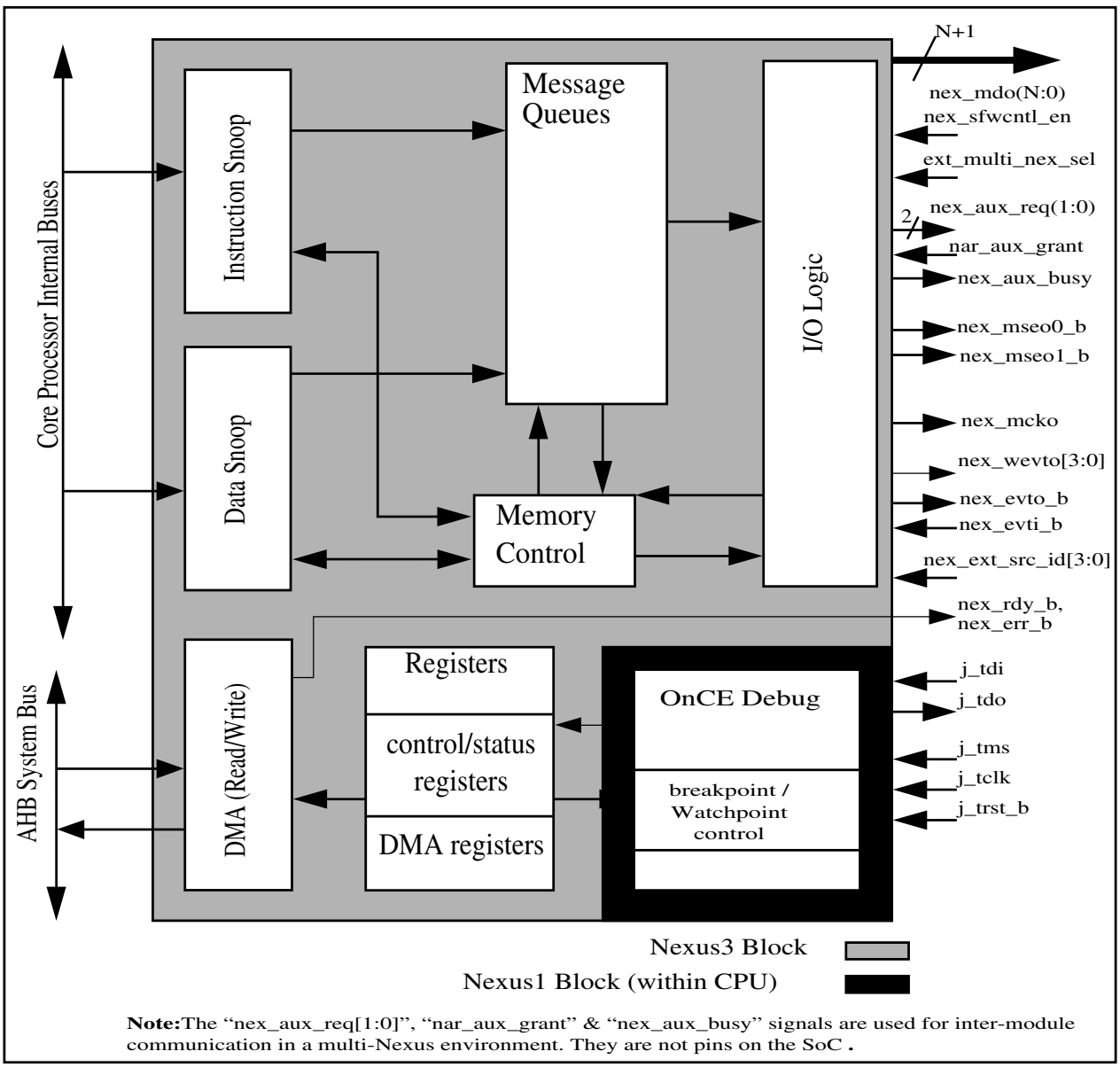
For multi-Nexus implementations, the configuration of the Message Data Out pins is controlled by the Port Control Register (@ the SoC level). For single Nexus implementations (not normally implemented on an SoC), this configuration is controlled by Development Control Register 1 (DC1) within the Nexus 3 module.

In either implementation, Full Port Mode (FPM - maximum number of MDO pins) or Reduced Port Mode (RPM - minimum number of MDO pins) are supported. This setting should not be changed while the system is running.

Note

The configuration of the Message Start/End Out pins (1 or 2) is determined at the SOC integration level. This option will be hard-wired based on SOC bandwidth requirements.

73.1.4 Functional Block Diagram



Note: The "nex_aux_req[1:0]", "nar_aux_grant" & "nex_aux_busy" signals are used for inter-module communication in a multi-Nexus environment. They are not pins on the SoC .

73.2 Enabling Nexus 3 Operation

The Nexus module is enabled by loading a single instruction (*NEXUS3-ACCESS*) into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus3 module, the OCMD value is 0b0001111100. The Nexus 3 module may alternately be enabled if the nex_evti_b input signal is asserted at the time that j_trst_b is initially negated, or is asserted during the Test-Logic-Reset TAP state. Once enabled, the module will be ready to accept control input via the JTAG/OnCE pins.

Enabling the Nexus 3 module automatically enables the generation of Debug Status Messages.

The Nexus 3 module is disabled when the JTAG state machine initially reaches the Test-Logic-Reset state from any other state. This state can be reached by the assertion of the **j_trst_b** pin or by cycling through the state machine using the **j_tms** pin. The Nexus module will also be disabled if a Power-on-Reset (POR) event occurs. If the Nexus 3 module is disabled, no trace output will be provided, and the module will disable (drive inactive) auxiliary port output pins (**nex_mdo[29:0]**, **nex_mseo[1:0]**, **nex_mcko**). Nexus registers will not be available for reads or writes.

In order to support software control of the Nexus 3 module when no external development tool is present, the Nexus 3 module is not forced to be disabled when the JTAG state remains in the Test-Logic-Reset state. Software is allowed to control the Nexus 3 module when the input signal **nex_sfwcntl_en** is asserted by the SoC. This signal is intended to provide a mechanism for allowing software to use the Nexus 3 module to fill on-chip trace buffers or other visibility mechanisms. It is up to the SoC to determine whether a top-level Nexus 3 controller has been enabled by a hardware debugger, or whether appropriate security mechanisms have granted the capability for software to use these resources, and to drive the appropriate value to the **nex_sfwcntl_en** input. Software can enable the module by a write to any of the module's DCRs when **nex_sfwcntl_en** is asserted.

Reset of the Nexus 3 module is accomplished by a transition on **j_trst_b**, on initial entry into the Test_Logic_Reset state from another state, or if a Power-on-Reset (POR) event occurs. The module is not reset by the CPU's **p_reset_b** signal, even when software has control of the module.

73.2.1 Interaction with Low Power Modes

The Nexus 3 module will continue to operate in the Waiting and Halted states, as long as **nex_clk** remains active. In the Stopped state, **nex_clk** is gated off internally to the Nexus 3 logic, therefore watchpoint or hardware triggering recognition, message generation, and Nexus 3 read-write access to memory is suspended. The Nexus 3 logic will wait to enter the Stopped state until the message FIFOs are empty and any in-progress Nexus R/W transfer has completed. If a block transfer has been requested, the remainder of the block transfer is not completed, and the RWCS AC bit is cleared, and the ERR bit is set. Once the Stopped state has been entered, no further messages are queued and no triggering conditions are monitored. Also, Nexus R/W accesses are no longer available. Upon exiting the Stopped state, normal functions will resume assuming **nex_clk** is active.

73.3 TCODEs supported

The Nexus 3 pins allow for flexible transfer operations via Public Messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001 2003 standard defines a set of public messages and allocates additional TCODEs for vendor-specific features outside the scope of the public messages. The Nexus 3 block supports the TCODEs shown in Table 73-2.

Table 73-2. Supported TCODEs

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Debug Status	6	6	TCODE	fixed	TCODE number = 0
	4	4	SRC	fixed	source processor identifier
	8	8	STATUS	fixed	Development Status Register (DS[31:24])
Ownership Trace Message	6	6	TCODE	fixed	TCODE number = 2
	4	4	SRC	fixed	source processor identifier
	1	32	PROCESS	variable	Task/Process ID tag
Program Trace - Direct Branch Message	6	6	TCODE	fixed	TCODE number = 3
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
Program Trace - Indirect Branch Message	6	6	TCODE	fixed	TCODE number = 4
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/ exceptions
Data Trace - Data Write Message	6	6	TCODE	fixed	TCODE number = 5
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 73-7)
	1	32	U-ADDR	variable	unique portion of the data write address
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)

Table continues on the next page...

Table 73-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Data Trace - Data Read Message	6	6	TCODE	fixed	TCODE number = 6
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 73-7)
	1	32	U-ADDR	variable	unique portion of the data read address
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
Data Acquisition Message	6	6	TCODE	fixed	TCODE number = 7
	4	4	SRC	fixed	source processor identifier
	8	8	DQTAG	fixed	identification tag taken from $DEVENT_{DQTAG}$ register field
	1	32	DQDATA	variable	exported data taken from DDAM register
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	source processor identifier
	4	4	ETYPE	fixed	error type
	8	8	ECODE	fixed	error code
Program Trace - Synchronization Message	6	6	TCODE	fixed	TCODE number = 9
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow. Cleared for most sync conditions.
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
Program Trace - Direct Branch Message w/ Sync	6	6	TCODE	fixed	TCODE number = 11
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
Program Trace - Indirect Branch Message w/Sync	6	6	TCODE	fixed	TCODE number = 12
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)

Table continues on the next page...

Table 73-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Data Trace - Data Write Message w/ Sync	6	6	TCODE	fixed	TCODE number = 13
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 73-7)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)
Data Trace - Data Read Message w/ Sync	6	6	TCODE	fixed	TCODE number = 14
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 73-7)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	source processor identifier
	1	32	WPHIT	variable	Field indicating watchpoint source(s) (leading zeros truncated)
Resource Full Message	6	6	TCODE	fixed	TCODE number = 27
	4	4	SRC	fixed	source processor identifier
	4	4	RCODE	fixed	resource code (Refer to Table 73-5) - indicates which resource is the cause of this message
	1	32	RDATA	variable	branch / predicate instruction history (See Section Resource Full Messages.)
Program Trace - Indirect Branch History Message	6	6	TCODE	fixed	TCODE number = 28 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/ exceptions
	1	32	HIST	variable	branch / predicate instruction history (see Branch Trace Messaging types)

Table continues on the next page...

Table 73-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Program Trace - Indirect Branch History Message w/ Sync	6	6	TCODE	fixed	TCODE number = 29 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
	1	32	HIST	variable	branch / predicate instruction history (See Branch Trace Messaging types.)
Program Trace - Program Correlation Message	6	6	TCODE	fixed	TCODE number = 33
	4	4	SRC	fixed	source processor identifier
	4	4	EVCODE	fixed	event correlated w/ program flow (Refer to Table 73-6)
	2	2	CDF	fixed	# fields of information in CDATA. 00 - reserved, 01 - one field (CDATA1) (reserved), 10 - two fields (CDATA1 + CDATA2), 11 - three fields (reserved)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	CDATA1	variable	correlation data field 1 - [branch / predicate instruction history] (See Program Correlation Messages.)
	0	32	CDATA2	variable	correlation data field 2- PID info (See Program Correlation Messages.)

Note

Program Trace can be implemented using either Branch History/Predicate Instruction Messages, or traditional Direct/ Indirect Branch Messages. The user can select between the two types of Program Trace. The advantages for each are discussed in Branch Trace Messaging types. If the Branch History method is selected, the shaded TCODES above will not be messaged out.

Table 73-3 shows the error code encodings used when reporting an error via the Nexus 3 Error Message.

Table 73-3. Error Code (ECODE) Encoding (TCODE = 8)

Error Code	Description
xxxxxxx1	Watchpoint Trace Message(s) Lost
xxxxxx1x	Data Trace Message(s) Lost

Table continues on the next page...

Table 73-3. Error Code (ECODE) Encoding (TCODE = 8) (continued)

Error Code	Description
xxxxx1xx	Program Trace Message(s) Lost
xxxx1xxx	Ownership Trace Message(s) Lost
xxx1xxxx	Status Message(s) Lost (Debug Status messages, etc.)
xx1xxxxx	Data Acquisition Message(s) Lost
x1xxxxxx	Reserved
1xxxxxxx	Reserved

Table 73-4 shows the error type encodings used when reporting an error via the Nexus 3 Error Message.

Table 73-4. Error Type (ETYPE) Encoding (TCODE = 8)

Error Type	Description
0000	Message Queue Overrun caused one or more messages to be lost
0001	Contention with higher priority messages caused one or more messages to be lost
0010	Reserved
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus Register unimplemented)
0110 - 1111	Reserved

Table 73-5 shows the encodings used for resource codes for certain messages.

Table 73-5. Resource Code (RCODE) values (TCODE = 27)

Resource Code	Description
0000	Program Trace Instruction counter reached 255 and was reset.
0001	Program Trace, Branch / Predicate Instruction History full. This type of packet is terminated by a stop bit set to 1 after the last history bit.

Table 73-6 shows the event code encodings used for certain messages.

Table 73-6. Event Code (EVCODE) Encoding (TCODE = 33)

Event Code	Description
0000	Entry into Debug Mode
0001	Entry into Low Power Mode (CPU only)
0010-0011	Reserved for future functionality
0100	Disabling Program Trace
0101	Process ID value is established in PID0/NPIDR via **mtspr PID0/NPIDR**
0110-1000	Reserved for future functionality

Table continues on the next page...

Table 73-6. Event Code (EVCODE) Encoding (TCODE = 33) (continued)

Event Code	Description
1001	Begin masking of program trace messages due to MSR_{PMM}=0 and $DC4_{PTMARK}$=1
1010	Branch and link occurrence (direct branch function call)
1011-1111	Reserved for future functionality

Table 73-7 shows the data trace size encodings used for certain messages.

Table 73-7. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14)

DTM Size Encoding	Transfer Size
0000	0 - no data
0001	Byte
0010	Halfword (2 bytes)
0011	Three bytes
0100	Word (4 bytes)
0101	Five bytes
0110	Six bytes
0111	Seven bytes
1000	Doubleword (8 bytes)
1001-1111	Reserved

73.4 Nexus 3 Programmer's Model

This section describes the Nexus 3 programmers model. Nexus 3 registers are accessed using the JTAG/OnCE port in compliance with IEEE 1149.1, or by software via DCRs. See Nexus 3 Register Access via JTAG/OnCE and Nexus 3 Register Access via Software for details on Nexus 3 register access.

Note

Nexus 3 registers and output signals are numbered using bit 0 as the least significant bit. This bit ordering is consistent with the ordering defined by the IEEE-ISTO 5001 standard.

The following table details the register map for the Nexus 3 module.

Table 73-8. Nexus 3 Register Map

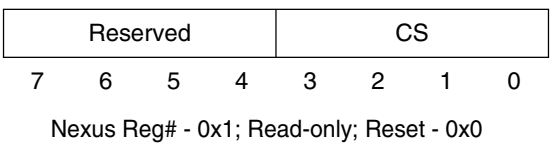
Nexus Register	Nexus Access Opcode	Read/ Write	Read Address	Write Address	DCR #[1]
Client Select Control (CSC)[2, 2]	0x1	R	0x02	-	
Port Configuration Register (PCR)[3]	PCR_INDEX[2]	R/W	-	-	
Development Control 1 (DC1)	0x2	R/W	0x04	0x05	368
Development Control 2 (DC2)	0x3	R/W	0x06	0x07	369
Development Control 3 (DC3)	0x4	R/W	0x08	0x09	370
Development Control 4 (DC4)	0x5	R/W	0x0A	0x0B	371
Reserved	0x6	R/W	0x18	0x19	
Read/Write Access Control/Status (RWCS)	0x7	R/W	0x0E	0x0F	-
Reserved	0x8	R/W	0x18	0x19	
Read/Write Access Address (RWA)	0x9	R/W	0x12	0x13	-
Read/Write Access Data (RWD)	0xA	R/W	0x14	0x15	-
Watchpoint Trigger (WT)	0xB	R/W	0x16	0x17	375
Reserved	0xC	R/W	0x18	0x19	
Data Trace Control (DTC)	0xD	R/W	0x1A	0x1B	376
Data Trace Start Address 1 (DTSA1)	0xE	R/W	0x1C	0x1D	377
Data Trace Start Address 2 (DTSA2)	0xF	R/W	0x1E	0x1F	378
Data Trace Start Address 3 (DTSA3)	0x10	R/W	0x20	0x21	379
Data Trace Start Address 4 (DTSA4)	0x11	R/W	0x22	0x23	380
Data Trace End Address 1 (DTEA1)	0x12	R/W	0x24	0x25	381
Data Trace End Address 2 (DTEA2)	0x13	R/W	0x26	0x27	382
Data Trace End Address 3 (DTEA3)	0x14	R/W	0x28	0x29	383
Data Trace End Address 4 (DTEA4)	0x15	R/W	0x2A	0x2B	408
Reserved	0x16 -> 0x2F	-	0x28->0x5E	0x29->5F	
Development Status (DS)	0x30	R	0x60	-	409
Reserved	0x31	R/W	0x62	0x63	
Overrun Control (OVCR)	0x32	R/W	0x64	0x65	410
Watchpoint Mask (WMSK)	0x33	R/W	0x66	0x67	411
Reserved	0x34	-	0x68	0x69	
Program Trace Start Trigger Control (PTSTC)	0x35	R/W	0x6A	0x6B	412
Program Trace End Trigger Control (PTETC)	0x36	R/W	0x6C	0x6D	413
Data Trace Start Trigger Control (DTSTC)	0x37	R/W	0x6E	0x6F	414
Data Trace End Trigger Control (DTETC)	0x38	R/W	0x70	0x71	415
Reserved	0x39 -> 0x3F	-	0x72->0x7E	0x73->7F	

1. Software access via the **mfdcr** and **mtdcr** instructions use these values for the DCR number. Software writes to these registers via **mtdcr** when **nex_sfwcntl_en** is negated are ignored.
2. The CSC and PCR registers are shown in this table as part of the Nexus programmer's model. They are only present at the top level SoC Nexus controller in a multi-Nexus implementation, not in the Nexus 3 module. The SoC's CSC Register is readable through Nexus, but the PCR is shown for reference only here.

3. The "PCR_INDEX" is a parameter determined by the SoC.

73.4.1 Client Select Control (CSC) register

The CSC Register determines which Nexus client is under development. This register is present at the top-level SOC Nexus 3 controller to select one of multiple on-chip Nexus 3 units.



Nexus Reg# - 0x1; Read-only; Reset - 0x0

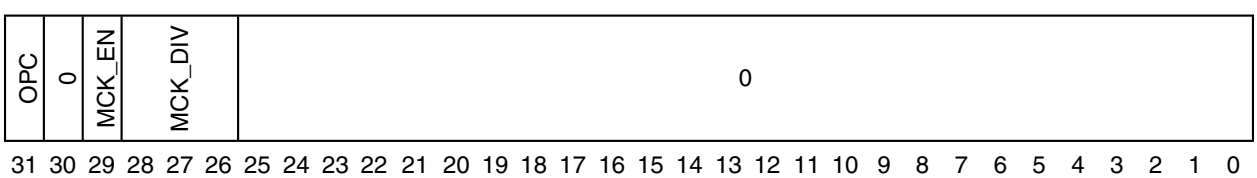
Figure 73-1. Client Select Control (CSC) register

Table 73-9. CSC field descriptions

Bits	Description
CSC[7:4]	Reserved for future Nexus Clients (read as 0)
CSC[3:0]	Client Select Control 0xx - Nexus client (SoC level)

73.4.2 Port Configuration Register (PCR) - reference only

The Port Configuration Register (PCR) controls the basic port functions for all Nexus modules in a multi-Nexus environment. This includes clock control and auxiliary port width. All bits in this register are writable only once after system reset.



Nexus Reg# - PCR_INDEX; Read/Write; Reset - 0x0

Figure 73-2. Port Configuration Register

Table 73-10. PCR field descriptions

Bit	Name	Description
31	OPC	Output Port Mode Control (SoC Level)
		0 Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined by SOC)
		1 Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined by SOC)

Table continues on the next page...

Table 73-10. PCR field descriptions (continued)

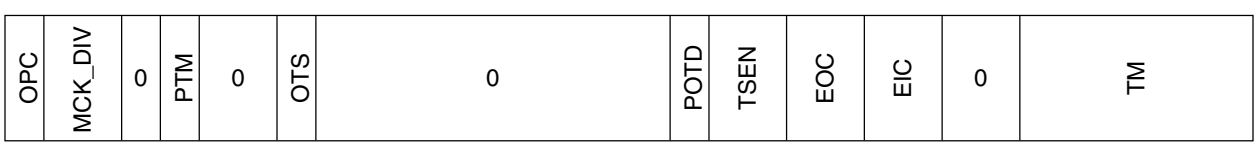
Bit	Name	Description
30	—	Reserved for future functionality
29	MCK_EN	MCKO Clock Enable (SoC Level)
		0 **nex_mcko** is disabled
		1 **nex_mcko** is enabled
28:26	MCK_DIV	MCKO Clock Divide Ratio (see note below) (SoC Level)
		000 **nex_mcko** is 1x processor clock freq.
		001 **nex_mcko** is 1/2x processor clock freq.
		010 Reserved (default to 1/2x processor clock freq.)
		011 **nex_mcko** is 1/4x processor clock freq.
		100 Reserved (default to 1/2x processor clock freq.)
		101 Reserved (default to 1/2x processor clock freq.)
		110 Reserved (default to 1/2x processor clock freq.)
		111 **nex_mcko** is 1/8x processor clock freq.
25:0	—	Reserved for future functionality

Note

The CSC and PCR Registers exist in a separate module at the SoC level in a multi-Nexus environment. If the core Nexus 3 module is the only Nexus module, these registers are not implemented and the Nexus 3 defined Development Control Register 1 (DC1) is used to control the SoC-level Nexus port functionality.

73.4.3 Nexus Development Control Register 1 (DC1)

Nexus Development Control Register 1 is used to control the basic development features of the Nexus 3 module. Development Control Register 1 is shown in Figure 73-3 and its fields are described in Table 73-11.



Nexus Reg# - 0x2; DCR - 368; Read/Write; Reset - 0x0

Figure 73-3. Development Control Register 1

Table 73-11. DC1 field descriptions

Bits	Name	Description
31	OPC	Output Port Mode Control
		0 Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined
		1 Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined
30:29	MCK_DIV	MCKO Clock Divide Ratio (see note below)
		00 **nex_mcko** is 1x processor clock freq.
		01 **nex_mcko** is 1/2x processor clock freq.
		10 **nex_mcko** is 1/4x processor clock freq.
		11 **nex_mcko** is 1/8x processor clock freq.
28	—	Reserved for future functionality
27	PTM	Program Trace Method
		0 Program Trace uses traditional Branch Messages
		1 Program Trace uses Branch History Messages
26:25	—	Reserved for future functionality
24	OTS	Ownership Trace PID Select
		0 PID0 data is transmitted within Ownership Trace Messages
		1 Nexus PID Register (NPIDR) data is transmitted within Ownership Trace Messages
26:15	—	Reserved for future functionality
14	POTD	Periodic Ownership Trace Disable
		0 Periodic Ownership Trace message events are enabled
		1 Periodic Ownership Trace message events are disabled
13:12	TSEN	Timestamp Enable - (not implemented, write to 00)
		00 Timestamp is disabled
11:10	EOC	EVTO Control
		00 **nex_evto_b** upon occurrence of Watchpoints (configured in DC2 and DC3)
		01 **nex_evto_b** upon entry into Debug Mode
		1x Reserved
9:8	EIC	EVTI Control
		00 **nex_evti_b** is used for synchronization (Program Trace Sync msg is generated)
		01 **nex_evti_b** is used for Debug request
		10 **nex_evti_b** is disabled
		1X Reserved
7:6	—	Reserved for future functionality
5:0	TM	Trace Mode[1]
		000000 All Trace Disabled
		XXXXX1 Ownership Trace enabled
		XXXX1X Data Trace enabled
		XXX1XX Program Trace enabled
		XX1XXX Watchpoint Trace enabled
		X1XXXX Reserved
		1XXXXX Data Acquisition Trace enabled

1. This field may be updated by hardware in response to watchpoint triggering or external hardware trigger inputs if not already enable for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers for more information on watchpoint triggering. Refer to External Hardware Trigger Controls for more information on external hardware triggering.

Note

The Output Port Mode Control bit (OPC) and MCKO Clock Divide Ratio bits (MCK_DIV) MUST ONLY be modified during system reset or debug mode to insure correct output port and output clock functionality. It is also recommended that all other bits of the DC1 also only be modified in one of these two modes.

73.4.4 Nexus Development Control Registers 2 & 3 (DC2, DC3)

Nexus Development Control Registers 2 and 3 are used to control output signaling on the Nexus 3 module. A table of watchpoints can be found in the Core Debug Support chapter.

Development Control Register 2 is shown in Figure 73-4 and its fields are described in Table 73-12.

WEVTO[3]C	WEVTO[2]C	WEVTO[1]C	WEVTO[0]C	EWC

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x3; DCR - 369; Read/Write; Reset - 0x0

Figure 73-4. Development Control Register 2

Table 73-12. DC2 field descriptions

Bits	Name	Description
31:28	WEVTO[3]C	Watchpoint Event Out 3 Configuration 0000 No Watchpoints #0-14 trigger **nex_wevto[3]** 0001 Watchpoint #0 triggers **nex_wevto[3]** 0010 Watchpoint #1 triggers **nex_wevto[3]** 0011 Watchpoint #2 triggers **nex_wevto[3]** 0100 Watchpoint #3 triggers **nex_wevto[3]** 0101 Watchpoint #4 triggers **nex_wevto[3]** 0110 Watchpoint #5 triggers **nex_wevto[3]** 0111 Watchpoint #6 triggers **nex_wevto[3]** 1000 Watchpoint #7 triggers **nex_wevto[3]** 1001 Watchpoint #8 triggers **nex_wevto[3]** 1010 Watchpoint #9 triggers **nex_wevto[3]** 1011 Watchpoint #10 triggers **nex_wevto[3]** 1100 Watchpoint #11 triggers **nex_wevto[3]** 1101 Watchpoint #12 triggers **nex_wevto[3]** 1110 Watchpoint #13 triggers **nex_wevto[3]** 1111 Watchpoint #14 triggers **nex_wevto[3]**
27:24	WEVTO[2]C	Watchpoint Event Out 2 Configuration 0000 No Watchpoints #0-14 trigger **nex_wevto[2]** 0001 Watchpoint #0 triggers **nex_wevto[2]** 0010 Watchpoint #1 triggers **nex_wevto[2]** 0011 Watchpoint #2 triggers **nex_wevto[2]** 0100 Watchpoint #3 triggers **nex_wevto[2]** 0101 Watchpoint #4 triggers **nex_wevto[2]** 0110 Watchpoint #5 triggers **nex_wevto[2]** 0111 Watchpoint #6 triggers **nex_wevto[2]** 1000 Watchpoint #7 triggers **nex_wevto[2]** 1001 Watchpoint #8 triggers **nex_wevto[2]** 1010 Watchpoint #9 triggers **nex_wevto[2]** 1011 Watchpoint #10 triggers **nex_wevto[2]** 1100 Watchpoint #11 triggers **nex_wevto[2]** 1101 Watchpoint #12 triggers **nex_wevto[2]** 1110 Watchpoint #13 triggers **nex_wevto[2]** 1111 Watchpoint #14 triggers **nex_wevto[2]**

Table continues on the next page...

Table 73-12. DC2 field descriptions (continued)

Bits	Name	Description
23:20	WEVTO[1]C	Watchpoint Event Out 1 Configuration 0000 No Watchpoints #0-14 trigger **nex_wevto[1]** 0001 Watchpoint #0 triggers **nex_wevto[1]** 0010 Watchpoint #1 triggers **nex_wevto[1]** 0011 Watchpoint #2 triggers **nex_wevto[1]** 0100 Watchpoint #3 triggers **nex_wevto[1]** 0101 Watchpoint #4 triggers **nex_wevto[1]** 0110 Watchpoint #5 triggers **nex_wevto[1]** 0111 Watchpoint #6 triggers **nex_wevto[1]** 1000 Watchpoint #7 triggers **nex_wevto[1]** 1001 Watchpoint #8 triggers **nex_wevto[1]** 1010 Watchpoint #9 triggers **nex_wevto[1]** 1011 Watchpoint #10 triggers **nex_wevto[1]** 1100 Watchpoint #11 triggers **nex_wevto[1]** 1101 Watchpoint #12 triggers **nex_wevto[1]** 1110 Watchpoint #13 triggers **nex_wevto[1]** 1111 Watchpoint #14 triggers **nex_wevto[1]**
19:16	WEVTO[0]C	Watchpoint Event Out 0 Configuration 0000 No Watchpoints #0-14 trigger **nex_wevto[0]** 0001 Watchpoint #0 triggers **nex_wevto[0]** 0010 Watchpoint #1 triggers **nex_wevto[0]** 0011 Watchpoint #2 triggers **nex_wevto[0]** 0100 Watchpoint #3 triggers **nex_wevto[0]** 0101 Watchpoint #4 triggers **nex_wevto[0]** 0110 Watchpoint #5 triggers **nex_wevto[0]** 0111 Watchpoint #6 triggers **nex_wevto[0]** 1000 Watchpoint #7 triggers **nex_wevto[0]** 1001 Watchpoint #8 triggers **nex_wevto[0]** 1010 Watchpoint #9 triggers **nex_wevto[0]** 1011 Watchpoint #10 triggers **nex_wevto[0]** 1100 Watchpoint #11 triggers **nex_wevto[0]** 1101 Watchpoint #12 triggers **nex_wevto[0]** 1110 Watchpoint #13 triggers **nex_wevto[0]** 1111 Watchpoint #14 triggers **nex_wevto[0]**

Table continues on the next page...

Table 73-12. DC2 field descriptions (continued)

Bits	Name	Description
15:0	EWC	EVTO Watchpoint Configuration[1]
		0000000000000000 No Watchpoints #0-15 trigger **nex_evto_b**
		XXXXXXXXXXXXXXX1 Watchpoint #0 triggers **nex_evto_b**
		XXXXXXXXXXXXXX1X Watchpoint #1 triggers **nex_evto_b**
		XXXXXXXXXXXXX1XX Watchpoint #2 triggers **nex_evto_b**
		XXXXXXXXXXXX1XXX Watchpoint #3 triggers **nex_evto_b**
		XXXXXXXXXXX1XXXX Watchpoint #4 triggers **nex_evto_b**
		XXXXXXXXXX1XXXXX Watchpoint #5 triggers **nex_evto_b**
		XXXXXXXXX1XXXXXX Watchpoint #6 triggers **nex_evto_b**
		XXXXXXXX1XXXXXXX Watchpoint #7 triggers **nex_evto_b**
		XXXXXXX1XXXXXXXX Watchpoint #8 triggers **nex_evto_b**
		XXXXXX1XXXXXXXXX Watchpoint #9 triggers **nex_evto_b**
		XXXXX1XXXXXXXXXX Watchpoint #10 triggers **nex_evto_b**
		XXXX1XXXXXXXXXXX Watchpoint #11 triggers **nex_evto_b**
		XXX1XXXXXXXXXXXX Watchpoint #12 triggers **nex_evto_b**
		XX1XXXXXXXXXXXXX Watchpoint #13 triggers **nex_evto_b**
		X1XXXXXXXXXXXXXX Watchpoint #14 triggers **nex_evto_b**
		1XXXXXXXXXXXXXXX Watchpoint #15 triggers **nex_evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

Development Control Register 3 is shown in Figure 73-5 and its fields are described in Table 73-13.

WEVTO[3]C	WEVTO[2]C	WEVTO[1]C	WEVTO[0]C	0	EWC

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x4; DCR - 370; Read/Write; Reset - 0x0

Figure 73-5. Development Control 3 (DCR3) register

Table 73-13. DC3 field descriptions

Bits	Name	Description
31:28	WEVTO[3]C	Watchpoint Event Out 3 Configuration 0000 No Watchpoints #15-#26 trigger **nex_wevto[3]** 0001 Watchpoint #15 triggers **nex_wevto[3]** 0010 Watchpoint #16 triggers **nex_wevto[3]** 0011 Watchpoint #17 triggers **nex_wevto[3]** 0100 Watchpoint #18 triggers **nex_wevto[3]** 0101 Watchpoint #19 triggers **nex_wevto[3]** 0110 Watchpoint #20 triggers **nex_wevto[3]** 0111 Watchpoint #21 triggers **nex_wevto[3]** 1000 Watchpoint #22 triggers **nex_wevto[3]** 1001 Watchpoint #23 triggers **nex_wevto[3]** 1010 Watchpoint #24 triggers **nex_wevto[3]** 1011 Watchpoint #25 triggers **nex_wevto[3]** 1100 Watchpoint #26 triggers **nex_wevto[3]** 1101 - 1111 Reserved
27:24	WEVTO[2]C	Watchpoint Event Out 2 Configuration 0000 No Watchpoints #15-#26 trigger **nex_wevto[2]** 0001 Watchpoint #15 triggers **nex_wevto[2]** 0010 Watchpoint #16 triggers **nex_wevto[2]** 0011 Watchpoint #17 triggers **nex_wevto[2]** 0100 Watchpoint #18 triggers **nex_wevto[2]** 0101 Watchpoint #19 triggers **nex_wevto[2]** 0110 Watchpoint #20 triggers **nex_wevto[2]** 0111 Watchpoint #21 triggers **nex_wevto[2]** 1000 Watchpoint #22 triggers **nex_wevto[2]** 1001 Watchpoint #23 triggers **nex_wevto[2]** 1010 Watchpoint #24 triggers **nex_wevto[2]** 1011 Watchpoint #25 triggers **nex_wevto[2]** 1100 Watchpoint #26 triggers **nex_wevto[2]** 1101 - 1111 Reserved

Table continues on the next page...

Table 73-13. DC3 field descriptions (continued)

Bits	Name	Description
23:20	WEVTO[1]C	Watchpoint Event Out 1 Configuration 0000 No Watchpoints #15-#29 trigger **nex_wevto[1]** 0001 Watchpoint #15 triggers **nex_wevto[1]** 0010 Watchpoint #16 triggers **nex_wevto[1]** 0011 Watchpoint #17 triggers **nex_wevto[1]** 0100 Watchpoint #18 triggers **nex_wevto[1]** 0101 Watchpoint #19 triggers **nex_wevto[1]** 0110 Watchpoint #20 triggers **nex_wevto[1]** 0111 Watchpoint #21 triggers **nex_wevto[1]** 1000 Watchpoint #22 triggers **nex_wevto[1]** 1001 Watchpoint #23 triggers **nex_wevto[1]** 1010 Watchpoint #24 triggers **nex_wevto[1]** 1011 Watchpoint #25 triggers **nex_wevto[1]** 1100 Watchpoint #26 triggers **nex_wevto[1]** 1101 Watchpoint #27 triggers **nex_wevto[1]** 1110 Watchpoint #28 triggers **nex_wevto[1]** 1111 Watchpoint #29 triggers **nex_wevto[1]**
19:16	WEVTO[0]C	Watchpoint Event Out 0 Configuration 0000 No Watchpoints #15-#29 trigger **nex_wevto[0]** 0001 Watchpoint #15 triggers **nex_wevto[0]** 0010 Watchpoint #16 triggers **nex_wevto[0]** 0011 Watchpoint #17 triggers **nex_wevto[0]** 0100 Watchpoint #18 triggers **nex_wevto[0]** 0101 Watchpoint #19 triggers **nex_wevto[0]** 0110 Watchpoint #20 triggers **nex_wevto[0]** 0111 Watchpoint #21 triggers **nex_wevto[0]** 1000 Watchpoint #22 triggers **nex_wevto[0]** 1001 Watchpoint #23 triggers **nex_wevto[0]** 1010 Watchpoint #24 triggers **nex_wevto[0]** 1011 Watchpoint #25 triggers **nex_wevto[0]** 1100 Watchpoint #26 triggers **nex_wevto[0]** 1101 Watchpoint #27 triggers **nex_wevto[0]** 1110 Watchpoint #28 triggers **nex_wevto[0]** 1111 Watchpoint #29 triggers **nex_wevto[0]**
15:14	—	Reserved for watchpoint expansion

Table continues on the next page...

Table 73-13. DC3 field descriptions (continued)

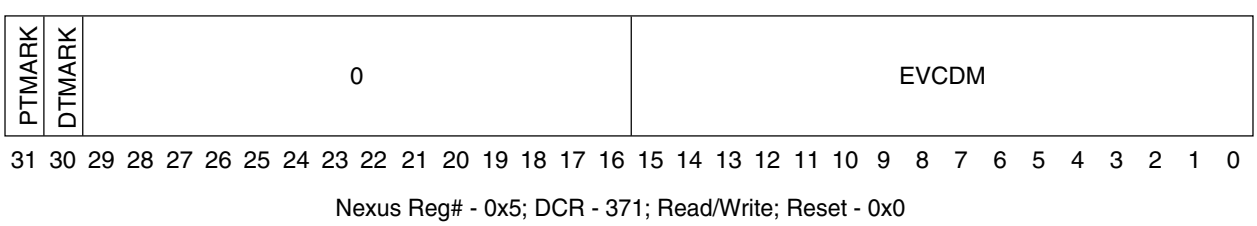
Bits	Name	Description
13:0	EWC	EVTO Watchpoint Configuration[1] 00000000000000 No Watchpoints #16-#29 trigger **nex_evto_b** XXXXXXXXXXXXX1 Watchpoint #16 triggers **nex_evto_b** XXXXXXXXXXXX1X Watchpoint #17 triggers **nex_evto_b** XXXXXXXXXXX1XX Watchpoint #18 triggers **nex_evto_b** XXXXXXXXXX1XXX Watchpoint #19 triggers **nex_evto_b** XXXXXXXXX1XXXX Watchpoint #20 triggers **nex_evto_b** XXXXXXXX1XXXXX Watchpoint #21 triggers **nex_evto_b** XXXXXXX1XXXXXX Watchpoint #22 triggers **nex_evto_b** XXXXXX1XXXXXXX Watchpoint #23 triggers **nex_evto_b** XXXXX1XXXXXXXX Watchpoint #24 triggers **nex_evto_b** XXXX1XXXXXXXXX Watchpoint #25 triggers **nex_evto_b** XXX1XXXXXXXXXX Watchpoint #26 triggers **nex_evto_b** XX1XXXXXXXXXXX Watchpoint #27 triggers **nex_evto_b** X1XXXXXXXXXXXX Watchpoint #28 triggers **nex_evto_b** 1XXXXXXXXXXXXX Watchpoint #29 triggers **nex_evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

73.4.5 Nexus Development Control Register 4 (DC4)

Nexus Development Control Register 4 is used to control mark selection for Program and Data Trace Messaging, as well as masking of events that initiate Program Correlation Messages on the Nexus 3 module.

Development Control Register 4 is shown in Figure 73-6 and its fields are described in Table 73-14.



Nexus Reg# - 0x5; DCR - 371; Read/Write; Reset - 0x0

Figure 73-6. Development Control 4 (DC4) register

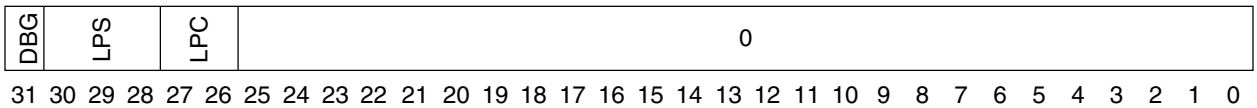
Table 73-14. DC4 field descriptions

Bits	Name	Description
31	PTMARK	Program Trace Mark 0 Ignore MSR_{PMM} for masking program trace messages 1 Mask program trace messages when MSR_{PMM}='0', unmask program trace messages when MSR_{PMM}='1'
30	DTMARK	DTMARK Data Trace Mark 0 Ignore MSR_{PMM} for masking data trace messages 1 Mask data trace messages when MSR_{PMM}='0', unmask data trace messages when MSR_{PMM}='1'
29:16	—	Reserved
15:0	EVCDM	Event Code (EVCODE) Mask[1] 0000000000000000 No EVCODEs masked for Program Correlation Messages XXXXXXXXXXXXXXX1 EVCODE #0 is masked for Program Correlation Messages XXXXXXXXXXXXXX1X EVCODE #1 is masked for Program Correlation Messages XXXXXXXXXXXXX1XX EVCODE #2 is masked for Program Correlation Messages XXXXXXXXXXXX1XXX EVCODE #3 is masked for Program Correlation Messages XXXXXXXXXXX1XXXX EVCODE #4 is masked for Program Correlation Messages XXXXXXXXXX1XXXXX EVCODE #5 is masked for Program Correlation Messages XXXXXXXXX1XXXXXX EVCODE #6 is masked for Program Correlation Messages XXXXXXXX1XXXXXXX EVCODE #7 is masked for Program Correlation Messages XXXXXXX1XXXXXXXX EVCODE #8 is masked for Program Correlation Messages XXXXXX1XXXXXXXXX EVCODE #9 is masked for Program Correlation Messages XXXXX1XXXXXXXXXX EVCODE #10 is masked for Program Correlation Messages XXXX1XXXXXXXXXXX EVCODE #11 is masked for Program Correlation Messages XXX1XXXXXXXXXXXX EVCODE #12 is masked for Program Correlation Messages XX1XXXXXXXXXXXXX EVCODE #13 is masked for Program Correlation Messages X1XXXXXXXXXXXXXX EVCODE #14 is masked for Program Correlation Messages 1XXXXXXXXXXXXXXX EVCODE #15 is masked for Program Correlation Messages

1. Refer to Table 73-6 for implemented EVCODEs.

73.4.6 Development Status Register (DS)

The Development Status Register is used to report system debug status. When Debug Mode is entered or exited, or an SoC- or core-defined Low Power Mode is entered (see note below), a Debug Status Message is transmitted with DS[31:24]. The external tool can read this register at any time.



Nexus Reg# - 0x4; DCR - 409; Read-only; Reset - 0x0

Figure 73-7. Development Status Register

Table 73-15. DS field description

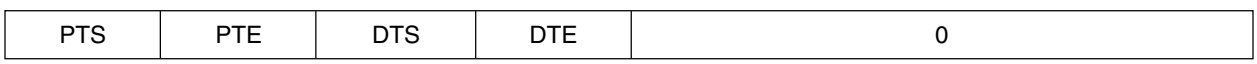
Bits	Name	Description
31	DBG	CPU Debug Mode Status 0 CPU not in Debug mode 1 CPU in Debug mode (**jd_debug_b** signal asserted)
30:28	LPS	System Low Power Mode Status 000 Normal (Run) mode 001 Waiting state (**p_waiting** signal asserted) 010 Halted State (**p_halted** signal asserted) 011 Reserved 1XX Reserved
27:26	LPC	CPU Low Power Mode Status 00 Normal (Run) mode 01 CPU in Halted state (**p_halted** signal asserted) 10 CPU in Stopped state (**p_stopped** signal asserted) 11 CPU in Waiting state (**p_waiting** signal asserted)
25:0	—	Reserved for future functionality (read as 0)

73.4.7 Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers

The Watchpoint Trigger Registers allows the watchpoints defined within the Nexus1 logic to trigger actions. These watchpoints can control Program and/or Data Trace enable and disable. The control bits can be used to produce a related "window" for triggering Trace Messages. Watchpoint trigger register WT is used to control triggering by a single selected watchpoint. The Program Trace Start Trigger Control (PTSTC), Program Trace End Trigger Control (PTETC), Data Trace Start Trigger Control (DTSTC), and Data Trace End Trigger Control (DTETC) are used for extended trigger controls for the respective function. If multiple watchpoints are desired for triggering, or a watchpoint beyond watchpoint #13 is required, then one or more of the extended watchpoint trigger registers may be used. A field encoding of 4'b1111 in one of the WT register fields enables the corresponding extended trigger register. For all other WT field encodings, the corresponding extended trigger register is disabled and the contents are ignored. Note that

direct writes to enable program trace and/or data trace in DC1 will override these controls, and trace will remain enabled until another direct write to DC1 to disable program and/or data trace occurs.

When a start trigger is detected, the designated trace features become enabled, and the corresponding enable bits of the DC1 register are set. Whenever a stop trigger is detected, the designated trace features become disabled, and the corresponding enable bits of the DC1 register are cleared. If the same trigger condition is used for both start and stop triggering, then the designated trace features will toggle between being enabled and disabled at each occurrence of the trigger condition. Similarly, if start and stop triggers for a trace feature occur simultaneously, then the designated trace feature will toggle between enabled and disabled depending on the enable state at the time of the trigger events. For example, if tracing is enabled, and a start and stop trigger occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature enable state that is derived from watchpoint triggering. A table of watchpoints can be found in the Core Debug Support chapter.



Nexus Reg# - 0xB; DCR - 375; Read/Write; Reset - 0x0

Figure 73-8. Watchpoint Trigger (WT) Register

Table 73-16 details the Watchpoint Trigger register fields.

Table 73-16. WT field descriptions

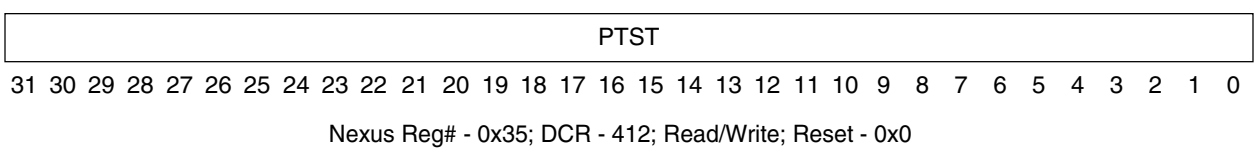
Bits	Name	Description
31:28	PTS	Program Trace Start Control
		0000 Trigger disabled
		0001 Use Watchpoint #0
		0010 Use Watchpoint #1
		.
		.
		1110 Use Watchpoint #13
		1111 Use control settings in the PTSTC register

Table continues on the next page...

Table 73-16. WT field descriptions (continued)

Bits	Name	Description
27:24	PTE	PTE - Program Trace End Control
		0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTETC register
23:20	DTS	Data Trace Start Control
		0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTSTC register
19:16	DTE	Data Trace End Control
		0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTETC register
15:0	—	Reserved for future functionality (read as 0)

For extended Program Trace start trigger control, the PTSTC register is used.



Nexus Reg# - 0x35; DCR - 412; Read/Write; Reset - 0x0

Figure 73-9. Program Trace Start Trigger Control (PTSTC) register

Table 73-17 details the PTSTC register fields.

Table 73-17. Program Trace Start Trigger Control Register Fields

Bits	Name	Description
31:0	PTST	Program Trace Start Trigger Control
		00000000000000000000000000000000 Trigger disabled
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2
		XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3
		XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4
		XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5
		XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6
		XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7
		XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8
		XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9
		XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10
		XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11
		XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12
		XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13
		XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14
		XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15
		XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16
		XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17
		XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18
		XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19
		XXXxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20
		XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #21
		XXXXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23
		XXXXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24
		XXXXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25
		XXXXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26
		XXXX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27
		XXX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #30
		1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #31

For extended Program Trace end trigger control, the PTETC register is used.

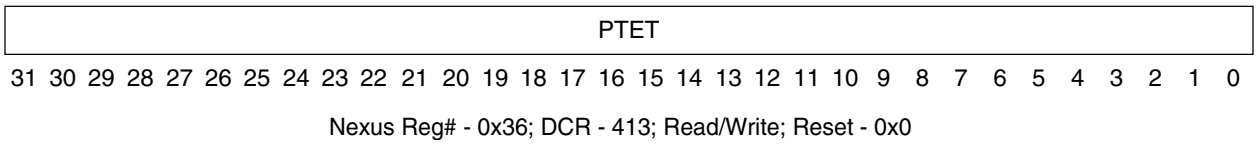


Figure 73-10. Program Trace End Trigger Control (PTETC) register

Table 73-18 details the PTETC register fields.

Table 73-18. PTETC field descriptions

Bits	Name	Description
31:0	PTET	PTET - Program Trace End Trigger Control
		00000000000000000000000000000000 Trigger disabled
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2
		XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3
		XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4
		XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5
		XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6
		XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7
		XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8
		XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9
		XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10
		XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11
		XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12
		XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13
		XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14
		XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15
		XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16
		XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17
		XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18
		XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19
		XXXxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20
		XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #21
		XXXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #22
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #23
		XXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #24
		XXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #25
		XXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #26
		XXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #27
		XXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #28
		XX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #29
		X1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #30
		1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #31

For extended Data Trace start trigger control, the DTSTC register is used.

	DTST

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x37; DCR - 414; Read/Write; Reset - 0x0

Figure 73-11. Data Trace Start Trigger Control (DTSTC) register

The following table details the DTSTC register fields.

Table 73-19. DTSTC field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)

Table continues on the next page...

Table 73-19. DTSTC field descriptions (continued)

Bits	Name	Description
29:0	DTST	Data Trace Start Trigger Control
		00000000000000000000000000000000 Trigger disabled
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2
		XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3
		XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4
		XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5
		XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6
		XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7
		XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8
		XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9
		XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10
		XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11
		XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12
		XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13
		XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14
		XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15
		XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16
		XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17
		XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18
		XXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19
		XxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #21
		XXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22
		XXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23
		XXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24
		XXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25
		XXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28
		1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29

For extended Data Trace end trigger control, the DTETC register is used.

0	DTET

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 038; DCR - 415; Read/Write; Reset - 0x0

Figure 73-12. Data Trace End Trigger Control (DTETC) register

The following table details the DTETC register fields.

Table 73-20. DTET field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)

Table continues on the next page...

Table 73-20. DTET field descriptions (continued)

Bits	Name	Description
29:0	DTET	Data Trace End Trigger Control
		00000000000000000000000000000000 Trigger disabled
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2
		XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3
		XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4
		XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5
		XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6
		XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7
		XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8
		XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9
		XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10
		XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11
		XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12
		XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13
		XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14
		XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15
		XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16
		XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17
		XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18
		XXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19
		XxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #21
		XXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22
		XXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23
		XXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24
		XXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25
		XXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28
		1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29

73.4.8 Nexus Watchpoint Mask (WMSK) register

The Nexus Watchpoint Mask register controls which watchpoint events are enabled to produce Watchpoint Trace Messages ($DC1_{TM}$ must also be programmed to generate Watchpoint Trace Messages).

WEM

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x33; DCR - 411; Read/Write; Reset - 0x0

Figure 73-13. Watchpoint Mask Register

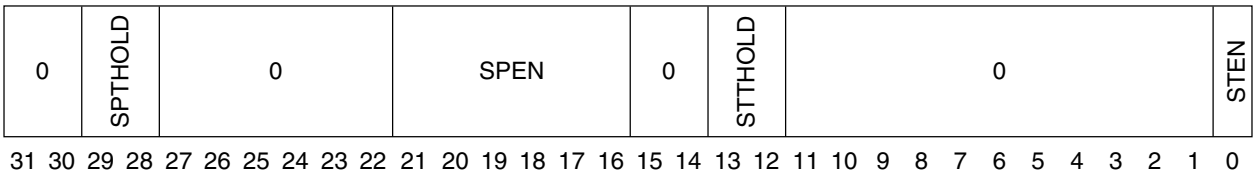
The following table details the Watchpoint Trigger register fields.

Table 73-21. WMSK field descriptions

Bits	Name	Description
31:0	WEM	Watchpoint Enable for Messaging
		00000000000000000000000000000000 No Watchpoints enabled for Watchpoint Trace Messaging
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Watchpoint #0 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Watchpoint #1 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Watchpoint #2 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Watchpoint #3 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Watchpoint #4 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Watchpoint #5 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Watchpoint #6 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Watchpoint #7 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Watchpoint #8 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Watchpoint #9 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Watchpoint #10 enabled for WTM
		XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Watchpoint #11 enabled for WTM
		XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Watchpoint #12 enabled for WTM
		XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Watchpoint #13 enabled for WTM
		XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Watchpoint #14 enabled for WTM
		XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Watchpoint #15 enabled for WTM
		XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Watchpoint #16 enabled for WTM
		XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Watchpoint #17 enabled for WTM
		XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Watchpoint #18 enabled for WTM
		XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Watchpoint #19 enabled for WTM
		XXXXXXXXXXX1XXXXXXXXXXXXXXXXXXXX Watchpoint #20 enabled for WTM
		XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Watchpoint #21 enabled for WTM
		XXXXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Watchpoint #22 enabled for WTM
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Watchpoint #23 enabled for WTM
		XXXXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #24 enabled for WTM
		XXXXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #25 enabled for WTM
		XXXXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #26 enabled for WTM
		XXXX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #27 enabled for WTM
		XXX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #28 enabled for WTM
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #29 enabled for WTM
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #30 enabled for WTM
		1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #31 enabled for WTM

73.4.9 Nexus Overrun Control Register (OVCR)

The Nexus Overrun Control register controls Nexus behavior as the internal message queues fill up. Response options include suppressing selected message types, or stalling processor instruction execution.



Nexus Reg# - 0x32; DCR - 410; Read/Write; Reset - 0x0

Figure 73-14. Nexus Overrun Control Register

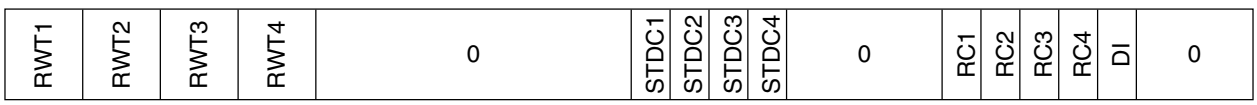
Bits	Name	Description
31:30	—	Reserved, should be cleared
29:28	SPTHOLD	Suppression Threshold 00 - Suppression threshold is when message queues are 1/4 full 01 - Suppression threshold is when message queues are 1/2 full 10 - Suppression threshold is when message queues are 3/4 full 11 - Reserved
27:22	—	Reserved, should be cleared
21:16	SPEN	Suppression Enable 000000 - Suppression is disabled xxxxx1 - Ownership Trace message suppression is enabled xxxx1x - Data Trace message suppression is enabled xxx1xx - Program Trace message suppression is enabled xx1xxx - Watchpoint Trace message suppression is enabled x1xxxx - Reserved 1xxxxx - Data Acquisition message suppression is enabled
15:14	—	Reserved, should be cleared
13:12	STTHOLD	Stall Threshold 00 - Stall threshold is when message queues are 1/4 full 01 - Stall threshold is when message queues are 1/2 full 10 - Stall threshold is when message queues are 3/4 full 11 - Reserved
11:1	—	Reserved, should be cleared
0	STEN	Stall Enable 0 - Stalling is disabled 1 - Stalling is enabled

Figure 73-15. Nexus Overrun Control Register Fields

73.4.10 Data Trace Control Register (DTC) register

The Data Trace Control Register controls whether DTM Messages are restricted to reads, writes, or both for a user programmable address range. In addition, control is provided to restrict data trace message generation to only those load or store accesses that are not stack-related, in order to minimize required data trace message bandwidth.

There are four Data Trace channels controlled by the DTC for the Nexus 3 module. Channels can be programmed to trace data accesses or instruction accesses, but not independently.



Nexus Reg# - 0xD; DCR - 376; Read/Write; Reset - 0x0

Figure 73-16. Data Trace Control Register

The following table details the Data Trace Control register fields.

Table 73-22. DTC field descriptions

Bits	Name	Description
31:30	RWT1	Read/Write Trace 1
		00 No trace enabled
		X1 Enable Data Read Trace
		1X Enable Data Write Trace
29:28	RWT2	Read/Write Trace 2
		00 No trace enabled
		X1 Enable Data Read Trace
		1X Enable Data Write Trace
27:26	RWT3	Read/Write Trace 3
		00 No trace enabled
		X1 Enable Data Read Trace
		1X Enable Data Write Trace
25:24	RWT4	Read/Write Trace 4
		00 No trace enabled
		X1 Enable Data Read Trace
		1X Enable Data Write Trace
23:16	—	Reserved for future functionality (read as 0)

Table continues on the next page...

Table 73-22. DTC field descriptions (continued)

Bits	Name	Description
15	STDC1	Stack Trace Disable Control 1
		0 Tracing of stack accesses are not disabled for range 1
		1 Tracing of stack accesses are disabled for range 1; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
14	STDC2	Stack Trace Disable Control 2
		0 Tracing of stack accesses are not disabled for range 2
		1 Tracing of stack accesses are disabled for range 2; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
13	STDC3	Stack Trace Disable Control 3
		0 Tracing of stack accesses are not disabled for range 3
		1 Tracing of stack accesses are disabled for range 3; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
12	STDC4	Stack Trace Disable Control 4
		0 Tracing of stack accesses are not disabled for range 4
		1 Tracing of stack accesses are disabled for range 4; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
11:8	—	Reserved for future functionality (read as 0)
7	RC1	Range Control 1
		0 Condition trace on address within range
		1 Condition trace on address outside of range
6	RC2	Range Control 2
		0 Condition trace on address within range
		1 Condition trace on address outside of range
5	RC3	Range Control 3
		0 Condition trace on address within range
		1 Condition trace on address outside of range
4	RC4	Range Control 4
		0 Condition trace on address within range
		1 Condition trace on address outside of range
3	DI	Data Access / Instruction Access Trace
		0 Condition trace on data accesses
		1 Condition trace on instruction accesses
		The support for monitoring instruction accesses is not guaranteed to be present in all versions of the Nexus 3 module.
		The setting of the DI bit also affects the information provided on the data trace port outputs (See the "Data Trace Port Signals" section in the Core Debug Support chapter.)
2:0	—	Reserved for future functionality (read as 0)

73.4.11 Data Trace Start Address Registers (DTSA1-4)

The Data Trace Start Address Registers define the start addresses for each trace channel.

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0xE; DCR - 377; Read/Write; Reset - 0x0

Figure 73-17. Data Trace Start Address 1 (DTSA1) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0xF; DCR - 378; Read/Write; Reset - 0x0

Figure 73-18. Data Trace Start Address 2 (DTSA2) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x10; DCR - 379; Read/Write; Reset - 0x0

Figure 73-19. Data Trace Start Address 3 (DTSA3) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x11; DCR - 380; Read/Write; Reset - 0x0

Figure 73-20. Data Trace Start Address 4 (DTSA4) register

73.4.12 Data Trace End Address (DTEA1-4) registers

The Data Trace End Address Registers define the end addresses for each trace channel.

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x12; DCR - 381; Read/Write; Reset - 0x0

Figure 73-21. Data Trace End Address 1 (DTEA1) register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x13; DCR - 382; Read/Write; Reset - 0x0

Figure 73-22. Data Trace End Address 2 (DTEA2) register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x14; DCR - 383; Read/Write; Reset - 0x0

Figure 73-23. Data Trace End Address 3 (DTEA3) register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x15; DCR - 408; Read/Write; Reset - 0x0

Figure 73-24. Data Trace End Address 4 (DTEA4) register

Table 73-23 illustrates the range that will be selected for Data Trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 73-23. Data trace - address range options

Programmed values	Range Control Bit Value	Range Selected	
DTSA < DTEA	0	DTSA ->	<- DTEA
DTSA < DTEA	1	<- DTSA	DTEA ->
DTSA > DTEA	N/A	Invalid Range - no trace	
DTSA = DTEA	N/A	Invalid Range - no trace	

Note

DTSA must be less than DTEA in order to guarantee correct Data Write/Read Traces. Data Trace ranges are *inclusive* of the DTSA and DTEA addresses for Range Control settings indicating "within range," and are *exclusive* of the DTSA and DTEA addresses for Range Control settings indicating "outside of range."

Accesses that meet the range and access type qualifiers will cause assertion of a watchpoint output for Ranges 1, 2, and 3. There are three dedicated watchpoint outputs, one for each DTSA/DTEA sets 1, 2, and 3. Range 4 does not provide a watchpoint output. Note that when DTC_{DI}=1, all instruction fetches and prefetches (including discarded prefetches) are monitored, and thus these range watchpoints differ from the IACx watchpoint outputs, which are not asserted for instructions that are not executed (i.e. when the instruction prefetch is discarded).

73.4.13 Read/Write Access Control/Status (RWCS) register

The Read Write Access Control/Status Register provides control for Read/Write Access. Read/Write access provides DMA-like access to memory-mapped resources on the AHB System bus either while the processor is halted, or during runtime. Control is provided over access type, size, count, and certain bus attributes.

Note that the internal CPU interfaces to the caches and local memory are not accessible or utilized by this mechanism. Since the external interface is used, base address port settings are used to access local memory, and not the settings of local memory control register base address values.

0The RWCS Register also provides Read/Write Access Status information per Table 73-25.

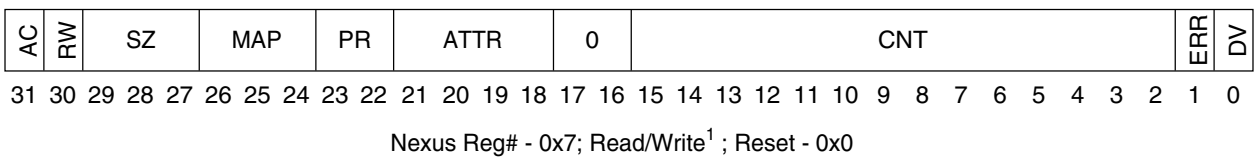


Figure 73-25. Read/Write Access Control/Status (RWCS) register

NOTES:

[1]ERR and DV are read-only

Table 73-24. RWCS field description

Bits	Name	Description
RWCS[31]	AC	Access Control
		0 End access/ Access has completed
		1 Start access
RWCS[30]	RW	Read/Write Select
		0 Read access
		1 Write access
RWCS[29:27]	SZ	Word Size
		000 8-bit (byte)
		001 16-bit (half-word)
		010 32-bit (word)
		011 64-bit (doubleword, requires two passes through RWD)
		100 - 111 Reserved (default to word)
RWCS[26:24]	MAP	MAP Select
		000 Primary memory map
		001-111 Reserved

Table continues on the next page...

Table 73-24. RWCS field description (continued)

Bits	Name	Description
RWCS[23:22]	PR[1]	Read/Write Access Priority
		00 Reserved (default to highest priority)
		01 Reserved (default to highest priority)
		10 Reserved (default to highest priority)
		11 Highest access priority
RWCS[21:18]	ATTR	Access Attributes
		0xxx Reserved
		1xxx Reserved
		x0xx **p_d_hprot[4]** driven to 0 for accesses
		x1xx **p_d_hprot[4]** driven to 1 for accesses
		xx0x **p_d_hprot[3]** driven to 0 for accesses
		xx1x **p_d_hprot[3]** driven to 1 for accesses
		xxx0 **p_d_hprot[2]** driven to 0 for accesses
		xxx1 **p_d_hprot[2]** driven to 1 for accesses
RWCS[17:16]	—	RES - Reserved for future functionality
RWCS[15:2]	CNT	Access Control Count
		hhhh Number of accesses of word size SZ
RWCS[1]	ERR[2, 2]	Read/Write Access Error (see Table 73-25)
RWCS[0]	DV[2]	Read/Write Access Data Valid (see Table 73-25)

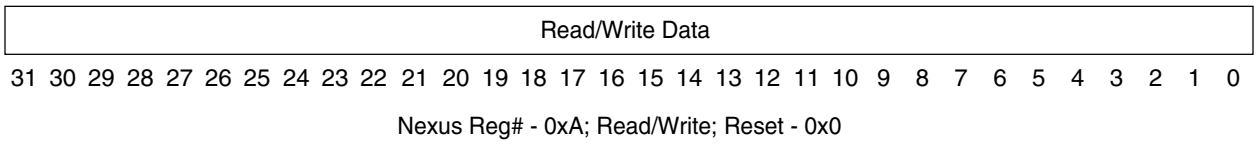
1. The priority functionality is not currently implemented.
2. ERR and DV are read-only.

Table 73-25. Read/Write Access Status Bit Encoding

Read Action	Write Action	ERR	DV
Read Access has not completed	Write Access completed without error	0	0
Read Access error has occurred	Write Access error has occurred	1	0
Read Access completed without error	Write Access has not completed	0	1
Not Allowed	Not allowed	1	1

73.4.14 Read/Write Access Data (RWD) register

The Read/Write Access Data (RWD) register provides the data to/from system bus memory-mapped locations when initiating a read or a write access.



Nexus Reg# - 0xA; Read/Write; Reset - 0x0

Figure 73-26. Read/Write Access Data (RWD) register

Read/Write accesses to the AHB require that the debug firmware properly retrieve/place the data in the RWD. Table 73-26 shows the proper placement of data into the RWD. Note that doubleword transfers require two passes through RWD.

Table 73-26. RWD data placement for ransfers

Transfer Size and byte offset	RWA(2:0)	RWCS[SZ]	RWD			
			31:24	23:16	15:8	7:0
Byte	x x x	0 0 0	-	-	-	X
Half	x x 0	0 0 1	-	-	X	X
Word	x 0 0	0 1 0	X	X	X	X
Doubleword	0 0 0	0 1 1				
first RWD pass (low order data)			X	X	X	X
second RWD pass (high order data)			X	X	X	X

Table Notes:

"X" indicates byte lanes with valid data

"-" indicates byte lanes that will contain unused data.

Table 73-27 shows the mapping of RWD bytes to byte lanes of the AHB read and write data buses.

Table 73-27. RWD byte lane mapping

Transfer Size and byte offset	RWA(2:0)	RWD			
		31:24	23:16	15:8	7:0
Byte @000	0 0 0	-	-	-	AHB[7:0]
Byte @001	0 0 1	-	-	-	AHB[15:8]
Byte @010	0 1 0	-	-	-	AHB[23:16]
Byte @011	0 1 1	-	-	-	AHB[31:24]
Byte @100	1 0 0	-	-	-	AHB[39:32]
Byte @101	1 0 1	-	-	-	AHB[47:40]
Byte @110	1 1 0	-	-	-	AHB[55:48]
Byte @111	1 1 1	-	-	-	AHB[63:56]
Half @000	0 0 0	-	-	AHB[15:8]	AHB[7:0]
Half @010	0 1 0	-	-	AHB[31:24]	AHB[23:16]
Half @100	1 0 0	-	-	AHB[47:40]	AHB[39:32]

Table continues on the next page...

Table 73-27. RWD byte lane mapping (continued)

Transfer Size and byte offset	RWA(2:0)	RWD			
		31:24	23:16	15:8	7:0
Half @110	1 1 0	-	-	AHB[63:56]	AHB[55:48]
Word @000	0 0 0	AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
Word @100	1 0 0	AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]
Doubleword @000	0 0 0				
first RWD pass		AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
second RWD pass		AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]

Table Notes:

"-" indicates byte lanes that will contain unused data.

73.4.15 Read/Write Access Address (RWA)

The Read/Write Access Address Register provides the system bus address to be accessed when initiating a read or a write access.

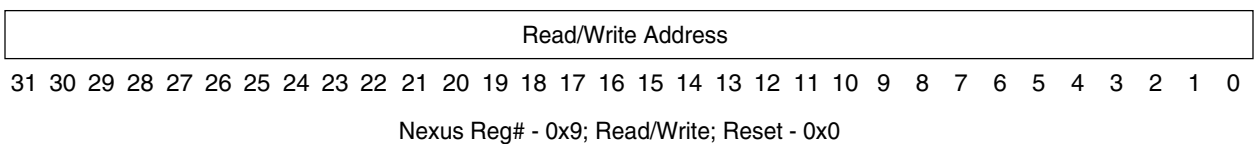


Figure 73-27. Read/Write Access Address (RWA) register

73.5 Nexus 3 Register Access via JTAG/OnCE

Access to Nexus 3 register resources is enabled by loading a single instruction ("*NEXUS3-ACCESS*") into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus 3 block, the OCMD value is 0b0001111100.

Once the NEXUS3-ACCESS instruction has been loaded, the JTAG/OnCE port allows tool/target communications with all Nexus 3 registers according to the register map in Table 73-8.

Reading/writing of a Nexus 3 register then requires two (2) passes through the Data-Scan (DR) path of the JTAG state machine. (See IEEE 1149.1 (JTAG) RD/WR sequences.)

1. The first pass through the DR selects the Nexus 3 register to be accessed by providing an index (see Table 73-8), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the following format:

(7bits)	(1 bit)
Nexus Register Index	R/W

RESET Value: 0x00

Nexus Register Index	Selected from values in Table 73-8
Read/Write (R/W):	0 - Read 1 - Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
   a. During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the "Capture-DR" state.
   b. During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the "Update-DR" state.

73.6 Nexus 3 Register Access via Software

Access to most Nexus 3 register resources by software is supported by mapping the Nexus 3 registers to privileged DCRs (device control registers) that are accessible via the **mfdcr** and **mtdcr** instructions. It is intended that these access only occur when no possible conflicts with hardware-based accesses are possible. A conflict between hardware and software accesses will produce undefined results. Table 73-8 shows the mappings of Nexus 3 registers to DCR numbers. Software writes to Nexus 3 register resources are blocked when the **nex_sfwcntl_en** input signal is negated, without signaling an exception. Note that the Nexus 3 Read/Write Access functionality is not available to software, since software can use load and store instructions to access memory.

73.7 Nexus Message Fields

Nexus messages are comprised of fields. Each field contains a distinct piece of information within a message, and each message contains multiple fields. Messages are transferred in packets over the Auxiliary Output protocol. A packet is a collection of fields. A packet may contain any number of fixed length fields, but may contain at most one variable length field. The variable length field must be the last field in a packet. The following sub-sections describe a subset of the message field types.

73.7.1 TCODE Field

The TCODE field is a 6-bit fixed length field that identifies the type of message and its format. The field encodings are assigned by IEEE-ISTO 5001.

73.7.2 Source ID Field (SRC)

Each Nexus module in a device is identified by a unique Client Source Identification Number. The number assigned to each Nexus module is determined by the SoC integrator, and is provided on the **nex3_ext_src_id[0:3]** input signals. Multi-threaded processors may assign additional source ID information to indicate which thread a message is associated with. The Nexus 3 module implements a 4-bit fixed length Source ID field consisting of a Client Source ID.

73.7.3 Relative Address Field (U-ADDR)

The non-sync forms of the Program and Data Trace messages include addresses that are relative to the address that was transmitted in the previous Program or Data Trace message, respectively. The relative address format is compliant with IEEE-ISTO 5001 and is designed to reduce the number of bits transmitted for address fields.

The relative address is generated by XORing the new address with the previous, and then using only the results up to the most significant '1'. To recreate the original address, the relative address is XORed with the previously decoded address.

The relative address of a Program Trace message is calculated with respect to the previous Program Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Program Trace messages.

The relative address of a Data Trace message is calculated with respect to the previous Data Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Data Trace messages.

Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

Previous Address (A1) =0x0003FC01, New Address (A2) = 0x0003F365

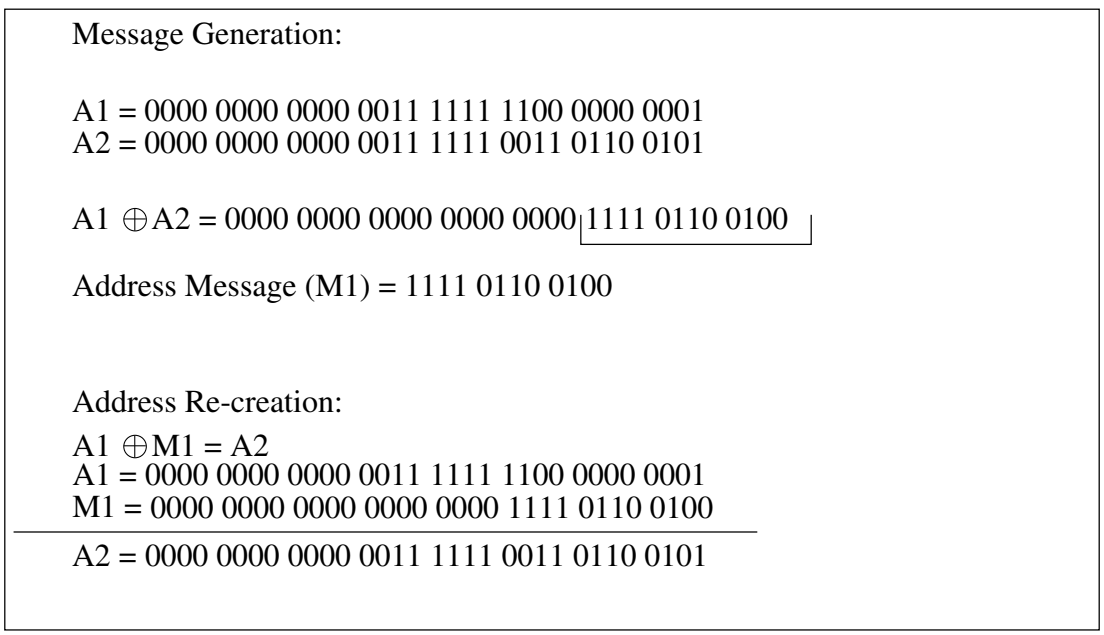


Figure 73-28. Relative address generation and recreation

73.7.4 Full Address Field (F-ADDR)

Program Trace synchronization messages provide the full address associated with the trace event (leading zeroes may be truncated) with the intent of providing a reference point for development tools to operate from when reconstructing relative addresses. Synchronization messages are generated at significant mode switches and are also generated periodically to ensure that development tools are guaranteed to have a reference address given a sufficiently large sample of trace messages. Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and

history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

73.8 Nexus Message Queues

The Nexus 3 module implements internal message queues capable of storing up to three messages per cycle into a small initial queue, which then fills a larger queue at up to two messages per cycle. Messages that enter the queues are transmitted in the order in which they are received.

If more than three messages attempt to enter the queue in the same cycle, the highest priority messages are stored and the remaining message(s) will be dropped due to a collision. Collision events are expected to be rare.

The Overrun Control register (OVCR) controls the Nexus behavior as the message queue fills. The Nexus block may be programmed to:

- Allow the queue to overflow, drain the contents, queue an overrun error message and resume tracing.
- Stall the processor when the queue utilization reaches the selected threshold.
- Suppress selected message types when the queue utilization reaches the selected threshold.

73.8.1 Message Queue Overrun

In this mode, the message queue will stop accepting messages when an overrun condition is detected. The contents of the queues will be allowed to drain until empty. Incoming messages are discarded until the queue is emptied. Once empty, an overrun error message is enqueued that contains information about the types of messages that were discarded due to the overrun condition.

73.8.2 CPU Stall

In this mode, processor instruction issue is stalled when the queue utilization reaches the selected threshold. The processor is stalled long enough to drop one threshold level below the level that triggered the stall. For example, if stalling the processor is triggered at 1/4 full, the stall will stay in effect until the queue utilization drops to empty. There may be

significant skid from the time that the stall request is made until the processor is able to stop completing instructions. This skid should be taken into consideration when programming the threshold. Refer to Nexus Overrun Control Register (OVCR) for complete programming options.

73.8.3 Message Suppression

In this mode, the message queue will disable selected message types when the queue utilization reaches the selected threshold. This allows lower bandwidth tracing to continue and possibly avoid an overrun condition. If an overrun condition occurs despite this message suppression, the queue will respond according to the behavior described in Message Queue Overrun. Suppressed message types are reported in the error message if they occur after overrun in addition to other discarded message types. Once triggered, message suppression will remain in effect until queue utilization drops to the threshold below the level selected to trigger suppression.

73.8.4 Nexus Message Priority

Nexus messages may be lost due to contention with other message types under the following circumstances:

- TBD - need examples
- More than three messages are generated in the same cycle

Table 73-28 lists the various message types and their relative priority from highest to lowest.

Up to three message requests can be queued into the message buffer in a given cycle. If more than three message requests exist in a given cycle, the three highest priority message classes are queued into the message buffer. The remaining messages that did not successfully queue into the message buffer in that cycle will generate subsequent responses, as detailed in Table 73-28.

The CPU is capable of completing two instructions per cycle. If multiple trace messages need to be queued at the same time, they will be queued with the following priority: Instruction 0 (oldest instruction) (WPM ->DQM -> PCM_{PIDMSG} -> OTM -> BTM -> DTM)-> Instruction1 (newer instruction) (WPM -> DQM -> OTM -> BTM -> DTM). As many as three messages may be simultaneously queued. Note that for the cycle following a dropped PTM, non-periodic OTM, or DQM message, only two other messages may be queued in addition to the dropped Error Message.

Watchpoint Messages from instructions that complete at the same time or events that occur during the same cycle will be combined.

Table 73-28. Message Type Priority and Message Dropped Responses

Message Type	Message	Priority	Message Dropped Response
Error	Error	0 (highest)	N/A[1, 1]
WP (Watchpoint Trace)	WPM (Watchpoint Message)	1	N/A[1]
DQ (Data Acquisition)	DQM (Data Acquisition Message)	2	DQM Error Message
Program Trace (PID MSG)	PCM - PID/NPIDR update (Program Correlation Message)	2	OTM Error Message
OT (Ownership)	OTM - PID/NPIDR update (Ownership Trace Message)	2	OTM Error Message[2]
Program Trace	BTM (Branch Trace Message)	2	BTM Error Message, Sync upgrade next BTM
	Sync (Program Sync Message)	3	Sync upgrade next BTM
	RFM (Resource Full for Instruction counter or history buffer)	4	BTM Error Message Sync upgrade next BTM
	DS (Debug Status Message)	5	Sync upgrade next BTM
	PCM (Program Correlation Message)	6	BTM Error Message Sync upgrade next BTM
DT (Data Trace)	DTM (Data Trace Message)	7	Sync upgrade next DTM
OT (Ownership)	OTM - Periodic update (Ownership Trace Message)	8 (lowest)	none

1. Error and Watchpoint messages are not dropped due to collisions, due to their priority.
2. Message will always be dropped if program trace is enabled, and program correlation messages for PID messages are not masked (Event Code = 0101). No error message is sent for this case since the PID value is contained in the higher priority message.

73.8.5 Data Acquisition Message Priority Loss Response

If a Data Acquisition Message (DQM) loses arbitration due to contention with higher priority messages, an error message will be generated to indicate that a DQM has been lost due to contention.

73.8.6 Ownership Trace Message Priority Loss Response

If an Ownership Trace message (OTM) due to software updates to the Process ID state loses arbitration due to contention with higher priority messages other than a program correlation message with EVCODE = 0101 (PID update), an error message will be generated to indicate that a OTM has been lost due to contention. If the pending OTM is a periodic update, the event is dropped without generating an error message.

73.8.7 Program Trace Message Priority Loss Response

If a Program Trace message (PTM) loses arbitration due to contention with higher priority messages, and the discarded PTM is a Program Correlation message, a Resource Full message for instruction count or history buffer, or a Branch Trace message, then an Error message is generated to indicate that branch trace information has been lost, and the next Branch Trace message will be upgraded to a sync-type message.

If the discarded PTM is a Program Correlation message with PID information (EVCODE=0101), the Error message will indicate a dropped OTM and a dropped Program Trace (Error code = xxxx11xx).

73.8.8 Program Trace Sync Message Priority Loss Response

If a Program Trace Sync message (SYNC) loses arbitration due to contention with higher priority messages, the Sync event is discarded, and the next Branch Trace message will be upgraded to a sync-type message.

73.8.9 Data Trace Message Priority Loss Response

If a Data Trace message (DTM) loses arbitration due to contention with higher priority messages, the DTM event is discarded, and the next DTM will be upgraded to a sync-type message.

73.9 Debug Status messages

Debug Status Messages report low power mode and debug status. Debug Status messages are enabled when Nexus 3 is enabled. Entering/exiting Debug Mode as well as entering, exiting, or changing Low Power Mode(s) will trigger a Debug Status message, indicating the value of the most significant byte in the Development Status register. Debug status information is sent out in the following format:

(8 bits)	(4 bits)	(6 bits)
DS[31:24]	Src. Proc.	TCODE (000000)

Fixed length = 18 bits

Figure 73-29. Debug Status message format

73.10 Error messages

Error messages are enabled whenever the debug logic is enabled. There are two conditions that will produce an error message, each receiving a separate error type designation:

- A message is discarded due to contention with other (higher priority) message types. These errors will have an Error Type value of 1.
- The message queue overruns. After the queue is drained, an error message is enqueued with an error code that indicates what types of messages were discarded during the interim. These errors will have an Error Type value of 0.

Note

The OVCR Register can be used in order to alleviate potential overrun situations.

Error information is messaged out in the following format (also see Table 73-3 and Table 73-4):

(6 bits)	(4 bits)	(4 bits)	(6 bits)
Error Code	Error Type	Src. Proc.	TCODE (001000)

Fixed length = 20 bits

Figure 73-30. Error message format

73.11 Ownership trace

This section details the ownership trace features of the Nexus 3 module.

73.11.1 Overview

Ownership trace provides a macroscopic view, such as task flow reconstruction, when debugging software written in a high level (or object-oriented) language. It offers the highest level of abstraction for tracking operating system software execution. This is especially useful when the developer is not interested in debugging at lower levels.

73.11.2 Ownership Trace Messaging (OTM)

Ownership trace information is messaged via the auxiliary port using an Ownership Trace Message (OTM). processors contain a *PowerISA 2.06* defined "Process ID" register within the CPU. It is updated by the operating system software to provide task/process ID information. The contents of this register are replicated on the pins of the processor and connected to Nexus. The Process ID (PID) register value can be accessed using the **mfspr/mtspr** instructions.

Note

The CPU includes a Process ID register (PID0), thus the Nexus UBA functionality is not implemented.

In addition, to decouple trace information from the PID0 register and to provide a full independent 32-bit process ID for debug use, the Nexus PID Register (NPIDR) may be used instead of PID0 to provide OTM process ID information. It is updated by the operating system software to provide task/process ID information. The Nexus Process ID (NPIDR) register value can be accessed using the **mfspr/mtspr** instructions. This register is accessible in user or supervisor mode.

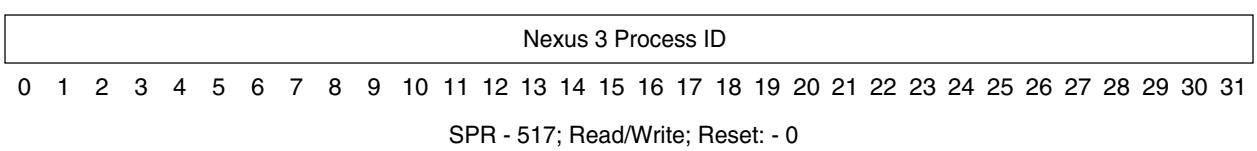


Figure 73-31. Nexus 3 Process ID Register (NPIDR)

Note

OS updates to NPIDR should be performed in addition to normal PID0 updates when a process switch occurs, in order to properly generate OTM messages with new process ID information when NPIDR is selected for OTM use.

The process ID source (PID0 or NPIDR) is selected by the setting of the $DC1_{OTS}$ control bit.

There are two conditions that will cause an Ownership Trace Message when Ownership Trace is enabled:

- When the PID0 register is written to by the processor and $DC1_{OTS}$ indicates PID0 should be used, or when the NPIDR register is written to by the processor and $DC1_{OTS}$ indicates NPIDR should be used, the data is latched within Nexus, and is messaged out via the auxiliary port, allowing development tools to trace ownership flow. However, if Program Trace is enabled, and program correlation messages for PID updates are not masked (Event Code = 0101), then an OTM will not be generated for an update to the selected process ID register, since the program correlation message will provide this process ID update information.
- Periodically, at least once every 256 messages, the most recent state of the selected process ID register is messaged out. The resulting Ownership Trace message will indicate in the PID Index sub-field that PID0/NPIDR status is being reported and the most recent value of the PID0/NPIDR register will be conveyed in the Process ID value sub-field. These periodic Ownership Trace message events can be disabled by setting $DC1_{POTD}$.

Ownership trace information is messaged out in the following format:

(1-32 bits)	(4 bits)	(4 bits)	(6 bits)
Process ID	PID Index (0000)	Src. Proc.	TCODE (000010)

Variable length = 15-46 bits

Figure 73-32. Ownership Trace Message format

73.12 Program trace

This section details the program trace mechanism supported by Nexus3 for the processor. Program trace is implemented via Branch Trace Messaging (BTM) as per the **IEEE-ISTO 5001** standard definition. Branch Trace Messaging for processors is accomplished by snooping the internal address bus (between the CPU and Cache), attribute signals, and CPU Status (**p_mode[0:3]**, **p_pstat_pipe{0,1}[0:5]**).

73.12.1 Branch Trace Messaging types

Traditional Branch Trace Messaging facilitates program trace by providing the following types of information:

- Messaging for taken direct branches includes how many sequential instructions were executed since the last taken branch or exception, including the taken direct branch. Branch instructions are included in the count of sequential instructions.
- Messaging for taken indirect branches and exceptions includes how many sequential instructions were executed since the last taken branch or exception and the unique portion of the branch target address or exception vector address. Branch instructions are included in the count of sequential instructions. For taken indirect branches that trigger generation of a message, the branch is also included in the count.

Branch History Messaging facilitates program trace by providing the following information.

- Messaging for taken indirect branches and exceptions includes a) how many sequential instructions (I-CNT) were executed since the last predicate instruction, taken/not taken direct branch, taken/not-taken indirect branch, or exception, b) the unique portion of the branch target address or exception vector address, and c) a branch/predicate instruction history field. Each bit in the history field represents a direct branch or predicated instruction where a value of one (1) indicates taken, and a value of zero (0) indicates not taken. Certain instructions (**zvselh**) generate a pair of predicate bits that are both reported as consecutive bits in the history field. Not-taken indirect branches will generate a history bit with a value of zero (0). Instructions that generate history bits are not included in instruction counts. For taken indirect branches that trigger generation of this message type, the branch is included in the count, but not in the history field.

73.12.1.1 Indirect Branch Message instructions

Table 73-29 shows the types of instructions and events that cause Indirect Branch Messages or Branch History Messages to be encoded.

Table 73-29. Indirect Branch Message sources

Source of Indirect Branch Message	Instructions / Detail
Taken branch relative to a register value	**bcctr, bcctrl, bclr, bclrl, se_bctr, se_bctrl, se_blr, se_blrl**
System Call / Trap exceptions taken	**se_sc, tw**
Return from interrupts / exceptions	**se_rfi, se_rfci, se_rfdi**
Exit from reset with Program Trace Enabled	**Indirect branch with Sync, target address is initial instruction, count = 1**

73.12.1.2 Direct Branch Message Instructions

Table 73-30 shows the types of instructions that will cause Direct Branch Messages or will toggle a bit in the instruction history buffer to be messaged out in a Resource Full Message or Branch History Message.

Table 73-30. Direct Branch Message sources

Source of Direct Branch Message	Instructions
Taken direct branch instructions Instruction Synchronize	**b, ba, bl, bla, bc, bca, bcl, bcla, se_b. se_bc, se_bl, e_b, e_bc, e_bl, e_bcl, se_isync**

73.12.1.3 BTM using Branch History Messages

Traditional BTM Messaging can accurately track the number of sequential instructions between branches, but cannot accurately indicate which instructions were conditionally executed, and which were not.

Branch History Messaging solves this problem by providing a predicated instruction history field in each Indirect Branch Message. Each bit in the history represents a predicated instruction or direct branch, or a not-taken indirect branch. A value of one (1) indicates the conditional instruction was executed or the direct branch was taken. A value of zero (0) indicates the conditional instruction was not executed or the branch was not taken. Certain instructions (**zvselh**) generate a pair of predicate bits that are both reported as consecutive bits in the history field.

Branch History Messages solve predicated instruction tracking and save bandwidth since only indirect branches cause messages to be queued.

73.12.1.4 BTM using Traditional Program Trace Messages

Based on the PTM bit in the DC1 Register, Program Tracing can utilize either Branch History Messages (PTM = 1) or traditional Direct/Indirect Branch Messages (PTM = 0).

Branch History will save bandwidth and keep consistency between methods of Program Trace, yet may lose temporal order between BTM messages and other types of messages. Since direct branches are not messaged, but are instead included in the history field of the Indirect Branch History Message, other types of messages may enter the FIFO between Branch History Messages. The development tool cannot determine the ordering of "events" that occurred with respect to direct branches simply by the order in which messages are sent out.

Traditional BTM messages maintain their temporal ordering because each event that can cause a message to be queued will enter the FIFO in the order it occurred and will be messaged out maintaining that order.

73.12.2 BTM Message For mats

The Nexus 3 block supports three types of traditional BTM Messages - Direct, Indirect, and Synchronization Messages. It supports two types of branch history BTM Messages - Indirect Branch History, and Indirect Branch History with Synchronization Messages.

73.12.2.1 Indirect branch messages (history)

Indirect branches include all taken branches whose destination is determined at run time, interrupts, and exceptions. If $DC1_{PTM}$ is set to '1', indirect branch information is messaged out in the following format:

(1-32 bits)	(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Branch History	Relative Address	Sequence Count	(0)	Source Proc.	TCODE (011100)

Max length = 83 bits; Min length = 14 bits

Figure 73-33. Indirect branch message (history) format

73.12.2.2 Indirect branch messages (traditional)

If $DC1_{PTM}$ is cleared to '0', indirect branch information is messaged out in the following format:

(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Relative Address	Sequence Count	(0)	Source Proc.	TCODE (000100)

Max length = 51 bits; Min length = 13 bits

Figure 73-34. Indirect branch message format

73.12.2.3 Direct branch messages (traditional)

Direct branches (conditional or unconditional) are all taken branches whose destination is fixed in the instruction opcode. Direct branch information is messaged out in the following format:

(1-8 bits)	(4 bits)	(6 bits)
Sequence Count	Src. Proc.	TCODE (000011)

Max length = 18 bits; Min length = 11bits

Figure 73-35. Direct Branch message format

Note

When $DC1_{PTM}$ is set, Direct Branch Messages will not be transmitted. Instead, each direct branch, not-taken indirect branch, or predicated instruction will be recorded in the history buffer.

73.12.3 Program Trace Message Fields

The following subsections describe specific fields used for Program Trace messages.

73.12.3.1 Sequential Instruction Count Field (ICNT)

Most of the program trace messages include an instruction count field. For traditional Branch messages, ICNT represents the number of sequential instructions including non-taken branches since the last Direct/Indirect Branch messages. Branch instructions that trigger message generation are included in the ICNT.

For Branch History messages, ICNT represents the number of instructions executed since the last taken/non-taken direct branch, predicate instruction, last taken/not-taken indirect branch, or exception. Branch instructions that trigger message generation are included in the ICNT. Instructions that generate history bits are not included in the ICNT.

The sequential instruction counter overflows after its value reaches 255 and is reset to 0. In addition, the next BTM Message (corresponding to the 256th or later instruction) will be converted to a w/sync type message.

The instruction counter is reset every time the instruction count is transmitted in a message or whenever there is a branch/predicate history event, as well as on exiting from debug mode.

73.12.3.2 Branch/Predicate Instruction History (HIST)

If $DC1_{PTM}$ is set, BTM messaging will use the Branch History format. The branch history (HIST) field in these messages provides a history of branch execution used for reconstructing the program flow. The branch/predicate history buffer stores information about branch and predicate instruction execution. The buffer is implemented as a left-shifting register. The buffer is preloaded with a one (1), which acts as a stop bit (the most significant 1 in the history field is a termination bit for the field). The pre-loaded bit itself is not part of the history, but is transmitted with the packet.

A value of one (1) is shifted into the history buffer for each taken direct branch (program counter relative branch) or predicate instruction whose condition evaluates to true. A value of zero (0) is shifted into the history buffer for each not-taken branch (including indirect branch instructions) or predicate instruction whose condition evaluates to false. For the **zvselh** instruction, two bits are shifted in, corresponding to the low element (shifted in first) and the high element (shifted in second) conditions.

This history buffer information is transmitted as part of an Indirect Branch with History message, as part of a Program Correlation message, or as part of a Resource Full message if the history buffer becomes full. The history buffer is reset every time the history information is transmitted in a message, as well as on exiting from debug mode.

Table 73-31. Branch/Predicate history events

Branch/Predicate History Event	History Bit(s)	Relevant Instructions
Not taken register indirect branches	0	**bcctr, bcctrl, bclr, bclrl**
Not taken direct branches	0	**b, ba, bc, bca, bla, bcla, bl, bcl**
Taken direct branches	1	**b, ba, bc, bca, bla, bcla, bl, bcl**[1]
zvselh instruction	00,01,10, or 11	**zvselh**

1. If the EVCODE for direct branch function calls is not masked in DC4, taken **bl** and **bcl** instructions will generate Program Correlation Messages and will not be logged in the history buffer.1

73.12.3.3 Execution Mode Indication

In order for a development tool to properly interpret instruction count and history information, it must be aware of the execution mode context of that information. VLE instructions will be interpreted differently from non-VLE instructions.

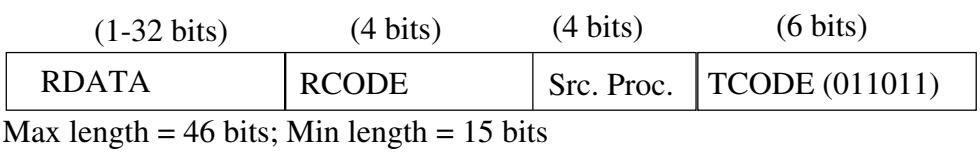
Program trace messages provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

73.12.4 Resource Full Messages

The Resource Full Message is used in conjunction with Branch Trace and Branch History Messages. The Resource Full Message is generated when either the internal branch/ predicate history buffer is full, or if the BTM Instruction sequence counter (I-CNT) overflows. If synchronization is needed at the time this message is generated, the synchronization is delayed until the next Branch Trace Message that is not a Resource Full Message.

For history buffer overflow, the Resource Full Message transmits a Resource Code (RCODE) of 0b0001 and the current contents of the history buffer, including the stop bit, are transmitted in the Resource Data (RDATA) field. This history information can be concatenated by the development tool with the branch/predicate history information from subsequent messages to obtain the complete branch/predicate history between indirect changes of flow.

For instruction counter overflow, the Resource Full Message transmits an RCODE of 0b0000 and a value of 0xFF is transmitted in the RDATA field, indicating that 255 sequential instructions have been executed since the last change of flow, or, if program trace is in history mode, since the last instruction that recorded history information.



Max length = 46 bits; Min length = 15 bits

Figure 73-36. Resource Full Message format

The following table shows the RCODE encodings and RDATA information used for Resource Full Messages.

Table 73-32. RCODE encoding

RCODE	Description	RDATA field
0000	Program Trace Instruction counter reached 255 and was reset.	0xFF
0001	Program Trace, Branch / Predicate Instruction History full.	Branch HIstory. This type of packet is terminated by a stop bit set to 1 after the last history bit.

73.12.5 Program Correlation Messages

Program Correlation Messages (PCMs) are used to correlate events to the program flow that may or may not be associated with the instruction stream. The following events will result in a PCM when program trace is enabled:

- When the CPU enters debug mode, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to debug mode entry.
- When the CPU first enters a low power mode in which instructions are no longer executed, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to low power mode entry.
- Whenever program trace is disabled by any means, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to disabling program trace.
- When a "Branch and Link" instruction executes (direct branch function call — **bl/bcl/bla/bcla**-type instructions)
- When program trace becomes masked due to MSR_{PMM}='0' and $DC4_{PTMARK}$='1'.
- When a write to the process ID register selected by $DC1_{OTS}$ (PID0 or NPIDR) is made via a **mtspr** PID0 or **mtspr** NPIDR.

Refer to Table 73-6 for the event codes that are supported in this implementation. Event code masking is available via the EVCDM field of the DC4 register to allow for control over generation of Program Correlation messages for each event type.

Program Correlation is messaged out in the following formats:

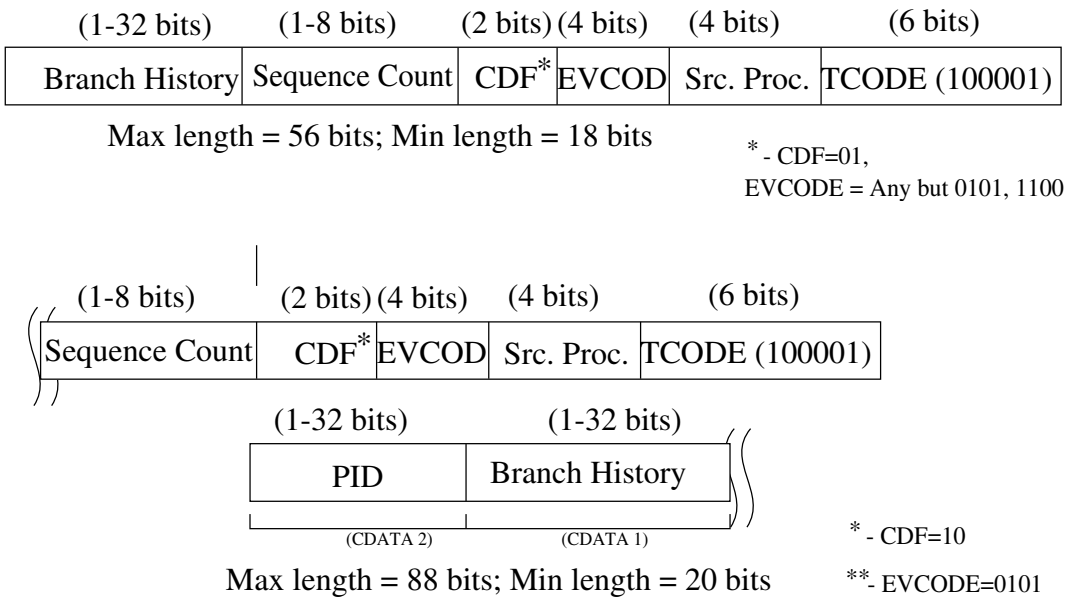


Figure 73-37. Program Correlation message formats

73.12.5.1 Program Correlation message generation for PID updates

When a (potentially) new value is established in the selected process ID register via a **mtspr** PID0/NPIDR, a Program Correlation message is generated containing the information regarding the new process ID value. This PCM also contains the current history and instruction count. The message is provided so that address translation information can be processed by the development tool in the proper program flow. The **mtspr** PID0/NPIDR is included in the instruction count information. Note that Ownership Trace Messages (other than the periodic OTM) are redundant with the information provided, and may be disabled to avoid unnecessary message bandwidth or collisions.

73.12.6 Program Trace Overflow Error messages

An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard incoming messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied.

73.12.7 Program Trace Synchronization messages

By default, program trace messages will perform XOR compression on the branch target address to produce the address field for the message. This compression is consistent with the specification in IEEE-ISTO 5001.

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see Table 73-33):

- Initial Program Trace message after exit from system reset or whenever program trace is enabled.
- Upon exiting from a CPU Low Power state.
- Upon exiting from Debug Mode.
- Upon occurrence of queue overrun (can be caused by any trace message), provided Program Trace is enabled.
- Upon assertion of the Event In (**nex_evti_b**) pin if the EIC bits within the DC1 Register have enabled this feature.
- When program trace becomes unmasked due to $MSR_{PMM} \rightarrow$ '1' with $DC4_{PTMARK}$='1'.

Note that the ICNT information for these messages is driven to zero, thus will not always be meaningful for some of these cases.

The format for Program Trace Synchronization messages is as follows:

(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Full Target Address	Seq. Count (0)	(0)	Source Proc.	TCODE (001001)

Max length = 51 bits; Min length = 13 bits

Exception conditions that result in Program Trace Synchronization are summarized in Table 73-33.

Table 73-33. Program Trace exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. Upon exiting system reset (**p_reset_b**), if Program Trace is already enabled, a Program Trace Sync Message is sent.
Program Trace Enabled	The first Program Trace Message (after Program Trace has been enabled or re-enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode, a Program Trace Sync Message is sent.

Table continues on the next page...

Table 73-33. Program Trace exception summary (continued)

Exception Condition	Exception Handling
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next BTM message in the queue will be a Program Trace Sync Message.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Program Trace Sync Message if Program Trace is enabled and the EIC bits of the DC1 Register have enabled this feature.

Note that for cases where program trace sync messages are generated due to program trace being enabled, the address contained in the sync message may either be the address of the instruction that caused program trace to be enabled, or may be the address of the first instruction of an exception handler that is executed in response to unsuccessful completion of that instruction.

73.12.8 Program Trace Direct/Indirect Branch with Sync messages

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization or a Direct/Indirect Branch with Sync message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see Table 73-33):

- Upon direct/indirect branch after a Program Sync Message was lost due to a collision while attempting to enter the message queue.
- Upon direct/indirect branch after the sequential instruction counter has expired indicating 255 instructions have occurred since the last change of flow.
- When the periodic program trace counter has expired indicating 255 *without-sync* Program Trace messages have occurred since the last *with-sync* message occurred.

Note that the ICNT and History information for the first message will not always be meaningful, since the temporary masking of program trace may result in ambiguous values. Subsequent w/sync messages will not have this issue.

The format for Program Trace Direct/Indirect Branch with Sync Messages is as follows:

(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Full Target Address	Sequence Count	(0)	Source Proc.	TCODE (001011 or 001100)

Max length = 51 bits; Min length = 13 bits

Figure 73-38. Direct/Indirect Branch with Sync message format

The format for Program Trace Indirect Branch History with Sync. messages is as follows:

(1-32 bits)	(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Branch History	Full Target Address	Sequence Count	(0)	Source Proc.	TCODE (011101)

Max length = 83 bits; Min length = 14 bits

Figure 73-39. Indirect Branch History w/ Sync. message format

Exception conditions that result in Program Trace Synchronization using a w/Sync message type are summarized in Table 73-34.

Table 73-34. Program Trace Exception Summary for w/Sync BTM messages

Exception Condition	Exception Handling
Periodic Program Trace Sync.	A forced synchronization occurs periodically after 255 non-sync Program Trace Messages have been queued. A Direct/Indirect Branch w/ Sync. Message is queued. The periodic program trace message counter then resets.
Sequential Instruction Count Overflow	After the sequential instruction counter reaches its maximum count (up to 255 sequential instructions may be executed), a forced synchronization occurs. The sequential counter then resets. A Program Trace Direct/Indirect Branch w/ Sync.Message is queued upon execution of the next branch. A Resource Full Message is Queued on the overflow event. If a branch instruction is the 255th instruction to occur, and causes a Program Trace message to be queued, then no Resource Full Message is queued, and the w/Sync message will be queued for the *next* Program Trace Direct/Indirect Branch Message.
Collision Priority	All Messages have the following priority: Instruction 0 (WPM -> DQM -> PCM_{PIDMSG} → OTM → BTM → DTM) → Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A BTM Message from Instruction1 that attempts to enter the queue at the same time as three higher priority messages from either instruction will be lost. An Error Message will be sent indicating the BTM was lost. The following direct/indirect branch will queue a Direct/Indirect Branch w/ Sync. Message. The count value within this message will reflect the number of sequential instructions executed after the last successful BTM Message was generated. This count will include the branch that did not generate a message due to the collision.

73.12.9 Enabling Program Trace

Program Trace Messaging can be enabled in one of three ways:

- Setting the TM field of the DC1 Register to enable Program Trace

- Using the PTS field of the WT Register to enable Program Trace on Watchpoint hits (watchpoints are configured within the CPU)
- Using the external hardware trace enable input signal (nex_ptm_starttr)
- Filtering of Program Trace messages may be performed using the MSR_{PMM} bit and the setting of $DC4_{PTMARK}$

73.12.10 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration)

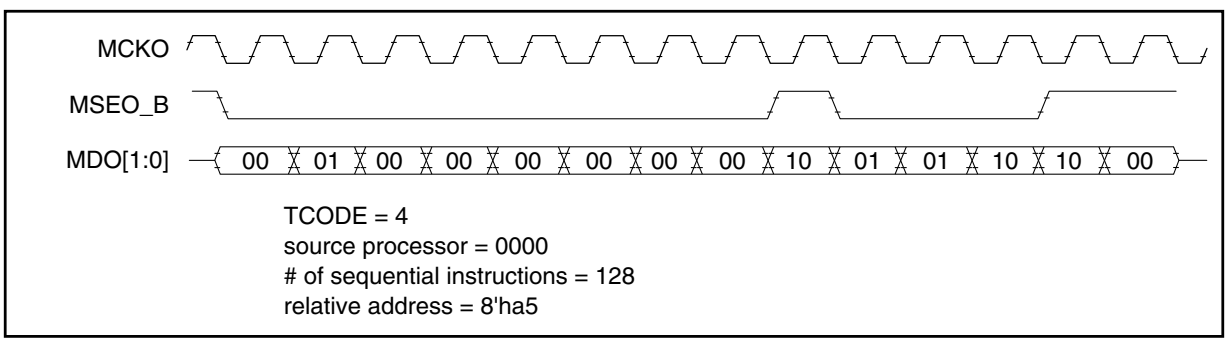


Figure 73-40. Program Trace - Indirect Branch message (Traditional)

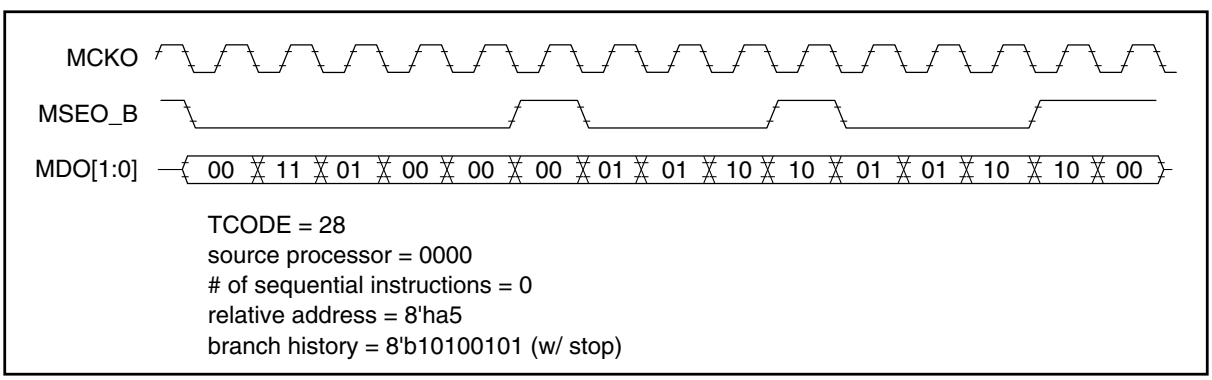


Figure 73-41. Program Trace - Indirect Branch message (history)

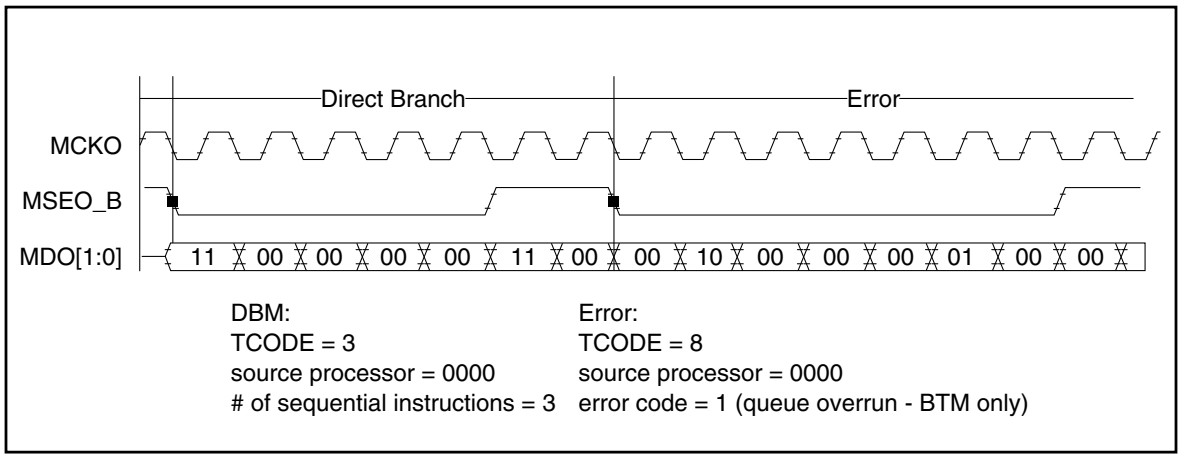


Figure 73-42. Program Trace - Direct Branch (traditional) & Error messages

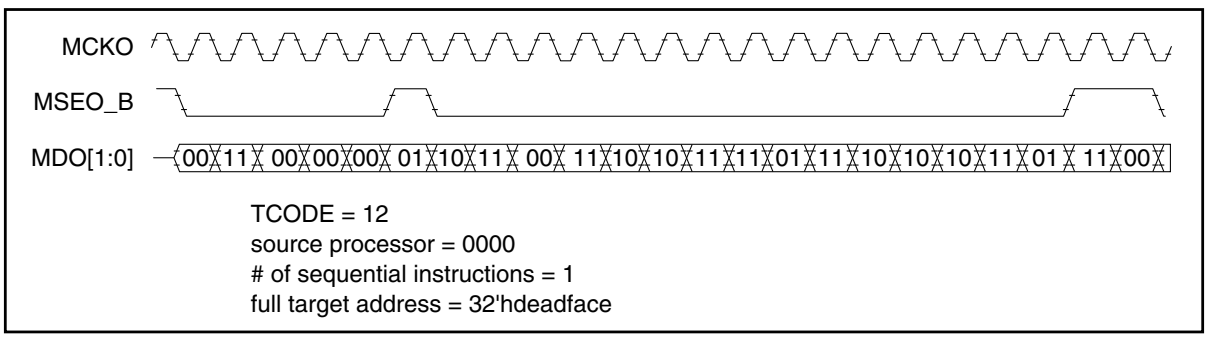


Figure 73-43. Program Trace - Indirect Branch w/ Sync. message

73.13 Data Trace

This section deals with the Data Trace mechanism supported by the Nexus 3 module. Data Trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the **IEEE-ISTO 5001** standard.

73.13.1 Data Trace Messaging (DTM)

Data Trace Messaging is accomplished by snooping the internal address and data buses, and storing the information for qualifying accesses (based on enabled features and matching target addresses). The Nexus 3 module traces all data access that meet the selected range and attributes.

Note

Data Trace is only performed on the internal data buses. This allows for data visibility for processors that incorporate a data cache. Only CPU initiated accesses will be traced. No DMA accesses to the AHB system bus will be traced.

Data Trace Messaging can be enabled in one of three ways.

- Setting the TM field of the DC1 Register to enable Data Trace.
- Using the DTS field of the WT Register to enable Data Trace on Watchpoint hits (watchpoints are configured within the Nexus1 module).
- Using the external hardware trace enable signal (**nex_dtm_starttr**)

73.13.2 DTM Message formats

The Nexus 3 block supports five types of DTM Messages — Data Write, Data Read, Data Write Synchronization, Data Read Synchronization and Error Messages.

73.13.2.1 Data Write messages

The Data Write message contains the data write value and the address of the write access, relative to the previous Data Trace Message. Data Write message information is messaged out in the following format:

(1-64 bits)	(1-32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000101)

Max length = 111 bits; Min length = 17 bits

Figure 73-44. Data Write message format

73.13.2.2 Data Read messages

The Data Read message contains the data read value and the address of the read access, relative to the previous Data Trace message. Data Read message information is messaged out in the following format:

(1-64 bits)	(1-32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000110)

Max length = 111 bits; Min length = 17 bits

Figure 73-45. Data Read fessage format

Note

* -based CPUs are capable of generating two (2) reads or writes per clock cycle in cases where multiple registers are accessed with a single instruction (lmw/stmw). These will have a double word pair size encoding (**p_tsiz** = 0b000). In these cases, the Nexus 3 module will send one (1) Data Trace Message with the two 32-bit data values as one combined 64-bit value for each message.

For -based CPUs, the doubleword encoding (**p_tsiz** = 0b000) may also indicate a doubleword access and will be sent out as a single Data Trace Message with a single 64-bit data value.

The debug/development tool will need to distinguish the two cases based on the family of processor.

73.13.2.3 Data Trace Synchronization messages

A Data Trace Write/Read with Sync. message is messaged via the auxiliary port (provided Data Trace is enabled) for the following conditions (see Table 73-35):

- Initial Data Trace Message after exit from system reset or whenever Data Trace is enabled.
- Upon returning from a CPU Low Power state.
- Upon returning from Debug Mode.
- After occurrence of queue overrun (can be caused by any trace message), provided Data Trace is enabled.
- After the periodic data trace counter has expired indicating 255 *without-sync* Data Trace messages have occurred since the last *with-sync* message occurred.
- Upon assertion of the Event In (**nex_evti_b**) pin, the first Data Trace Message will be a synchronization message if the EIC bits of the DC1 Register have enabled this feature.

- Upon Data Trace Write/Read after the previous DTM message was lost due to an attempted access to a secure memory location (for SOC's w/ security).
- Upon Data Trace Write/Read after the previous DTM message was lost due to a collision entering the FIFO between the DTM message and any two of the following: Watchpoint message, Ownership Trace message, or Program Trace message.

Data Trace Synchronization Messages provide the full address (without leading zeros) and insure that development tools fully synchronize with Data Trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the Data Trace address is transmitted. The format for Data Trace Write/Read with Sync. Messages is as follows:

(1-64 bits)	(1-32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value	Full Address	Data Size	(0)	Source Proc.	TCODE (001101 or 001110)

Max length = 111 bits; Min length = 17 bits

Figure 73-46. Data Write/Read with Sync. message format

Exception conditions that result in Data Trace Synchronization are summarized in Table 73-35.

Table 73-35. Data Trace Exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. If Data Trace is enabled, the first Data Trace Message is a Data Write/Read w/ Sync. Message.
Data Trace Enabled	The first Data Trace Message (after Data Trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode the next Data Trace Message will be converted to a Data Write/Read with Sync. Message.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue will be a Data Write/Read w/ Sync. Message.
Periodic Data Trace Sync.	A forced synchronization occurs periodically after 255 Data Trace Messages have been queued. A Data Write/Read w/ Sync. Message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Data Trace Write/Read w/ Sync. Message upon the next data write/read (if Data Trace is enabled and the EIC bits of the DC1 Register have enabled this feature).
Collision Priority	All Messages have the following priority: Instruction 0 (WPM → DQM → PCM_{PIDMSG} → OTM → BTM → DTM) → Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A DTM Message that attempts to enter the queue at the same time as three other higher priority messages will be lost. A subsequent read/write will queue a Data Trace Read/Write w/ Sync. Message.

73.13.3 DTM Operation

73.13.3.1 Data Trace windowing

Data Write/Read Messages are enabled via the RWT field in the Data Trace Control Register (DTC) for each DTM channel. Data Trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC field in the DTC register. All CPU initiated read/write accesses that fall inside or outside these address ranges, as programmed, are candidates to be traced.

73.13.3.2 Data Access / Instruction Access Data Tracing

The Nexus3 module is capable of tracing either instruction access data or data access data and can be configured for either type of data trace by setting the DI1 field within the Data Trace Control Register. This setting applies to all DTM channels.

73.13.3.3 Data Trace filtering

Data Trace filtering is available base on the settings of MSR_{PMM} and $DC4_{DTMARK}$.

73.13.3.4 Bus Cycle special cases

Table 73-36. Bus Cycle Special Cases

Special Case	Action
Bus cycle aborted	Cycle ignored
Bus cycle with data error ($\overline{TEA}$)[1, 1]	Data Trace Message discarded
Bus cycle completed without error[1]	Cycle captured & transmitted
AHB bus cycle initiated by Nexus 3	Cycle ignored
Bus cycle is an instruction fetch	Cycle selectively ignored based on DTC_{DI} setting
Bus cycle accesses misaligned data (across 64-bit boundary) - both 1st & 2nd transactions within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction within data trace range; 2nd transaction out of data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction within data trace range; 2nd transaction (regardless of within range or not) receives a bus error	Data Trace Message discarded
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction out of data trace range; 2nd transaction within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)

Table continues on the next page...

Table 73-36. Bus Cycle Special Cases (continued)

Special Case	Action
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction out of data trace range; 2nd transaction within range, receives a bus error	Data Trace Message discarded

1. Buffering of stores in the CPU store buffer may generate a DTM prior to the actual memory access, regardless of an error termination condition from memory.

Note

For misaligned accesses (crossing 64-bit boundary), the access is broken into two accesses by the CPU. If either access is within the data trace range, a single DTM will be sent with a size encoding indicating the size of the original access (i.e. word), and the address indicating the original misaligned accesses, unless the misaligned access wraps into the doubleword at address 0, or unless the misaligned access wraps over the end of a circular buffer when using the LSP specialized load or store with circular addressing. In this case, since the two portions of the misaligned access are not contiguous, two DTMs will be sent, one for each portion. The size encodings and the addresses of the DTMs will indicate the accessed bytes of data. The data trace port handles these cases in the same manner. (See the Data Trace Port section in the Core Debug Support chapter.)

Note

A store to the store buffer within the data trace range may initiate a DTM message prior to completion of the actual memory access.

73.13.4 Data Trace Timing Diagrams (8 MDO / 2 MSEO configuration)

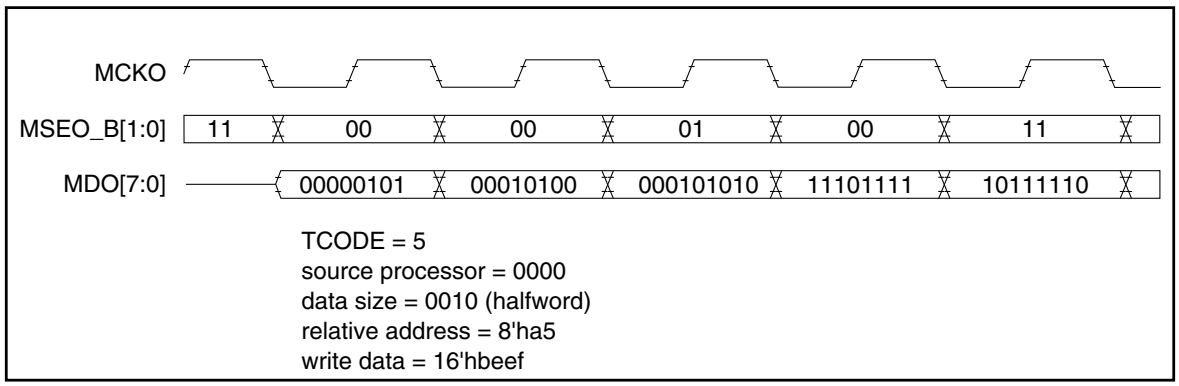


Figure 73-47. Data Trace - Data Write Message

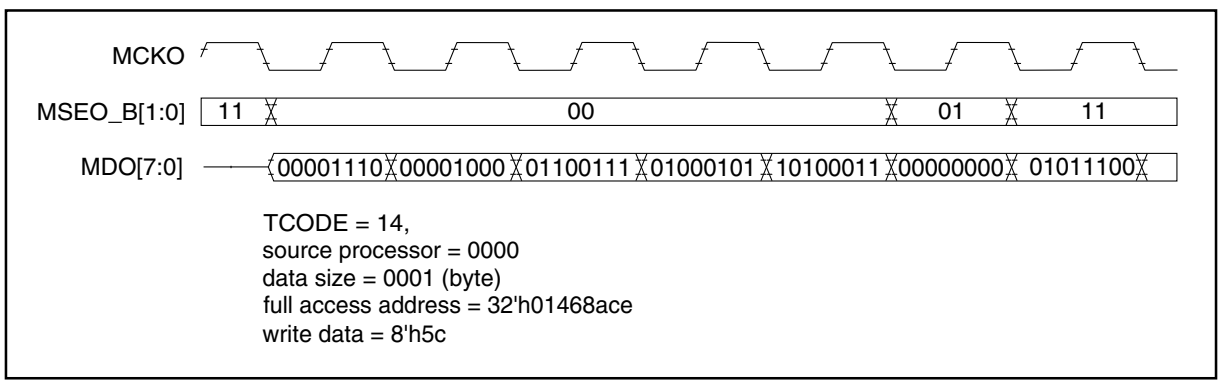


Figure 73-48. Data Trace — Data Read w/ Sync message

73.14 Data Acquisition messaging

This section details the Data Acquisition mechanisms supported by the Nexus 3 module. Data Acquisition Trace is implemented using Data Acquisition Trace Messages in accordance with IEEE-ISTO 5001 definitions. The control mechanism to export the data is different from the recommendations of the standard, however.

Data Acquisition Trace provides a convenient and flexible mechanism for the debugger to observe the architectural state of the machine through software instrumentation.

73.14.1 Data Acquisition ID Tag field

The DQTAG Tag field (DQTAG) is an 8-bit value specifying control or attribute information for the data included in the Data Acquisition message. DQTAG is sampled from $DEVENT_{DQTAG}$ when a write to DDAM is performed via **mtspr** operations. The usage of the DQTAG is left to the discretion of the development tool to be used in whatever manner is deemed appropriate for the application.

73.14.2 Data Acquisition Data field

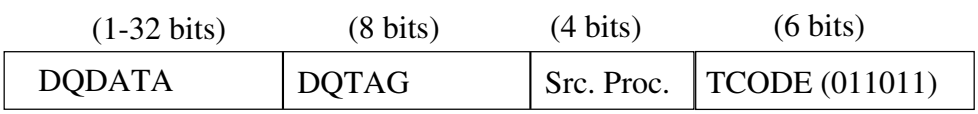
The Data Acquisition Data field (DQDATA) is the data captured from the DDAM write operation via **mtspr** operations. Leading zeros are omitted from the message.

73.14.3 Data Acquisition Trace event

For DQM, a dedicated SPR has been allocated (DDAM). It is expected that the general use case is to instrument the software and use **mtspr** operations to generate Data Acquisition messages.

There is no explicit error response for failed accesses as a result of contention between an internal and external debugger. Software may be blocked or given ownership of DDAM and the DQTAG field of the DEVENT register via control in EDBRAC0 while in External Debug Mode. Hardware always has access to these registers. Refer to the "External Debug Resource Allocation Control Register (EDBRAC0)" section in the Core Debug Support chapter for more detail on EDBRAC0.

Reads from the Data Acquisition channel do not generate a Data Acquisition event and will return zeroes for the read data.



Max length = 50 bits; Min length = 19 bits

Figure 73-49. Data Acquisition message format

73.15 Watchpoint Trace messaging

Enabling Watchpoint messaging is done by setting the Watchpoint Trace Enable bit in the DC1 Register. Setting the individual Watchpoint sources is supported through the Nexus1 module and the Performance Monitor unit. The Nexus1 module is capable of setting multiple types of watchpoints. Please refer to the Core Debug Support chapter for details on watchpoint initialization.

When watchpoints occur due to one or more asserted watchpoint event signals and Watchpoint Trace Messaging is enabled, a Watchpoint Trace message will be sent to the message queue to be messaged out. This message includes the watchpoint number indicating which watchpoint(s) caused the message. If more than one enabled watchpoint occurs in a single cycle, only one Watchpoint Trace message is generated and multiple bits of the watchpoint hit field will be set. The settings of the $WMSK_{WEM}$ field control which watchpoints are enabled to generate watchpoint trace messages.

The occurrence of any of the defined watchpoints can also be programmed to assert the Event Out (**nex_evto_b**) pin for one (1) period of the output clock (**nex_mcko**) based on settings in the DC2 and DC3 registers. See Table 73-40 for details on **nex_evto_b**.

Watchpoint information is messaged out in the following format:

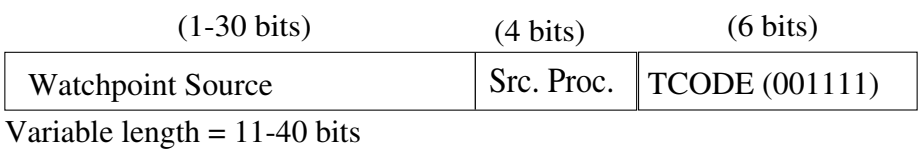


Figure 73-50. Watchpoint message format

The Watchpoint Source message field will contain a '1' for each asserted watchpoint. Leading zeros are truncated.

Table 73-37. Watchpoint Source Encoding

Watchpoint Source (1-30 bits)	Watchpoint Description
00000000000000000000000000000 - No Watchpoints enabled for Watchpoint Trace Messaging	
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 - Watchpoint #0 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXXXXX1X - Watchpoint #1 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXXXX1XX - Watchpoint #2 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXXX1XXX - Watchpoint #3 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXX1XXXX - Watchpoint #4 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXX1XXXXX - Watchpoint #5 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXX1XXXXXX - Watchpoint #6 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXX1XXXXXXX - Watchpoint #7 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXX1XXXXXXXX - Watchpoint #8 enabled for WTM	
XXXXXXXXXXXXXXXXXXXX1XXXXXXXXX - Watchpoint #9 enabled for WTM	
XXXXXXXXXXXXXXXXXXX1XXXXXXXXXX - Watchpoint #10 enabled for WTM	
XXXXXXXXXXXXXXXXXX1XXXXXXXXXXX - Watchpoint #11 enabled for WTM	
XXXXXXXXXXXXXXXXX1XXXXXXXXXXXX - Watchpoint #12 enabled for WTM	
XXXXXXXXXXXXXXXX1XXXXXXXXXXXXX - Watchpoint #13 enabled for WTM	
XXXXXXXXXXXXXXX1XXXXXXXXXXXXXX - Watchpoint #14 enabled for WTM	
XXXXXXXXXXXXXX1XXXXXXXXXXXXXXX - Watchpoint #15 enabled for WTM	
XXXXXXXXXXXXX1XXXXXXXXXXXXXXXX - Watchpoint #16 enabled for WTM	
XXXXXXXXXXXX1XXXXXXXXXXXXXXXXX - Watchpoint #17 enabled for WTM	
XXXXXXXXXXX1XXXXXXXXXXXXXXXXXX - Watchpoint #18 enabled for WTM	
XXXXXXXXXX1XXXXXXXXXXXXXXXXXXX - Watchpoint #19 enabled for WTM	
XXXXXXXXX1XXXXXXXXXXXXXXXXXXXX - Watchpoint #20 enabled for WTM	
XXXXXXXX1XXXXXXXXXXXXXXXXXXXXX - Watchpoint #21 enabled for WTM	
XXXXXXX1XXXXXXXXXXXXXXXXXXXXXX - Watchpoint #22 enabled for WTM	
XXXXXX1XXXXXXXXXXXXXXXXXXXXXXX - Watchpoint #23 enabled for WTM	
XXXXX1XXXXXXXXXXXXXXXXXXXXXXXX - Watchpoint #24 enabled for WTM	
XXXX1XXXXXXXXXXXXXXXXXXXXXXXXX - Watchpoint #25 enabled for WTM	
XXX1XXXXXXXXXXXXXXXXXXXXXXXXXX - Watchpoint #26 enabled for WTM	
XX1XXXXXXXXXXXXXXXXXXXXXXXXXXX - Watchpoint #27 enabled for WTM	
X1XXXXXXXXXXXXXXXXXXXXXXXXXXXX - Watchpoint #28 enabled for WTM	
1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX - Watchpoint #29 enabled for WTM	

73.15.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration)

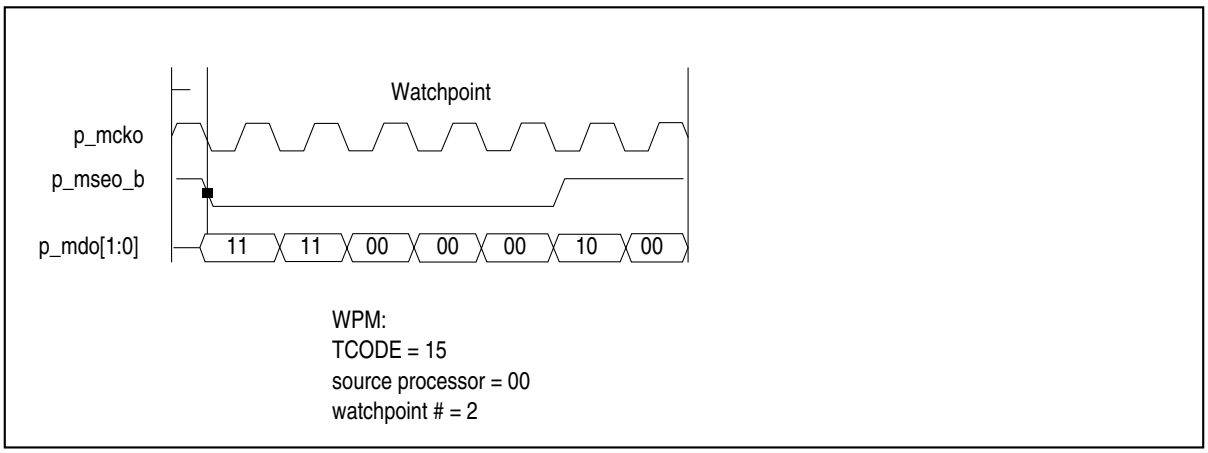


Figure 73-51. Watchpoint Message & Watchpoint Error message

73.16 External Hardware Trigger Controls

The Nexus 3 module provides a set of hardware trigger inputs to allow for external hardware to also independently control enabling/disabling of Program Trace, Data Trace, Ownership Trace, and Watchpoint Trace messaging. These signals can be used to produce a "window" for trace messaging for each type.

When a start trigger condition is detected for one of the trace types, and the corresponding trace type has not already been enabled by a software write to the DC1 register, the designated trace features become enabled, and the corresponding enable bit(s) of the DC1 register is set. When a stop trigger condition is detected and the corresponding trace type has not already been enabled by a software write to the DC1 register, the designated trace feature becomes disabled, and the corresponding enable bit of the DC1 register is cleared. If a simultaneous trigger condition is received for both start and stop triggering, then the designated trace feature will toggle between enabled and disabled depending on the enable state at the time of the trigger events. For example, if tracing is enabled, and a start and stop trigger condition occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature control action that is derived from external hardware triggering, and if the corresponding trace feature has been enabled by a direct write to DC1, then the corresponding hardware trigger signals will have no effect. The following table details the external hardware trigger signals.

Table 73-38. External Hardware Trigger Signals

Name	Description
nex_dtm_starttr	Data Trace Messaging start trigger input. A transition from 0 → 1 on this input signals a start trace condition.
nex_dtm_stoptr	Data Trace Messaging stop trigger input. A transition from 0 → 1 on this input signals a stop trace condition.
nex_otm_starttr	Ownership Trace Messaging start trigger input. A transition from 0 → 1 on this input signals a start trace condition.
nex_otm_stoptr	Ownership Trace Messaging stop trigger input. A transition from 0 → 1 on this input signals a stop trace condition.
nex_ptm_starttr	Program Trace Messaging start trigger input. A transition from 0 → 1 on this input signals a start trace condition.
nex_ptm_stoptr	Program Trace Messaging stop trigger input. A transition from 0 → 1 on this input signals a stop trace condition.
nex_wtm_starttr	Watchpoint Trace Messaging start trigger input. A transition from 0 → 1 on this input signals a start trace condition.
nex_wtm_stoptr	Watchpoint Trace Messaging stop trigger input. A transition from 0 → 1 on this input signals a stop trace condition.

These signals are active-high inputs and are transition sensitive. A 0 → 1 transition on the signal indicates the corresponding condition is requested. These signals are not synchronized, and must meet setup and hold requirements relative to **nex_clk**.

73.17 Nexus 3 Read/Write access to memory-mapped resources

The Read/Write access feature allows access to memory-mapped resources via the JTAG/ OnCE port. The Read/Write mechanism supports single as well as block reads and writes to AHB resources.

The Nexus 3 module is capable of accessing resources on the system bus (AHB). Memory-mapped registers and other non-cached memory can be accessed via the standard memory map settings.

All accesses are setup and initiated by the Read/Write Access Control/Status Register (RWCS), as well as the Read/Write Access Address (RWA) and Read/Write Access Data Registers (RWD). Nexus 3 read/write accesses are run as privileged data non-cacheable accesses by default, and drive the **p_d_hprot[5:0]** bus access attributes to 6'b000011 accordingly. The $RWCS_{ATTR}$ field is provided to allow a portion of these default values to be modified when performing read or write accesses using the Nexus 3 Read/Write access mechanism.

Using the Read/Write Access Registers (RWCS/RWA/RWD), memory mapped AHB resources can be accessed through Nexus 3. The following subsections describe the steps required to access memory-mapped resources.

Note

Read/Write Access can only access memory mapped resources when system reset is de-asserted and clocks are running.

Misaligned accesses are NOT supported in the Nexus 3 module.

Uncorrectable ECC errors on Nexus 3 read access will result in the RWD register being updated with the raw data received.

73.17.1 Single write access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Write Address → 32h'xxxxxxxx (write address)
2. Initialize the Read/Write Access Control/Status Register (RWCS) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure the bits as follows:
   - Access Control (AC) → 1b'1 (to indicate start access)
   - Map Select (MAP) → 3b'000 (primary memory map)
   - Access Priority (PR) → 2b'11 (highest priority)
   - Read/Write (RW) → 1b'1 (write access)
   - Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
   - Access Count (CNT) → 14h'0000 or 14h'0001(single access)

   Note

   Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. Initialize the Read/Write Access Data Register (RWD) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Write Data → 32h'xxxxxxxx (write data)

4. The Nexus block will then arbitrate for the AHB system bus and transfer the data value from the data buffer RWD Register to the memory mapped address in the Read/Write Access Address Register (RWA). The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed, Nexus clears the AC and DV bits in the RWCS Register. If the access has completed without error, (ERR=1'b0), Nexus asserts the **nex_rdy_b** pin (see Table 73-40 for details) and clears the ERR bit in the RWCS Register. Otherwise, if the access completes with an error, the **nex_err_b** pin will be asserted and the ERR bit will be set to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once DV has been cleared, this indicates that the device is ready for the next access, or that an error has occurred.

Note

Only the **nex_ahb_start, nex_rdy_b,** and **nex_err_b** pins as well as the AC, DV, and ERR bits within the RWCS provide Read/Write Access status to the external development tool.

73.17.2 Block write access

1. For a block write access, follow Steps 1, 2, and 3 outlined in Single write access to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS Register.
2. The Nexus block will then arbitrate for the AHB system bus and transfer the first data value from the RWD Register to the memory mapped address in the Read/Write Access Address Register (RWA). The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, and the **nex_rdy_b** pin is asserted. If the access has completed without error, Nexus clears the ERR and DV bits in the RWCS Register, otherwise the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the **nex_err_b** pin is asserted, the block transfer is aborted, and the AC bit in the RWCS register is cleared. Once DV has been cleared, this indicates that the device is ready for the next access in the block transfer, or an error has occurred.

3. If AC has not been cleared due to an error, repeat Step 3 in Single write access until the internal CNT value is zero (0). When this occurs, the AC bit within the RWCS will be cleared to indicate the end of the block write access.

Note

The actual RWA value as well as the CNT field within the RWCS are not changed when executing a block write access. The original values can be read by the external development tool at any time.

73.17.3 Single read access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Read Address → 32h'xxxxxxxx (read address)
2. Initialize the Read/Write Access Control/Status Register (RWCS) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure the bits as follows:
   - Access Control (AC) → 1b'1 (to indicate start access)
   - Map Select (MAP) → 3b'000 (primary memory map)
   - Access Priority (PR) → 2b'11 (highest priority)
   - Read/Write (RW) → 1b'0 (read access)
   - Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
   - Access Count (CNT) → 14h'0000 or 14h'0001(single access)

Note

Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD Register. The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS Register and asserts the **nex_rdy_b** pin (see Table 73-40 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin

will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.

4. The data can then be read from the Read/Write Access Data Register (RWD) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE.

Note

Only the **nex_ahb_start, nex_rdy_b,** and **nex_err_b** pins as well as the AC, DV and ERR bits within the RWCS provide Read/Write Access status to the external development tool.

73.17.4 Block read access

1. For a block read access, follow Steps 1 and 2 outlined in Single read access to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS Register.
2. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD Register. The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS Register and asserts the **nex_rdy_b** pin (see Table 73-40 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.
3. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, Nexus clears the ERR bit and sets the DV bit in the RWCS Register, and asserts the **nex_rdy_b** pin (see Table 73-40 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the AC bit is cleared and the block transfer is aborted, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that

the device is ready for the next access, or that an error has occurred. Once DV has been set, this indicates that the device is ready for the next access in the block transfer, or if ERR is set (AC will be cleared), the block transfer has been aborted.

4. The data can then be read from the Read/Write Access Data Register (RWD) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE.
5. If AC has not been cleared due to an error, repeat Steps 3 and 4 in Single read access until the CNT value is zero (0). When this occurs, the AC bit within the RWCS is cleared to indicate the end of the block read access.

Note

The data values must be shifted out 32 bits at a time LSB first (i.e. doubleword read = two word reads from the RWD).

Note

The actual RWA value as well as the CNT field within the RWCS are not changed when executing a block read access. The original values can be read by the external development tool at any time.

73.17.5 Error handling

The Nexus 3 module handles various error conditions as follows:

73.17.5.1 AHB Read/Write error

All address and data errors that occur on read/write accesses to the AHB system bus will return a transfer error encoding on the **p_hresp[1:0]** signals. If this occurs:

1. The access is terminated without retrying (AC bit is cleared)
2. The ERR bit in the RWCS Register is set
3. The Error Message is sent (TCODE = 8) indicating Read/Write error

73.17.5.2 Access termination

The following cases are defined for sequences of the Read/Write protocol that differ from those described in the above sections.

1. If the AC bit in the RWCS Register is set to start Read/Write accesses and invalid values are loaded into the RWD and/or RWA, then an AHB access error may occur. This is handled as described above.
2. If a block access is in progress (all cycles not completed), and the RWCS Register is written, then the original block access is terminated at the boundary of the nearest completed access.
    a. If the RWCS is written with the AC bit set, the next Read/Write access will begin and the RWD can be written to/ read from.
    b. If the RWCS is written with the AC bit cleared, the Read/Write access is terminated at the nearest completed access. This method can be used to break (early terminate) block accesses.

73.17.6 Read/Write access error message

The Read/Write Access Error Message is sent out when an AHB system bus access error (read or write) has occurred.

Error information is messaged out in the following format:

(5 bits)	(4 bits)	(6 bits)
Error Code (00011)	Src. Proc.	TCODE (001000)

Fixed length = 15 bits

Figure 73-52. Error message format1

73.18 Nexus 3 pin interface

This section details information regarding the Nexus 3 pins and pin protocol.

The Nexus 3 pin interface provides the function of transmitting messages from the messages queues to the external tools. It is also responsible for handshaking with the message queues.

73.18.1 Pins implemented

The Nexus 3 module implements an auxiliary port consisting of one (1) **nex_evti_b** and one (1) **nex_mseo_b** or two (2) **nex_mseo_b[1:0]**. It also implements a configurable number of **nex_mdo[n:0]** pins, (1) **nex_rdy_b** pin, (1) **nex_err_b** pin, (1) **nex_evto_b** pin, (4) **nex_wevto[3:0]** pins, and one (1) clock output pin (**nex_mcko**), as well as additional configuration pins described in Table 73-40. The output pins are synchronized to the Nexus 3 output clock (**nex_mcko**).

All Nexus 3 input functionality may be controlled through the JTAG/OnCE port in compliance with IEEE 1149.1 (see Nexus 3 Register Access via JTAG/OnCE for details). The JTAG pins are incorporated as I/O to the processor, and are further described in the "JTAG/OnCE Pins" section of the Core Debug Support chapter.

Table 73-39. JTAG Pins for Nexus 3

JTAG Pins	Input/ Output	Description of JTAG Pins (included in Nexus 1)
j_tdo	O	The Test Data Output (**j_tdo**) pin is the serial output for test instructions and data. **j_tdo** is three-stateable and is actively driven in the "Shift-IR" and "Shift-DR" controller states. **j_tdo** changes on the falling edge of **j_tclk**.
j_tdi	I	The Test Data Input (**j_tdi**) pin receives serial test instruction and data. TDI is sampled on the rising edge of **j_tclk**.
j_tms	I	The Test Mode Select (**j_tms**) input pin is used to sequence the OnCE controller state machine. **j_tms** is sampled on the rising edge of **j_tclk**.
j_tclk	I	The Test Clock (**j_tclk**) input pin is used to synchronize the test logic, and control register access through the JTAG/OnCE port.
j_trst_b	I	The Test Reset (**j_trst_b**) input pin is used to asynchronously initialize the JTAG/OnCE controller.

The auxiliary pins are used to send and receive messages and are described in Table 73-40.

Table 73-40. Nexus 3 Auxiliary pins

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_mcko	O	Message Clock Out (**nex_mcko**) is a free running output clock to development tools for timing of **nex_mdo[n:0]** & **nex_mseo_b[1:0]** pin functions. **nex_mcko** is programmable through the DC1 Register.
nex_mdo[n:0]	O	Message Data Out (**nex_mdo[n:0]**) are output pins used for OTM, BTM, and DTM. External latching of **nex_mdo[n:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**).
nex_mseo_b[1:0]	O	Message Start/End Out (**nex_mseo_b[1:0]**) are output pins that indicate when a message on the **nex_mdo[n:0]** pins has started, when a variable length packet has ended, and when the message has ended. External latching of **nex_mseo_b[1:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**). One or two pin MSEO functionality is determined at integration time per SOC implementation.

Table continues on the next page...

Table 73-40. Nexus 3 Auxiliary pins (continued)

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_ahb_start	O	AHB Start (**nex_ahb_start**) is an output pin used to indicate to the external tool or SoC that the Nexus block is requesting the next Read/Write Access on the system bus. If Nexus is enabled, this signal is asserted upon an acknowledged request to start an AHB system bus transfer (Nexus read or write) and is pulsed asserted for one **nex_clk** clock period, corresponding to the first clock of the address phase of the transfer. Upon exit from system reset or if Nexus is disabled, **nex_ahb_start** remains de-asserted.
nex_rdy_b	O	Ready (**nex_rdy_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access. If Nexus is enabled, this signal is asserted upon successful (without error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the "Capture_DR" state. Upon exit from system reset or if Nexus is disabled, **nex_rdy_b** remains deasserted.
nex_err_b	O	Error (**nex_err_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access and an error has occurred on the previous access. If Nexus is enabled, this signal is asserted upon unsuccessful (with error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the "Capture_DR" state. Upon exit from system reset or if Nexus is disabled, **nex_err_b** remains deasserted.
nex_evto_b	O	Event Out (**nex_evto_b**) is an output that, when asserted, indicates one of two events has occurred based on the EOC bits in the DC1 Register. **nex_evto_b** is held asserted for one (1) cycle of **nex_mcko**: 1. One (or more) watchpoints has occurred (from Nexus1) & EOC = 2'b00 2. Debug mode was entered (jd_debug_b asserted from Nexus1) & EOC = 2'b01
nex_evti_b	I	Event In (**nex_evti_b**) is an input that, when asserted, will initiate one of two events based on the EIC bits in the DC1 Register (if the Nexus module is enabled at reset): 1. Program Trace & Data Trace synchronization messages (provided Program Trace & Data Trace are enabled & EIC = 2'b00). 2. Debug request to Nexus1 module (provided EIC = 2'b01 and this feature is implemented).
nex_wevto[3:0]	O	Watchpoint Event Out 3:0 (**nex_wevto[3:0]**) are outputs that, when asserted, indicates one or more watchpoint events has occurred based on the settings in the DC2 and DC3 registers. **nex_wevto[3:0]** is held asserted for one (1) cycle of **nex_mcko**.
nex_ext_src_id[0:3]	I	**nex_ext_src_id[0:3]** is used to provide the SRC field value used in each message. These pins are tied to a predetermined value at SoC integration time.
nex_sfwcntl_en]	I	**nex_sfwcntl_en** is used to allow software to control the module resources via the DCR register mappings. SoC logic should drive this signal appropriately in a semi-static manner based on the presence of an external hardware debugger and appropriate security precautions.

The Nexus auxiliary port arbitration pins are used when the Nexus 3 module is implemented in a multi-Nexus SoC that shares a single auxiliary output port. The arbitration is controlled by an SoC level Nexus Aurora Router (NAR). Refer to Auxiliary port arbitration for detail on Nexus port arbitration.

Table 73-41. Nexus port arbitration signals

Nexus port arbitration pins	Input/ Output	Description of arbitration pins
nex_aux_req[1:0]	O	Nexus Auxiliary Request (**nex_aux_req[1:0]**) output signals indicate to an SoC level Nexus arbiter a request for access to the shared Nexus auxiliary port in a multi-Nexus implementation. The priority encodings are determined by how many messages are currently in the message queues (See Table 73-43.)
nex_aux_busy	O	Nexus Auxiliary Busy (**nex_aux_busy**) is an output signal to an SoC level Nexus arbiter indicating that the Nexus 3 module is currently transmitting its message after being granted the Nexus auxiliary port.
nar_aux_grant	I	Nexus Auxiliary Grant **(nar_aux_grant)** is an input from the SoC level NAR that the auxiliary port has been granted to the Nexus 3 module to transmit its message.
ext_multi_nex_sel	I	Multi-Nexus Select (**ext_multi_nex_sel**) is a static signal indicating that the Nexus 3 module is implemented within a multi-Nexus environment. If set, port control and arbitration is controlled by the SoC level arbitration module (NAR).

73.18.2 Pin protocol

The protocol for the processor transmitting messages via the auxiliary pins is accomplished with the MSEO pin function outlined in Table 73-42. Both single and dual pin cases are shown.

nex_mseo_b[1:0] is used to signal the end of variable-length packets, and not fixed length packets. **nex_mseo_b[1:0]** is sampled on the rising edge of the Nexus 3 clock (**nex_mcko**).

Single pin MSEO is not supported on the .

Table 73-42. MSEO pin(s) protocol

nex_mseo_b function	Single nex_mseo_b data (serial)	Dual nex_mseo_b[1:0] data
Start of message	1-1-0	11-00
End of message	0-1-1-(more 1's)	00 (or 01)-11-(more 1's)
End of variable length packet	0-1-0	00-01
Message transmission	0's	00's
Idle (no message)	1's	11's

Figure 73-53 illustrates the state diagram for single pin MSEO transfers.

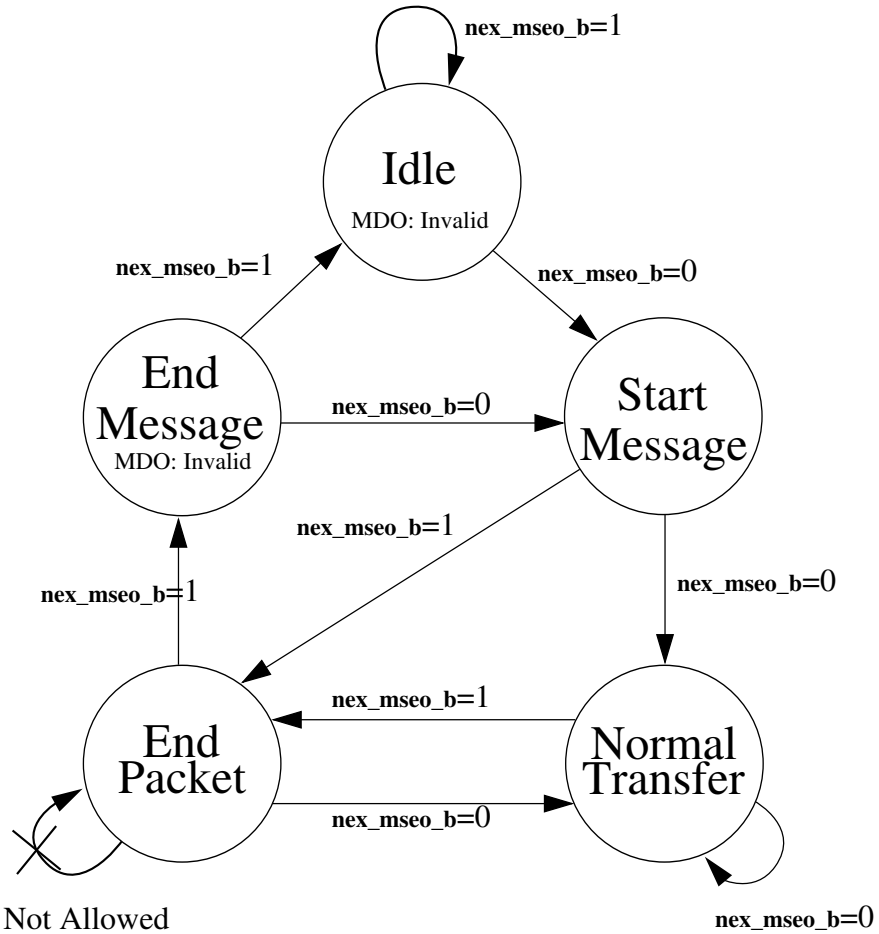


Figure 73-53. Single pin MSEO transfers

Note that the "End Message" state does not contain valid data on the **nex_mdo[n:0]** pins. Also, It is not possible to have two consecutive "End Packet" messages. This implies the minimum packet size for a variable length packet is 2x the number of **nex_mdo[n:0]** pins. This ensures that a false end of message state is not entered by emitting two consecutive '1's on the **nex_mseo_b** pin before the actual end of message.

Figure 73-54 illustrates the state diagram for dual pin MSEO transfers.

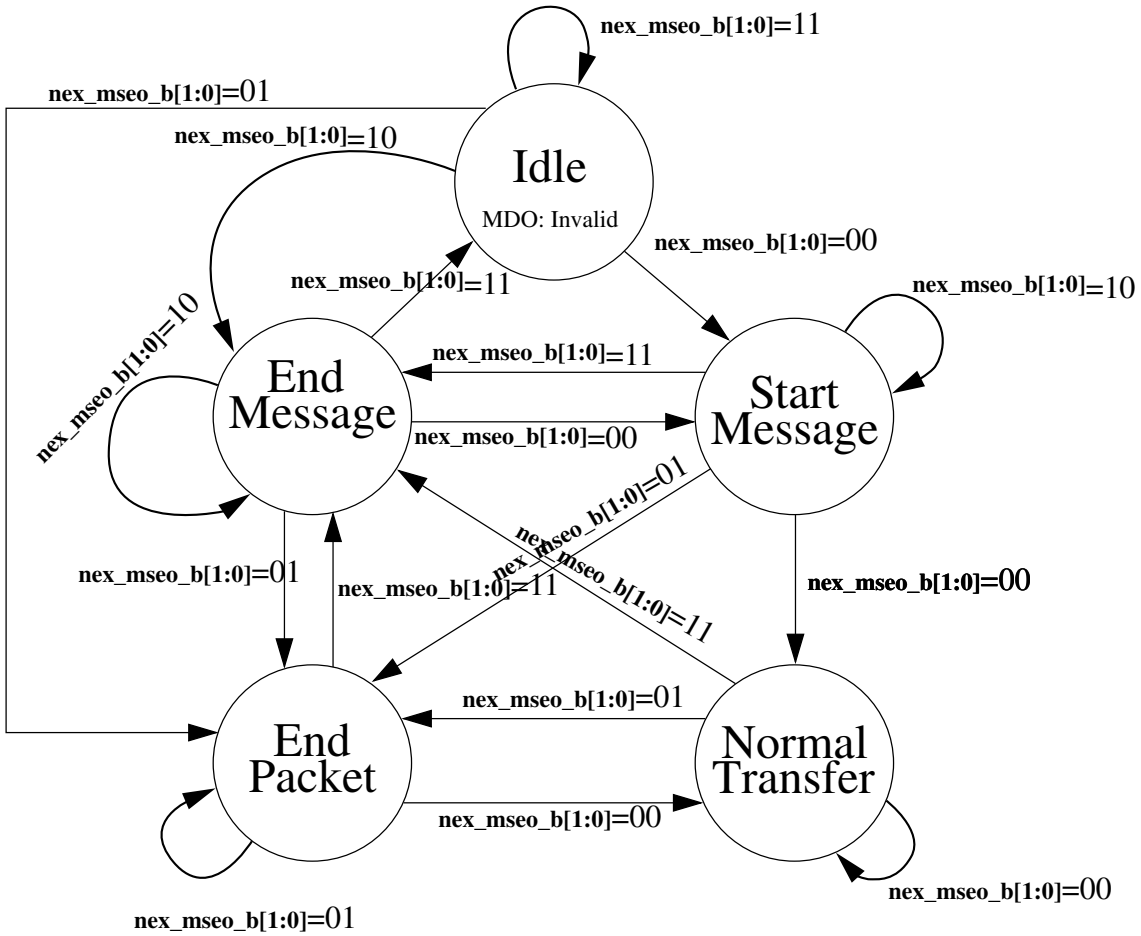


Figure 73-54. Dual pin MSEO transfers

The dual pin MSEO option is more robust that the single pin option. Termination of the current message may immediately be followed by the start of the next message on the consecutive clocks. An extra clock to end the message is not necessary as with the one MSEO pin option. The dual pin option also allows for consecutive "End Packet" states. This can be an advantage when small, variable sized packets are transferred.

Note

The "End Message" state may also indicate the end of a variable-length packet as well as the end of the message when using the dual pin option.

73.19 Rules for output messages

Class 3 compliant embedded processors must provide messages via the auxiliary port in a consistent manner as described below:

- A variable-sized packet within a message must end on a port boundary.
- A variable-sized packet may start within a port boundary only when following a fixed length packet. (If two variable-sized packets end and start on the same clock, it is impossible to know which bit is from the last packet and which bit is from the next packet.)
- Whenever a variable-length packet is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest-order bit so that it can end on a port boundary.

  For example, if the **nex_mdo[n:0]** port is 2 bits wide, and the unique portion of an indirect address TCODE is 5 bits, then the remaining 1 bit of **nex_mdo[n:0]** must be packed with a 0.

73.20 Auxiliary port arbitration

In a multi-Nexus environment, the Nexus 3 module must arbitrate for the shared Nexus port at the SoC level.The request scheme is implemented as a 2-bit request with various levels of priority. The priority levels are defined in Table 73-43 below. The Nexus 3 module will receive a 1-bit grant signal (**nar_aux_grant**) from the SoC level arbiter. When a grant is received, the Nexus 3 module will begin transmitting its message following the protocol outlined in Pin protocol. The Nexus 3 module will maintain control of the port, by asserting the **nex_aux_busy** signal, until the $\overline{\text{MSEO}}$ state machine reaches the "End Message" state.

Table 73-43. MDO Request Encodings

Request level	MDO request encoding (nex_aux_req[1:0])	Condition of queue
No Request	00	No message to send
Low Priority	01	Message queue less than 1/2 full
—	10	Reserved
High Priority	11	Message queue 1/2 full or more

73.21 Examples

The following are examples of Program Trace and Data Trace Messages.

Table 73-44 illustrates an example Indirect Branch Message with 2 MDO / 1MSEO configuration. Table 73-45 illustrates the same example with an 8 MDO / 2 MSEO configuration.

Note that T0 and S0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Ix = Number of instructions (variable)
- Ax = Unique portion of the address (variable)

Note that during clock 13, the **nex_mdo[n:0]** pins are ignored in the single MSEO case.

Table 73-44. Indirect Branch Message Example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I0	MAP	0	Normal Transfer
7	I2	I1	0	Normal Transfer
8	I4	I3	1	End Packet
9	A1	A0	0	Normal Transfer
10	A3	A2	0	Normal Transfer
11	A5	A4	0	Normal Transfer
12	A7	A6	1	End Packet
13	0	0	1	End Message
14	T1	T0	0	Start Message

Table 73-45. Indirect branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table continues on the next page...

Table 73-45. Indirect branch message example (8 MDO / 2 MSEO) (continued)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
2	I4	I3	I2	I1	I0	MAP	S3	S2	0	1	End Packet
3	A7	A6	A5	A4	A3	A2	A1	A0	1	1	End Packet/End Message
4	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 73-46 and Table 73-47 illustrate examples of Direct Branch Messages: one with 2 MDO / 1 MSEO, and one with 8 MDO / 2 MSEO.

Note that T0 and I0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- Ix = Number of Instructions (variable)

Table 73-46. Direct branch message example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I1	I0	1	End Packet
7	0	0	1	End Message

Table 73-47. Direct branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	0	0	0	0	I1	I0	S3	S2	1	1	End Packet/End Message
3	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 73-48 illustrates an example Data Write Message with 8 MDO / 1 MSEO configuration, and Table 73-49 illustrates the same DWM with 8 MDO / 2 MSEO configuration

Note that T0, A0, D0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Zx = Data size (fixed)
- Ax = Unique portion of the address (variable)
- Dx = Write data (variable 8-, 16- or 32-bit)

Table 73-48. Data write message example (8 MDO / 1 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b	State
0	X	X	X	X	X	X	X	X	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	0	Normal Transfer
4	0	0	0	0	0	0	0	0	1	End Packet
5	0	0	0	0	0	0	0	0	1	End Message

Table 73-49. Data write message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	0	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	1	1	End Packet/ End Message

73.22 Electrical characteristics

For all electrical characteristics related to core processor and Nexus 3 operation, please refer to the Data Sheet.

73.23 IEEE 1149.1 (JTAG) RD/WR sequences

This section contains example JTAG/OnCE sequences used to access resources.

73.23.1 JTAG sequence for accessing internal Nexus registers

Table 73-50. Accessing internal Nexus 3 registers via JTAG/OnCE

Step #	TMS Pin	Description
1	1	IDLE -> SELECT-DR_SCAN
2	0	SELECT-DR_SCAN -> CAPTURE-DR (Nexus Command Register value loaded in shifter)
3	0	CAPTURE-DR -> SHIFT-DR
4	0	(7) TCK clocks issued to shift in direction (rd/wr) bit and first 6 bits of Nexus reg. addr.
5	1	SHIFT-DR -> EXIT1-DR (7th bit of Nexus reg. shifted in)
6	1	EXIT1-DR -> UPDATE-DR (Nexus shifter is transferred to Nexus Command Register)
7	1	UPDATE-DR -> SELECT-DR_SCAN
8	0	SELECT-DR_SCAN -> CAPTURE-DR (Register value is transferred to Nexus shifter)
9	0	CAPTURE-DR -> SHIFT-DR
10	0	(31) TCK clocks issued to transfer register value to TDO pin while shifting in TDI value
11	1	SHIFT-DR -> EXIT1-DR (MSB of value is shifted in/out of shifter)
12	1	EXIT1-DR -> UPDATE -DR (if access is write, shifter is transferred to register)
13	0	UPDATE-DR -> RUN-TEST/IDLE (transfer complete - Nexus controller to Reg. Select state)

73.23.2 JTAG sequence for read access of memory-mapped resources

Table 73-51. Accessing memory-mapped resources (reads)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Address Register (RWA)
2	37	Write RWA (initialize starting read address — data input on TDI)
3	13	Nexus Command = write to Read/Write Control/Status Register (RWCS)
4	37	Write RWCS (initialize read access mode and CNT value — data input on TDI)
5	—	Wait for falling edge of **nex_rdy_b** or **nex_err_b** pin
6	13	Nexus Command = read Read/Write Access Data Register (RWD)
7	37	Read RWD (data output on TDO)
8	—	If CNT > 0, go back to Step #5

73.23.3 JTAG sequence for write access of memory-mapped resources

Table 73-52. Accessing memory-mapped resources (writes)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Control/Status Register (RWCS)
2	37	Write RWCS (initialize write access mode and CNT value - data input on TDI)
3	13	Nexus Command = write to Read/Write Address Register (RWA)
4	37	Write RWA (initialize starting write address - data input on TDI)
5	13	Nexus Command = read Read/Write Access Data Register (RWD)
6	37	Write RWD (data output on TDO)
7	—	Wait for falling edge of **nex_rdy_b** or **nex_err_b** pin
8	—	If CNT > 0, go back to Step #5

Chapter 74
Nexus Crossbar Multi-Master Client (NXMC)

74.1 NXMC chip-specific module information

74.1.1 Generating In Circuit Trace Messages (ICTM)

To generate In Circuit Trace Messages (ICTM) on this chip, follow the steps below.

1. Configure the Nexus trace memory path in NAR:

   - Trace memory port base addresses-NAR_TBALO, NAR_TBAHI
   - Trace memory bus Control register-NAR_TCR (Transfer mode, Block Size, Threshold, Wrap around and Max transfer size)
   - Trace memory fill level and partition-NAR_AHFPAR
   - Nexus Aurora PHY Control (if available and used)-NAR_NAPCR
   - NAR Control Register-NAR_CR (to start NAR module with other attributes)

2. Configure the NXMC[x] to enable trace from peripheral module:

   - Development control register 1 - DC1 (Example: 0x0000_0002 ; Data trace enabled)
   - Development control register 2 - DC2 (Example: 0x0000_FFFF ; Enable peripheral interface)

3. Configure the SPU to enable trigger event to NXMC[x]:

   - Configure sequence-Sequence Control (SCTRL) (start, end and enable; example 0x0000_0011)
   - Configure connects between states:
     - State0 Goto Control (S0GC) (Enable Go to control (TE, NF), Next State on True Condition (Go to State 1), Next State on False Condition (Go to State 0))
     - State1 Goto Control (S1GC) (Enable Go to control (NF), Next State on False Condition (Go to State 1))

4. Configure actions for timestamp message into trace stream :

- State0 True Action 0 (S0TA0) (Action1: group = system, client = 0 , action = Timestamp0_NXMC0)
- State0 True Action 0 (S0TA0) (Action2: group = counter, client = 1 , action = Reset Counter)
- State0 True Action 1 (S0TA1) (Action3: group = counter, client = 2 , action = Reset Counter)
- State1 True Action 0 (S1TA0) (Action1: group = system, client = 0 , action = Timestamp0_NXMC1)
- State1 True Action 0 (S1TA0) (Action2: group = counter, client = 1 , action = Capture)
- State1 True Action 1 (S1TA1) (Action3: group = counter, client = 2 , action = Capture)

Configure Input triggers:

- Level1 Mux Selection 0 (L1SEL0)
- Level1 Mux Selection 1 (L1SEL1)

Map inputs to state inputs:

- Level2 Mux state n selection 0 (L2nSEL0)

Configure counters:

- Counter Control 0 (CCTRL0) (enable, increment)
- Counter Control 1 (CCTRL1) (enable, increment)

74.2 Introduction

The Nexus Crossbar Multi-master Client (NXMC) module provides real-time trace capabilities in compliance with the IEEE-ISTO Nexus 5001-2012 standard. This module provides development support capabilities for devices without requiring address and data pins for internal visibility. A portion of the pin interface is also compliant with the IEEE 1149.1 JTAG standard. The IEEE-ISTO 5001 standard defines an extensible auxiliary port which is used in conjunction with the JTAG port.

Many bus masters start accesses to internal address locations via the system bus. The NXMC module monitors the system bus and provides real-time trace information to debug or development tools. This also provides the capability of sending user defined messages for the peripherals attached to it.

74.2.1 Features

The NXMC module is compliant with the IEEE-ISTO 5001-2012 standard. The following features are implemented:

- Data trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to (selected) internal memory resources.
- Multi-master tracing capability for multiple master accesses
- Watchpoint messaging based on internal/external triggers, via the auxiliary pins
- Watchpoint trigger events out based on internal/external triggers
- Watchpoint trigger enable of data trace messaging

- Peripheral interface for data message transfers based on In Circuit Trace Message (ICTM) format
- Start or stop of Tracing (Data Trace, Watchpoint Trace) based on external triggers.
- Optional timestamping of the messages based on timer value and controls received at the interface
- Nexus Auxiliary port for higher data throughput
  - One $\overline{\text{EVTO}}$ (Watchpoint Event) pin
  - One $\overline{\text{EVTI}}$ (Event In) pin
- Registers for data trace, watchpoint generation, and watchpoint trigger
- All features controllable and configurable via the JTAG port

74.3 External signal description

This section details information regarding the standard ports and protocol. The NXMC pin interface provides the function of transmitting messages from the messages queues to NAR. It is also responsible for handshaking with the message queues.

The NXMC module implements the following signals:

- One $\overline{\text{EVTI}}$
- One $\overline{\text{EVTO}}$

These pins are described in Table 74-2.

All NXMC input functionality is controlled through the JTAG port in compliance with IEEE 1149.1 (see IEEE 1149.1 (JTAG) Test Access Port for details). A separate JTAG TAP controller is instantiated within the NXMC. The following table describes the JTAG pins.

Table 74-1. JTAG pins for NXMC

JTAG port	Input/ output	Description
TDO	O	The Test Data Output (TDO) pin is the serial output for test instructions and data. TDO is three-stateable and is actively driven in the shift-IR and shift-DR controller states. TDO changes on the falling edge of TCK.
TDI	I	The Test Data Input (TDI) pin receives serial test instruction and data. TDI is sampled on the rising edge of TCK and has an internal pull-up resistor.
TMS	I	The Test Mode Select (TMS) input pin is used to sequence the JTAG test controllers' state machine. TMS is sampled on the rising edge of TCK and has an internal pull-up resistor.
TCK	I	The Test Clock (TCK) input pin is used to synchronize the test logic, and control register access through the JTAG port.
TRST	I	The Test Reset ($\overline{\text{TRST}}$) input pin is used to asynchronously initialize the JTAG controller. The $\overline{\text{TRST}}$ pin has an internal pull-down resistor.

Table 74-2. NXMC auxiliary pins

Auxiliary port	Input/ output	Description
EVTO	O	Event Out ($\overline{\text{EVTO}}$) is an output which, when asserted, indicates one of two events has occurred based on the EOC bits in the DC Register: • one of the internal or external watchpoints has occurred and EOC = 2'b00 • system-level debug mode was entered (ipg_debug) and EOC = 2'b01 $\overline{\text{EVTO}}$ is held asserted for one cycle.
EVTI	I	Event In ($\overline{\text{EVTI}}$) is an input which initiates synchronization messages. If the Nexus module is enabled at reset, (see Enabling NXMC operation), assertion initiates data trace synchronization messages (provided data trace is enabled).

74.4 Supported messages and TCODEs

The NXMC pins allow for flexible transfer operations via public messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2012 standard defines a set of public messages. The NXMC block supports the public TCODEs listed in the following table. The SRC and MASTER field values are described in the device-specific chapter that describes how the modules are configured.

Table 74-3. Supported public TCODEs

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
Data Trace - Data Write Message	6	6	TCODE	fixed	TCODE number = 58
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 74-4)
	1	32	U-ADDR	variable	Unique portion of the data write address
	8	64	DATA	fixed	Data write value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)

Table continues on the next page...

Table 74-3. Supported public TCODEs (continued)

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
Data Trace - Data Read Message	6	6	TCODE	fixed	TCODE number = 59
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 74-4)
	1	32	U-ADDR	variable	Unique portion of the data read address
	8	64	DATA	fixed	Data read value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	ETYPE	fixed	Error type (See Table 74-5)
	14	14	ECODE	fixed	Error code (See Table 74-6)
	1	26	TSTAMP	variable	Timestamp (Optional)
Data Trace - Data Write Message w/ Sync	6	6	TCODE	fixed	TCODE number = 60
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 74-4)
	1	32	F-ADDR	variable	Full access address (leading zero (0) truncated)
	8	64	DATA	fixed	Data write value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)
Data Trace - Data Read Message w/ Sync	6	6	TCODE	fixed	TCODE number = 61
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 74-4)
	1	32	F-ADDR	variable	Full access address (leading zero (0) truncated)
	8	64	DATA	fixed	Data read value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)

Table continues on the next page...

Table 74-3. Supported public TCODEs (continued)

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	1	8	WPHIT	variable	Watchpoint source(s) number (See Table 74-22)
	1	26	TSTAMP	variable	Timestamp (Optional)
Peripheral Message (In circuit Trace Message: ICTM	6	6	TCODE	fixed	TCODE number = 34
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	6	6	CKSRC	fixed	Peripheral source identifier
	1	44	CKDATA	variable	Peripheral data
	1	26	TSTAMP	variable	Timestamp (Optional)

Table 74-4. Data trace size (DSZ) encodings (TCODE = 58, 59, 60, 61)

DSZ	Transfer size
001	8-bit
010	16-bit
011	32-bit
100	64-bit
000, 101-111	Reserved

Table 74-5. Error Type (ETYPE) encodings (applicable to all error messages)

ETYPE	Description
0000	Queue overrun caused message (one or more) to be lost
0001	Contention with higher priority message caused message(s) to be lost
0010	Reserved
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus register unimplemented)
0110–1111	Reserved

Table 74-6. Error Code (ECODE) encodings (applicable based on error source)

ECODE		Description[1]
13 to 8	7 to 0	
Source (SRC)		

Table continues on the next page...

Table 74-6. Error Code (ECODE) encodings (applicable based on error source) (continued)

ECODE		Description[1]
13 to 8	7 to 0	
000000	XXXXXXX1	Watchpoint Message(s) lost
000000	XXXXXX1X	Data Trace Message(s) lost
000000	XXXXX1XX	Reserved
000000	XXXX1XXX	Reserved
000000	XXX1XXXX	Reserved
000000	XX1XXXXX	Reserved
000000	X1XXXXXX	Reserved
000000	1XXXXXXX	Reserved
Source (SRC) n = peripheral index		
CKSRCn	XXXXXXX1	Reserved
CKSRCn	XXXXXX1X	Reserved
CKSRCn	XXXXX1XX	Reserved
CKSRCn	XXXX1XXX	Reserved
CKSRCn	XXX1XXXX	Reserved
CKSRCn	XX1XXXXX	Reserved
CKSRCn	X1XXXXXX	In-Circuit Trace Message(s) Lost
CKSRCn	1XXXXXXX	Reserved

1. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

74.5 Register description

This section describes the NXMC register definition. Nexus registers are accessed using the JTAG port in compliance with IEEE 1149.1. See IEEE 1149.1 (JTAG) Test Access Port for details. The complete list of registers is shown in this table.

Table 74-7. NXMC registers

Nexus register	Nexus access opcode (hex)	Read/ write	Read address (hex)	Write address (hex)
Development Control 1 (DC1)	2	R/W	04	05
Development Control 2 (DC2)	3	R/W	06	07
Watchpoint Trigger (WT)	B	R/W	16	17
Data Trace Control (DTC)	D	R/W	1A	1B
Data Trace Start Address1 (DTSA1)	E	R/W	1C	1D
Data Trace Start Address2 (DTSA2)	F	R/W	1E	1F
Data Trace End Address1 (DTEA1)	12	R/W	24	25
Data Trace End Address2 (DTEA2)	13	R/W	26	27

Table continues on the next page...

Table 74-7. NXMC registers (continued)

Nexus register	Nexus access opcode (hex)	Read/ write	Read address (hex)	Write address (hex)
Breakpoint/Watchpoint Control Register1 (BWC1)	16	R/W	2C	2D
Breakpoint/Watchpoint Control Register2 (BWC2)	17	R/W	2E	2F
Breakpoint/Watchpoint Address Register1 (BWA1)	1E	R/W	3C	3D
Breakpoint/Watchpoint Address Register2 (BWA2)	1F	R/W	3E	3F

74.5.1 Development control register 1 (DC1)

The Development Control Registers control the basic development features of the NXMC module. This table shows the format of the DC1.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	EOC		0	0	WEN	0	0	0	0	0	0	0	0	0	C	0	0	0	0	0	0	0	0	0	EIC		TM		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DC1 is described in this table.

Table 74-8. DC1 register field description

Field	Description	
31:29	Reserved for future functionality (read as 0)	
28:27 EOC	$\overline{\text{EVTO}}$ Control.	
	00	$\overline{\text{EVTO}}$ upon occurrence of watchpoint (internal or external)
	01	$\overline{\text{EVTO}}$ upon entry into system-level Debug Mode
	1x	Reserved
26:25	Reserved for future functionality (read as 0)	
24 WEN	Watchpoint Trace Enable.	
	0	Watchpoint Messaging disabled
	1	Watchpoint Messaging enabled
23:5	Reserved for future functionality (read as 0)	
4:3 EIC	$\overline{\text{EVTI}}$ Control.	
	00	$\overline{\text{EVTI}}$ for synchronization (Data Trace)
	01	Reserved
	10	$\overline{\text{EVTI}}$ disabled
	11	Reserved

Table continues on the next page...

Table 74-8. DC1 register field description (continued)

Field	Description	
2:0	Trace Mode.	
TM	000	No Trace
	1xx	Reserved
	x1x	Data Trace enabled
	xx1	Reserved

74.5.2 Development control register 2 (DC2)

The Development Control Registers control the basic development features of the NXMC module. This table shows the format of the DC2.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EWC								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PME0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DC2 is described in this table.

Table 74-9. DC2 register field description

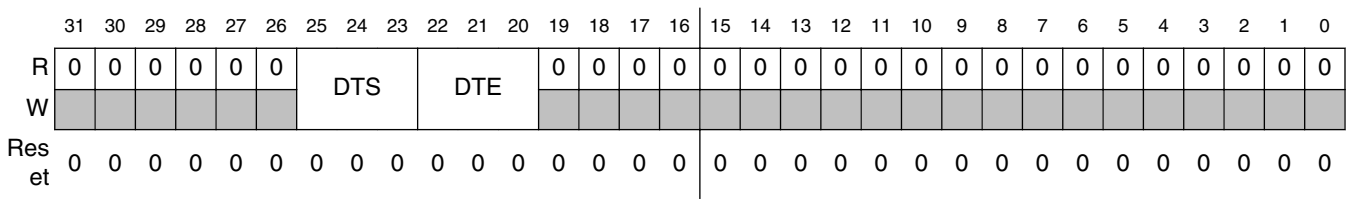
Field	Description	
31:24	$\overline{\text{EVTO}}$ Watchpoint Configuration[1].	
EWC	00000000	No Watchpoints trigger $\overline{\text{EVTO}}$
	1XXXXXXX	External Watchpoint #1 triggers $\overline{\text{EVTO}}$
	X1XXXXXX	External Watchpoint #2 triggers $\overline{\text{EVTO}}$
	XX1XXXXX	Reserved
	XXX1XXXX	Reserved
	XXXX1XXX	Internal Watchpoint #1 triggers $\overline{\text{EVTO}}$
	XXXXX1XX	Internal Watchpoint #2 triggers $\overline{\text{EVTO}}$
	XXXXXX1X	Reserved
	XXXXXXX1	Reserved
23:1	Reserved for future functionality (read as 0)	
0	Peripheral Messaging Enable for the In-Circuit Trace (ICT) feature. [2]	
PME0	0	Peripheral Interface disabled
	1	Peripheral Interface enabled

1. The EOC bits in DC1 must be programmed to trigger EVTO on watchpoint occurrence for the EWC bits to have any effect.
2. The peripheral details are described in the device-specific chapter that describes how the modules are configured.

74.5.3 Watchpoint trigger register (WT)

The Watchpoint Trigger Register allows the watchpoints defined either internally to the NXMC module or by an external module to trigger actions. These watchpoints can control data trace enable and disable.

The WT bits can be used to produce an address related window for triggering trace messages. This table shows the format of the WT register.



The WT register is described in this table.

Table 74-10. WT register field description

Field	Description[1]	
31:26	Reserved for future functionality (read as 0)	
25:23 DTS	Data Trace Start Control.	
	000	Trigger disabled
	001	Use External Watchpoint #1
	010	Use External Watchpoint #2
	011	Reserved
	100	Reserved
	101	Use Internal Watchpoint #1 (BWA1 Register)
	110	Use Internal Watchpoint #2 (BWA2 Register)
	111	Reserved
22:20 DTE	Data Trace End Control.	
	000	Trigger disabled
	001	Use External Watchpoint #1
	010	Use External Watchpoint #2
	011	Reserved
	100	Reserved
	101	Use Internal Watchpoint #1 (BWA1 Register)
	110	Use Internal Watchpoint #2 (BWA2 Register)
	111	Reserved
19:0	Reserved for future functionality (read as 0)	

1. The WT bits only enable data trace if the TM bits in the Development Control Register (DC) have not already been set to enable data trace.

74.5.4 Data trace control register (DTC)

The Data Trace Control Register controls whether DTM messages are restricted to reads, writes or both for a user programmable address range. There are two data trace channels controlled by the DTC for the NXMC module. This table shows the format of the DTC register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RWT1		RWT2		0	0	0	MME	0	0			MID				0	0	0	0	0	0	0	0	RC1	RC2	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DTC register is described in this table.

Table 74-11. DTC field description

Field	Description	
31:30 RWT1	Read/Write Trace 1.	
	00	No trace messages generated
	x1	Enable data read trace
	1x	Enable data write trace
29:28 RWT2	Read/Write Trace 2.	
	00	No trace messages generated
	x1	Enable data read trace
	1x	Enable data write trace
27:25	Reserved for future functionality	
24 MME	Multi Master Tracing enable.	
	0	Tracing enabled only for Master with ID = MID (DTC[19:16])
	1	Tracing enabled for all the Masters
23:20	Reserved for future functionality	
19:16 MID	ID of the system bus Master to be traced[1]	
15:8	Reserved for future functionality	
7 RC1	Range Control 1.	
	0	Condition trace on address within range (endpoints inclusive)
	1	Condition trace on address outside of range (endpoints exclusive)

Table continues on the next page...

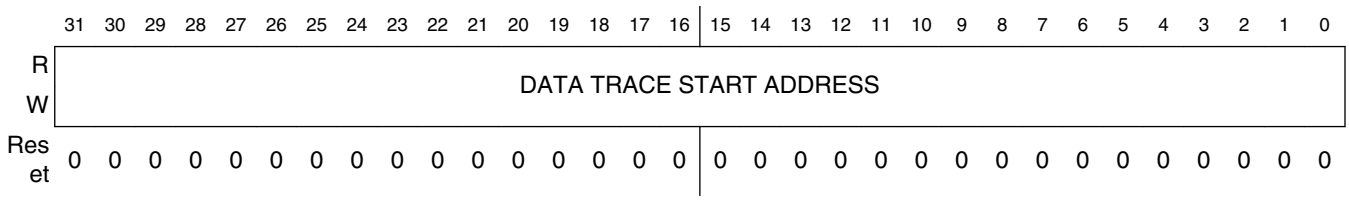
Table 74-11. DTC field description (continued)

Field	Description	
6 RC2	Range Control 2.	
	0	Condition trace on address within range (endpoints inclusive)
	1	Condition trace on address outside of range (endpoints exclusive)
5:0	Reserved for future functionality	

1. The MASTER field values are described in the device-specific chapter that describes how the modules are configured.

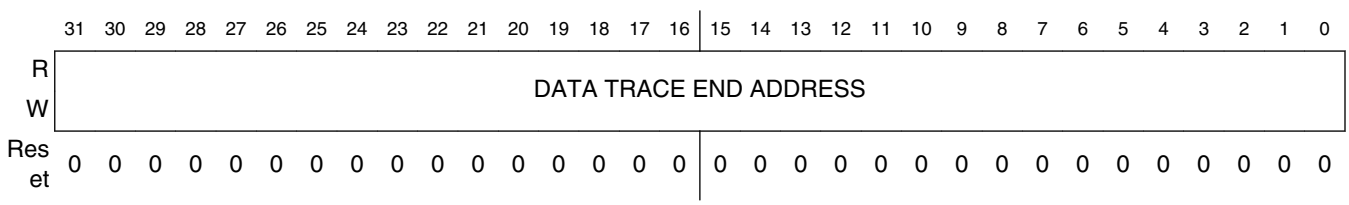
74.5.5 Data trace start address registers 1 and 2 (DTSA1 and DTSA2)

The Data Trace Start Address Registers define the start addresses for each trace channel. This table shows the format of the DTSA1 and DTSA2 registers.



74.5.6 Data trace end address registers 1 and 2 (DTEA1 and DTEA2)

The Data Trace End Address Registers define the end addresses for each Trace channel. This table shows the format of the DTEA1 and DTEA2 registers.



This table illustrates the range that are selected for data trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 74-12. Data trace address range options

Programmed values	Range control bit value	Range selected
DTSA ≤ DTEA	0	DTSA -> <- DTEA

Table continues on the next page...

Table 74-12. Data trace address range options (continued)

Programmed values	Range control bit value	Range selected
DTSA ≤ DTEA	1	<- DTSA DTEA ->
DTSA > DTEA	N/A	Invalid range—no trace

Note

DTSA must be less than (or equal to) DTEA in order to guarantee correct data write/read traces.

When the range control bit is 0 (internal range), accesses to DTSA and DTEA addresses are traced. When the range control bit is 1 (external range), accesses to DTSA and DTEA are not traced.

74.5.7 Breakpoint/watchpoint control register 1 (BWC1)

Breakpoint/watchpoint control register 1 controls the attributes for generation of NXMC watchpoint#1. This table shows the format of the BWC1 register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BWE1		BRW1		0	0	0	0	0	0	0	0	0	0	BWR1		BWT1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The BWC1 register is described in this table.

Table 74-13. BWC1 field description

Field	Description	
31:30	Breakpoint/Watchpoint #1 Enable.	
BWE1	00	Internal Nexus Watchpoint #1 disabled
	01	Reserved
	10	Reserved
	11	Internal Nexus Watchpoint #1 enabled
29:28	Breakpoint/Watchpoint #1 Read/Write Select.	
BRW1	00	Watchpoint #1 hit on read accesses
	01	Watchpoint #1 hit on write accesses
	10	Watchpoint #1 on read or write accesses
	11	Reserved

Table continues on the next page...

Table 74-13. BWC1 field description (continued)

Field	Description	
27:18	Reserved for future functionality (read as 0)	
17:16	Breakpoint/Watchpoint #1 Register Compare.	
BWR1	00	No Register Compare (same as BWC1[31:30] = 2'b00)
	01	Reserved
	10	Compare with BWA1 value
	11	Reserved
15	Breakpoint/Watchpoint #1 Type.	
BWT1	0	Reserved
	1	Watchpoint #1 on data accesses
14:0	Reserved for future functionality (read as 0)	

74.5.8 Breakpoint/watchpoint control register 2 (BWC2)

Breakpoint/Watchpoint Control Register2 controls attributes for generation of NXMC watchpoint#2. This table shows the format of the BWC2 register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BWE2		BRW2		0	0	0	0	0	0	0	0	0	0	BWR2		BWT2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The BWC2 register is described in this table.

Table 74-14. BWC2 field description

Field	Description	
31:30	Breakpoint/Watchpoint #2 Enable.	
BWE2	00	Internal Nexus Watchpoint #2 disabled
	01	Reserved
	10	Reserved
	11	Internal Nexus Watchpoint #2 enabled
29:28	Breakpoint/Watchpoint #2 Read/Write Select.	
BRW2	00	Watchpoint #2 hit on read accesses
	01	Watchpoint #2 hit on write accesses
	10	Watchpoint #2 on read or write accesses
	11	Reserved
27:18	Reserved for future functionality (read as 0)	

Table continues on the next page...

Table 74-14. BWC2 field description (continued)

Field	Description	
17:16 BWR2	Breakpoint/Watchpoint #2 Register Compare.	
	00	No Register Compare (same as BWC2[31:30] = 2b00)
	01	Reserved
	10	Compare with BWA2 value
	11	Reserved
15 BWT2	Breakpoint/Watchpoint #2 Type.	
	0	Reserved
	1	Watchpoint #2 on data accesses
14:0	Reserved for future functionality (read as 0)	

74.5.9 Breakpoint/watchpoint address registers 1 and 2 (BWA1 and BWA2)

The Breakpoint/Watchpoint Address Registers are compared with system bus addresses in order to generate internal watchpoints. This table shows the format of the BWA1 and BWA2 registers.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BREAKPOINT / WATCHPOINT ADDRESS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

74.5.10 Unimplemented registers

Unimplemented registers are those with client select and index value combinations other than those listed in Table 74-7. For unimplemented registers, the NXMC module drives TDO to zero during the SHIFT-DR state. It also transmits an error message with the invalid access opcode encoding (ETYPE = 0101).

74.6 Programming considerations (RESET)

If NXMC register configuration is to occur during system reset (as opposed to debug mode), all NXMC configuration should be completed between the negation of $\overline{\text{TRST}}$ and system reset de-assertion. The JTAG ID register should be read by the tool after negation of $\overline{\text{TRST}}$ and should be programmed before deassertion of system reset.

74.6.1 IEEE 1149.1 (JTAG) Test Access Port

The NXMC module uses the IEEE 1149.1 TAP controller for accessing Nexus resources. The JTAG signals themselves are shared by all TAP controllers on the device. The NXMC module implements a 4-bit Instruction Register (IR). The valid instructions and method for register access are outlined in NXMC register access via JTAG.

74.6.2 Enabling NXMC operation

The Nexus module is enabled by loading a single instruction into the JTAG Instruction Register (IR). Once enabled, the module is ready to accept control input via the JTAG pins.

The Nexus module is disabled when the JTAG state machine reaches the Test-Logic-Reset state. This state can be reached by the assertion of the $\overline{\text{TRST}}$ pin or by cycling through the state machine using the TMS pin. The Nexus module is also disabled if a Power-on-Reset (POR) event occurs.

If the NXMC module is disabled, no trace output is provided. Nexus registers are not available for reads or writes.

74.6.3 Enabling NXMC TAP controller

Assertion of a Power-on-Reset signal or assertion of the $\overline{\text{TRST}}$ pin resets all TAP controllers. Upon exit from the Test-Logic-Reset state, the IR value is loaded with the JTAG ID. When the NXMC TAP is accessed, this information helps the development tool obtain information about the Nexus module it is accessing, such as version, sequence, feature set, etc.

74.6.4 NXMC register access via JTAG

Access to Nexus register resources is enabled by loading a single instruction into the JTAG Instruction Register (IR).

Once the JTAG NEXUS-ACCESS instruction has been loaded, the JTAG port allows tool/target communications with all Nexus registers according to the map in Table 74-7.

Reading/writing of a Nexus register then requires two passes through the Data-Scan (DR) path of the JTAG state machine.

1. The first pass through the DR selects the Nexus register to be accessed by providing an index (see Table 74-7), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the format shown below.

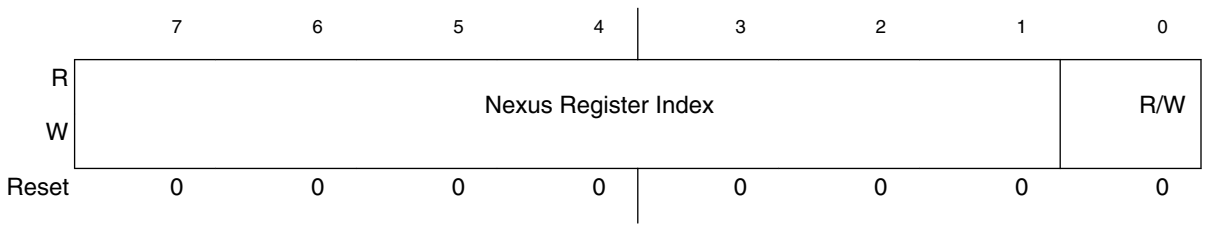


Table 74-15. JTAG DR field description for Nexus register access

Field	Description	
Nexus Register Index	Selected from values in Table 74-7	
Read/Write (R/W)	0	Read
	1	Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
   a. During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the Capture-DR state.
   b. During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the Update-DR state.

74.7 Functional description

The following sections describe the functionality of the NXMC module.

74.7.1 Data trace

This section describes the data trace mechanism supported by the NXMC module. Data trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the IEEE-ISTO 5001-2012 standard.

74.7.1.1 Data trace messaging (DTM)

NXMC data trace messaging is accomplished by snooping the system bus and storing the information for qualifying accesses (based on enabled features and matching target addresses). The NXMC module traces all data access that meet the selected range and attributes.

74.7.1.2 DTM message formats

The NXMC block supports five types of DTM messages:

- Data write
- Data read
- Data write synchronization
- Data read synchronization
- Error messages

74.7.1.2.1 Data write and data read messages

The data write and data read messages contain the data write/read value and the address of the write/read access, relative to the previous data trace message. The DATA field is considered fixed based on the DSZ encodings to either 8, 16, 32 , 64. The TSTAMP field is 26 bits. Data write message and data read message information is messaged out in the format shown in the following figure. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

Table 74-16. DTM message format for DATA width = 64 bits

8/16/32/64 bits	1–32 bits	3 bits	4 bits	4 bits	6 bits
DATA	U-ADDR	DSZ	MASTER	SRC	TCODE (111010 or 111011)

TSTAMP is another optional field which is inserted based on the controls received from the NAR, see Timestamping feature.

74.7.1.2.2 Data trace with non-contiguous byte strobes asserted

The v6 extensions to AMBA 2.0 allow system bus transfers with bytes non-asserted byte strobes in between asserted byte strobes. The NXMC does not support transfers with non-contiguous byte strobes asserted. In such a condition, the invalid bytes in the DATA packet are driven to the value of 0xFF.

74.7.1.2.3 DTM overflow error messages

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards incoming messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied.

If only a data trace message attempts to enter the queue while it is being emptied, the error message incorporates the data trace only error encoding (ECODE = 00000000000010). If a watchpoint also attempts to be queued while the FIFO is being emptied, then the error message incorporates error encoding (ECODE = 00000000000011).

Error information is messaged out in the format shown in this figure. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

Table 74-17. DTM error message format (fixed length = 28 bits)

14 bits	4 bits	4 bits	6 bits
ECODE	ETYPE	SRC	TCODE (001000)

74.7.1.2.4 Data trace synchronization messages

A data trace write/read with synchronization message is messaged via the auxiliary port (provided data trace is enabled) for the following conditions (see Table 74-19):

- Initial data trace message upon exit from system reset or whenever data trace is enabled is a synchronization message.
- Upon returning from a low power state, the first data trace message is a synchronization message.

- Upon returning from debug mode, the first data trace message is a synchronization message.
- After occurrence of queue overrun (can be caused by any trace message), the first data trace message is a synchronization message.
- After the periodic data trace counter has expired indicating 255 without-sync data trace messages have occurred since the last with-sync message occurred.
- Upon assertion of the event in (EVTI) pin, the first data trace message is a synchronization message if the EIC bits of the DC Register have enabled this feature.
- Upon data trace write/read after the previous DTM Message was lost due to a collision while entering the FIFO between the DTM message and any of the following: error message, or watchpoint message or peripheral message.This is a very rare occurrence and possible only in case of continuous burst on the system bus as well as watchpoint events which is unrealistic.

Data trace synchronization messages provide the full address (without leading zeros) and ensure that development tools fully synchronize with data trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the data trace address is transmitted. The format for data trace write/read with synchronization messages is shown in the following figure. The SRC and MASTER field values are described in the device-specific chapter that describes how the modules are configured.

Table 74-18. Data write/read with synchronization message format

8-64 bits	1–32 bits	3 bits	4 bits	4 bits	6 bits
DATA	F-ADDR	DSZ	MASTER	SRC	TCODE (111100 or 111101)

Exception conditions that result in data trace synchronization are summarized in this table.

Table 74-19. Data trace exception summary

Exception condition	Exception handling
System Reset Negation	At the negation of JTAG reset ($\overline{\text{TRST}}$), queue pointers, counters, state machines, and registers within the NXMC module are reset. If data trace is enabled, the first data trace message is a data write/read with synchronization message.
Data Trace Enabled	The first data trace message (after data trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a low power mode or debug mode the next data trace message is converted to a data write/read with synchronization message.

Table continues on the next page...

Table 74-19. Data trace exception summary (continued)

Exception condition	Exception handling
Queue Overrun	An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue is a data write/read with synchronization message.
Periodic Data Trace Synchronization	A forced synchronization occurs periodically after 255 data trace messages have been queued. A data write/read with synchronization message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, an $\overline{\text{EVTI}}$ assertion initiates a data write/read with synchronization message upon the next data write/read (if data trace is enabled and the EIC bits of the DC Register have enabled this feature).
Collision Priority	All messages have the following priority: Error > WPM > ICTM > DTM. Although a very rare occurrence, during simultaneous burst lower priority messages can be lost due to burst of high priority messages. Proper error messages are generated and in case of DTM lost a subsequent read/write queues a data write/read with synchronization message.

74.7.1.3 Enabling data trace messaging operation

Data trace messaging can be enabled in one of three ways:

- Setting the TM field of the DC Register via JTAG to enable data trace (DC[1]).
- Using the DTS field of the WT Register to enable data trace on watchpoint hits (either watchpoints configured within the NXMC module or external watchpoint inputs controlled by the device).
- By toggling the external start control, which in turn sets the TM field of the DC register to enable data trace. (The TM bit clears and hence the data tracing stops on the toggle of external stop control. In case of conflict between JTAG write and external start/stop control, JTAG write takes precedence.

Note

When enabling the DTM, either via JTAG or external start/stop control, it is important to make sure that all the other controls (i.e. DTC, DTSA/E1/2 and WT register fields) are correctly configured to ensure correct DTM functionality.

74.7.1.4 DTM queueing

The NXMC implements a programmable depth queue for queuing all messages. Messages that enter the queue are transmitted via the auxiliary pins in the order in which they are queued.

Note

If multiple trace messages need to be queued at the same time, watchpoint messages have the highest priority (WPM > ICTM > DTM).

74.7.1.5 Relative addressing

The relative address feature is compliant with IEEE-ISTO 5001-2012 and is designed to reduce the number of bits transmitted for addresses of data trace messages. The address transmitted is relative to the address of the previous data trace message. It is generated by XORing the new address with the previous address, and then using only the results up to the most significant 1 in the result. To recreate this address, an XOR of the (most-significant 0-padded) message address with the previously decoded address gives the current address.

For example, if the previous address (A1) = 0x0003FC01, the new address (A2) = 0x0003F365 as shown in this figure.

Message Generation:

A1 = 0000 0000 0000 0011 1111 1100 0000 0001
A2 = 0000 0000 0000 0011 1111 0011 0110 0101

A1 ⊕ A2 = 0000 0000 0000 0000 0000 1111 0110 0100

Address Message (M1) = 1111 0110 0100

Address Re-creation:

A1 ⊕ M1 = A2
A1 = 0000 0000 0000 0011 1111 1100 0000 0001
M1 = 0000 0000 0000 0000 0000 1111 0110 0100

A2 = 0000 0000 0000 0011 1111 0011 0110 0101

Figure 74-1. Relative addressing example

74.7.1.6 Data trace windowing

Data write/read messages are enabled via the RWT1(2) field in the Data Trace Control Register (DTC) for each DTM channel. Data trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC1(2) field in the DTC. All system bus DMA initiated read/write accesses which fall inside or outside these address ranges, as programmed, are candidates to be traced.

74.7.1.7 System bus cycle special cases

The system bus cycle special cases are shown in this table.

Table 74-20. System Bus cycle special cases

Special case	Action
System bus cycle with data error	Data trace message discarded
System bus cycle completed without error	Cycle captured and transmitted
System bus cycle is an instruction fetch	Cycle ignored

74.7.1.8 System bus multi-master tracing

DTM tracing can be done either for all the masters or for a single master (indicated by the MID field of the DTC register) based on the DTC register MME field. In case of single master, the correct value needs to be configured in the MID register before enabling the data trace operation.

The MASTER field of the DTM message (see figure in Data write and data read messages for an example) indicates the system bus master for which the trace has been generated. For each masters (0 to 15) there is a parameter associated (MSID0-15) for indicating its actual ID, which finally gets inserted in the MASTER field. The MASTER field values are described in the device-specific chapter that describes how the modules are configured.

74.7.2 Watchpoint support

The NXMC module provides watchpoint messaging via the auxiliary pins, as defined by IEEE-ISTO 5001-2012. Watchpoint messages can be generated by either using the NXMC defined internal watchpoints or by externally defined watchpoint signals.

74.7.2.1 Watchpoint messaging

Enabling watchpoint messaging is accomplished by setting the WEN bit in the DC Register. This bit is set either via JTAG write 1 access or external start indication, and cleared via JTAG write 0 access or external start indication. In case of conflict between JTAG write and external start/stop control, JTAG write takes precedence.

Note

In case of enabling the watchpoint messaging, either via JTAG or external start/stop control, it is important to make sure that all the other Watch Point Message (WPM) controls (i.e. BWC1/2, BWA1/2 register fields) are configured with the correct values in order to ensure correct WPM operation.

Watchpoint setting is supported through two methods:

- Internal watchpoints—Using the BWC1(2) registers, two independently controlled internal watchpoints can be initialized. When a system bus access address matches on BWA1(2), a watchpoint message is transmitted. The system bus multi-master or single master controls applies here too as explained for DTM (in System bus multi-master tracing).
- External watchpoints—The NXMC module supports two external watchpoint inputs. The optional logic to generate these watchpoint signals is implemented at the device level. The watchpoints are described in the device-specific chapter that describes how the modules are configured. If either of these signals asserts (i.e, on 0-to-1 transition), a watchpoint message is transmitted.

The Nexus module provides watchpoint messaging using the IEEE-ISTO 5001 defined TCODE. When any of the four possible watchpoint sources asserts, a message is sent to the queue to be messaged out. This message indicates the watchpoint number as shown in Table 74-21 and Table 74-22.

Table 74-21. Watchpoint message format (fixed length = 18 bits)

8 bits	4 bits	6 bits
WPHIT (RRRRXXXX)	SRC	TCODE (001111)

Table 74-22. Watchpoint source description

Watchpoint source (8-bits)	Watchpoint description
RRRRXXX1[1]	External Watchpoint #1(device defined)
RRRRXX1X	External Watchpoint #2 (device defined)
RRRRX1XX	Internal Watchpoint #1 (BWA1 match)

Table continues on the next page...

Table 74-22. Watchpoint source description (continued)

Watchpoint source (8-bits)	Watchpoint description
RRRR1XXX	Internal Watchpoint #2 (BWA2 match)

1. R = reserved for future use (insert 0s).

74.7.2.2 Watchpoint error message

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied.

If only a watchpoint message attempts to enter the queue while it is being emptied, the error message incorporates the watchpoint only error encoding (ECODE = 00000000000001). If a data trace message also attempts to enter the queue while it is being emptied, the error message incorporates error encoding (00000000000011). Error information is messaged out in the format shown in this figure.

Table 74-23. Error message format (fixed length = 28 bits)

14 bits	4 bits	4 bits	6 bits
ECODE	ETYPE	SRC	TCODE (001000)

74.7.3 Peripheral interface

The NXMC provides an asynchronous interface to peripherals for sending data messages. The NXMC can back pressure or delay the data transfer using a data accept signal. A round-robin scheme is followed between the peripherals for forwarding received data to the queue; therefore, each peripheral has an equal priority and hence bandwidth for data transfers.

To enable messaging from peripherals there are controls present in the DC2 register for individual peripherals: DC2[PME0]. There are no message generated and hence data accepted from a peripheral unless the corresponding enable bit is set to 1.

Table 74-24. Peripheral DATA bus fields

MSB	6 bits	(54-6) bits			LSB
CKSRCn		DATAn			
54-1	54-6	54-6-1	2	1	0

n = peripheral number

Another input port is present which indicates data loss inside the peripheral itself due to over-run, the module in turn generates error message to communicate such events to the debugger.

74.7.3.1 Peripheral data message format (ICTM)

The message formats for the peripheral data messages (formatted as ICTM) is shown in the following figure. The SRC field values are described in the device-specific chapter that describes how the modules are configured. The CKSRC field is received from the MSB 6 bits of the individual peripheral data bus, and CKDATA is a variable field and contains 44 LSBits received on the databus of the peripheral interface.

Table 74-25. Peripheral message format (max length = 60 bits, min length = 17 bits)

1–44 bits	6 bits	4 bits	6 bits
CKDATAn	CKSRCn	SRC	TCODE (100010)

74.7.3.2 Peripheral error message format

The error messages are generated based data over-run indication from a peripheral. Error information is messaged out in the format shown in this figure.

Table 74-26. Error message format (fixed length = 28 bits)

14 bits	4 bits	4 bits	6 bits
ECODE(CKSRCn,ErrCODE)	ETYPE	SRC	TCODE (001000)

The SRC field values are described in the device-specific chapter that describes how the modules are configured. ETYPE and ECODE encoding are listed in Table 74-5 and Table 74-6.

74.7.4 Timestamping feature

This feature allows a timestamp packet to be appended to all messages, based on the timer value and controls received externally from the NAR. If enabled, this packet is added as the last field to all the outgoing messages. The TSTAMP packet width is 26 bits.

The TSTAMP packet is appended at the end of a message during message formatting, before transfer. This figure shows an example of ICTM message with a TSTAMP field.

Table 74-27. Message with TSTAMP field example

26 bits	1–44 bits	6 bits	4 bits	6 bits
TSTAMP	CKDATAn	CKSRCn	SRC	TCODE (100010)

The enable or disable indication should remain static for the complete debug session. Dynamically enabling or disabling of the timestamping is not allowed during running debug tracing. The following steps allow seamless enabling/disabling of the timestamping:

1. All tracing (DTM/WPM/ICTM) must be disabled
2. Wait for the internal queues to be emptied (no further request is asserted at the auxiliary interface)
3. Change the timestamp enable indication to assert/deassert
4. Enable the required tracing controls (via TM, PME0 etc. control bits)

Chapter 75
Functional Safety

75.1 Introduction

The MPC5746R offers a set of features to support applications that need to fulfill functional safety requirements as defined by automotive safety integrity level ASIL-D of ISO26262. This chapter gives an overview of the safety implementation for the MPC5746R and discusses the operation of the safety mechanisms.

75.2 Safety overview

The MPC5746R has been designed for automotive applications executing safety-relevant tasks that require either a fail-silent or fail-indicate MCU. The MPC5746R supports a Fault Tolerant Time Interval (FTTI) of at least 10 ms. Shorter intervals are possible depending on the actual combination of failure detection and signaling mechanisms as documented in the Safety Manual (SM), but, in general, the MPC5746R supports an FTTI of 10 ms.

75.3 Module categorization

The definition of the MPC5746R safety concept and its validation (particularly the computation of Probabilistic Metric for Random Hardware Failures (PMHF)) has been done by splitting system modules into four categories as shown in the following table.

Table 75-1. PMHF module categories

Module category	Category abbreviation	Description
Vital system modules	ViMo	All modules that can directly influence the correct operation of the software execution (by influence of the correct function of CPU, RAM, flash, and the bus interconnect). Examples of ViMo are the core, the cache, the crossbar, and the interrupt controller (INTC), as well as clock or power generation.
Peripheral modules	PeMo	All microcontroller modules involved in I/O operation, specifically including the I/O bridge.
Non-safety modules	NoSaMo	Modules that belong in one of the above categories but that do not require hardware-level safety measures. Only their non-interference with modules from the other categories is ensured by hardware. In case they are used in a safety-relevant way, application measures are necessary to ensure their safe operation.
Other support modules	SuMo	These are modules which, although safety-relevant and in principle potential members of the ViMo category, cannot lead to a violation of the safety goal on their own. Typically, these will only have multiple point faults, but not single point faults. Example SuMo's are the RCCU and Clock Monitoring Unit (CMU).

75.4 System implementation

75.4.1 General concept

In general, safety integrity is achieved in the following ways:

- For the Safety Core, and its closely related periphery, safety is ensured by a delayed lockstep approach (see section "Dual-core lockstep" in this chapter).
- The safety of storage and of the data path to storage and periphery is ensured by End-to-End ECC (E2E ECC) with address encoding and selected additional measures for individual modules. For the periphery, end-to-end ECC ends at the I/O bridges (see section "End-to-End protection on data path" in this chapter).
- Clock and power, generation and distribution, are supervised by dedicated monitors (see section "Clock and power monitoring" in this chapter).
- The safety of the periphery is ensured by application-level measures (such as connecting one sensor to different I/O modules, sensor validation by sensor fusion, and so on). Hardware supports this application-level redundancy by providing redundant I/O modules connected to different peripheral bridges (PBRIDGE) to maximize the independence between the monitored and monitoring resources (see sections "I/O peripherals" and "Communication controllers" in this chapter).
- MBIST and LBIST are provided to avoid the accumulation of latent faults in the functional logic as well as the safety integrity mechanisms (see section "Built-In Self Tests (BIST)" in this chapter). Dedicated mechanisms are provided to check the

availability of safety mechanisms and the functionality of each error reaction path (such as by fake fault injection). The MPC5746R Safety Manual includes detailed information on BISTs implemented on the device.

- The Fault Collection and Control Unit (see "Fault Collection and Control Unit (FCCU)" chapter for FCCU details) is responsible for collecting and reacting to failure notifications (see section "FCCU and failure monitoring" in this chapter). MEMU (see section "Memory Error Management Unit (MEMU)" in this chapter) is responsible for the collection and reporting (to the FCCU) of error events in system memories, flash memories, and peripheral memories, as well as E2E ECC errors caused by the XBAR, or RAM or flash memory controllers. Error events include ECC corrections and detections as well as faults detected by memory BISTs.
- Common Cause Failures (CCF) are handled by measures for control and avoidance of CCFs, including system-level approaches (such as temperature and nonfunctional signal monitoring) and back-end techniques (such as isolated silicon areas and routing constraints) (see section "Common Cause Failure measures" in this chapter).

75.4.1.1 End-to-End protection on data path

Connections between XBAR masters and slaves (clients) are denoted as data paths. Data corruption on all data paths between the Safety Core and any client is detected via two main safety mechanisms: data from the replicated cores is encoded using Error Correcting Code (ECC), which is implemented with a Single-Error Correction, Double-Error Detection (SECDED) code with a Hamming distance of 4 and includes coverage of addressing information; control signals and address decoding are monitored to verify the data reaches all of the intended clients, from all possible connections to these clients and the intended operation is performed on the target address. The following figure shows a general view of ECC schema.

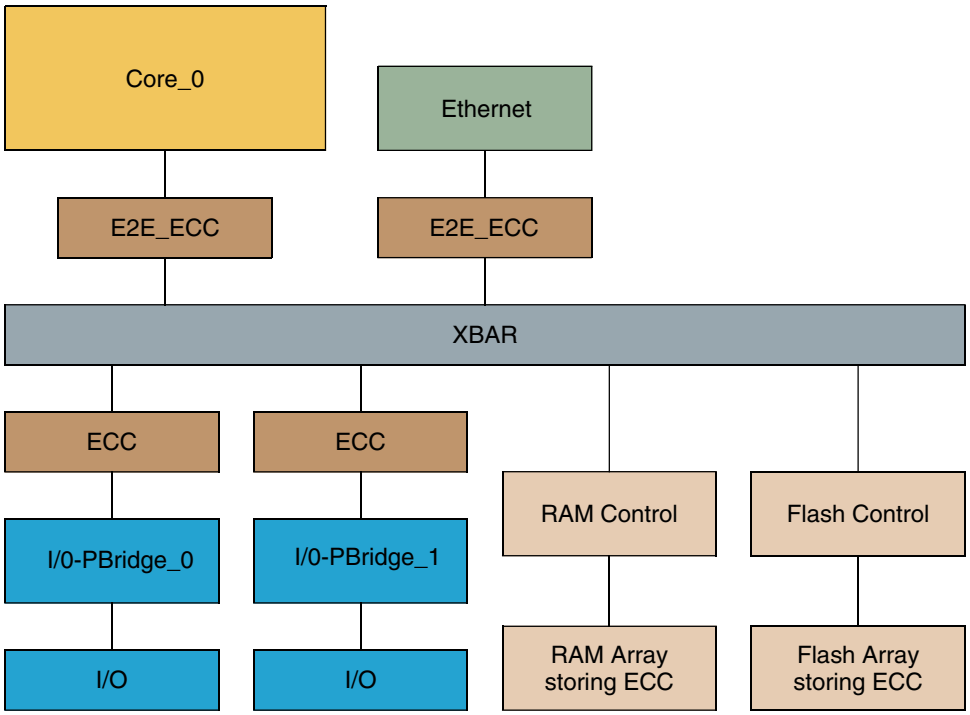


Figure 75-1. General view of E2E ECC

NOTE

Specific implementation for the MPC5746R may vary depending on the special requirements of RAM and flash memory concerning ECC handling.

ECC bits are generated on writes by XBAR masters (including, but not limited to the Safety Core) and checked on reads. The ECC correction bits are stored alongside the data in flash memory and RAM so, in principle, no ECC logic is necessary at the memories themselves. For this reason the ECC schema is referred to as End-to-End ECC (E2E ECC) in the following sections. For XBAR slaves, other than memories, new ECC logic is added as these clients cannot store or produce the ECC correction bits. This resolves the problem where ECC needs to be calculated in real time before entering or exiting the ECC-protected data path. This is particularly true with peripherals connected to the I/O bridges. This setup is considered sufficient to fulfill safety requirements because the data path not protected by ECC, which is downstream from the I/O bridges, is replicated and is used redundantly by the application (see section "I/O peripherals" in this chapter).

The E2E ECC schema provides high detection capabilities against failures affecting the data content of the transaction. The inclusion of the target address in the computation of the redundancy bits (8 ECC bits) allows the partial detection of addressing faults as well. To reach the desired integrity level, additional dedicated safety mechanisms are implemented in the data path particularly to:

- Improve the detection capability over addressing failures (no/multiple/wrong address selected), considering faults affecting address transmission (from master to client) as well as the decoding of the address;
- Provide coverage for control failures affecting, for example, the type (read vs. write) or size of a transaction.

Though safety mechanisms protecting the XBAR, the RAM controller, or the flash memory controller are different, they are all based on the feedback of address and control information from the target to the source of the transaction, which is responsible for checking for consistency with respect to the intended transaction. Depending on the portion of the data path covered by the specific safety mechanism, the source can be an XBAR master port rather than the XBAR interface of the RAM or Flash Memory Controllers; the target is respectively an XBAR slave port, the RAM array, or the flash memory module. See the separate chapters dedicated to the Crossbar Switch (XBAR), Flash Memory Controller, and RAM Controller (PRAM) for details.

75.4.1.2 Clock and power monitoring

75.4.1.2.1 Clock

Clocks in the MPC5746R are supervised by Clock Monitor Units (CMUs). The CMUs are driven by the IRCOSC (16 MHz internal oscillator) for independent operation from the monitored clocks. CMUs flag errors associated with conditions due to loss of clock, clock out of a programmable bound (over and under frequency), and loss of reference. If a supervised clock leaves the specified range for the device, a signal is sent to the FCCU.

Please see the Clock Management Unit chapter for the clocks that are supervised.

The clock tree for Core_0 and Core_0 - Checker (the two lock step cores comprising the Safety Core) consists of separate clock branches that are not shared with any other module.One CMU (CMU_1) is connected to the RCCU clock path, downstream from the common part of the clock tree for the Master and Checker Cores. In this way any frequency failure affecting the RCCU clock or the common clock (SYS_CLK) is detected by the CMU, and any frequency failure affecting the Main or Checker Core clock is detected by the Redundancy Control and Checker Units (RCCUs) (see "Clocking" chapter for detailed clocking information).

75.4.1.2.2 Power

There are two types of voltage supervisors on the MPC5746R, Low Voltage Detect (LVD) and High Voltage Detect (HVD) monitors. Safety relevant voltages (recommended operating voltages), are supervised for voltage levels that are out of correct operating voltage ranges. Since any voltage running outside of the correct operating voltage range has the potential to disable the failure indication mechanisms of the MCU, the indication of these voltage variations can be used to cause a direct transition of the MCU into the safe state (reset assertion) (see "Power Management Controller digital interface (PMC_dig)" and "Power Control Unit (MC_PCU)" chapters for details on power monitoring).

75.4.1.3 I/O peripherals

To allow a safety application to use redundant I/O peripherals, each individual peripheral from a pair of redundant peripherals is connected to a different PBRIDGE. For example, if DSPI is provided by the MCU, two DSPI modules (DSPIn, DSPIm) are included and connected externally through different pins. Internally, DSPIn is connected to PBRIDGE_0 and DSPIm is connected to PBRIDGE_1 and is accessible via different addresses.

The arrangement of I/O peripherals onto two PBRIDGEs, as well as further CCF prevention measures, allow redundant use of peripherals while limiting possible causes of CCFs. Redundant usage includes usage of equivalent peripherals in a replicated way as well as usage of functionally different peripherals in, for example, feedback measurement loops. Comparison of redundant operation is the responsibility of the application software, not the safety hardware mechanism.

75.4.1.4 Communication controllers

Communication controllers provide the ability to exchange information with external components and therefore fall under the same safety requirements as I/O peripherals. High bandwidth communication controllers require additional software measures which provide mechanisms for adhering to functional safety requirements.

The following communication controllers do not contain special safety mechanisms (above what is included in their protocol specifications) nor are they duplicated or spread over the PBRIDGE (in contrast to other I/Os, see section "I/O peripherals" in this chapter):

- FlexCAN Communication Controller

- Universal Asynchronous Communication Controller (LINFlexD)
- Fast Ethernet Controller (FEC)

Typically, software measures for the communication controllers (also called fault-tolerant communication layer) can include E2E CRC data protection, sender identification, sequence numbering, and an acknowledgement mechanism.

75.4.1.5 Built-In Self Tests (BIST)

BIST is a mechanism that permits a device to perform self-test. BIST can be used (typically at startup) to ensure the absence of latent faults. On the MPC5746R, BIST provides a mechanism for meeting this requirement for avoiding latent faults as defined by the ISO 26262 standard. Different BIST mechanisms are implemented in the MPC5746R: LBIST for digital logic and MBIST for memories. LBIST and MBIST execution is managed by the STCU2 (see "Self-Test Control Unit (STCU2)" chapter).

75.4.1.5.1 BIST during boot

A device BIST is performed every time the device boots. BIST is performed transparently for the application while the device is still under reset. In case the BIST fails, the device is held in reset, but the BIST continues running, if possible, and informs the system of the failure. Application software can start executing when the BIST finishes successfully without detecting a fault. The boot time BIST comprises:

- Memory BIST for all RAMs and ROM
- Scan-based Logic BIST for digital logic, which is divided into multiple partitions that can be configured to be tested in parallel or sequentially to find the best time versus power consumption trade off.

75.4.1.6 Redundancy Control and Checker Units (RCCUs)

The RCCU compares a set of equivalent input signals provided by different sources and issues an alarm in case of a mismatch. For example, an RCCU compares the output of the checker core against the output of the Master core and issues an alarm if they do not match. In case of a compare mismatch the fault stays stable until it is explicitly cleared.

75.4.1.7 FCCU and failure monitoring

The FCCU offers a hardware mechanism to aggregate error notifications and a configurable means to bring the device to a safe state. No CPU intervention is required for collection and control operation. Error indications are passed from the individual hardware components to the FCCU where the appropriate action is decided (according to the FCCU configuration).

For a list of fault inputs see section "FCCU failure inputs from other modules" in the FCCU chapter.

75.4.1.7.1 External error indication

Failure of the MCU is signaled to one or two pins, FCCU_F[0] and FCCU_F[1]. See the "Fault Collection and Control Unit (FCCU)" chapter for details on the fault output signals.

The error indication on pins FCCU_F[0] and FCCU_F[1] is controlled by the FCCU.

The error status flag (FCCU_STAT[ESTAT]) can be read to determine whether the FCCU is in an error state. This flag can be written by software to either a 1 (fault) or 0 (operational) when the FCCU is in operational state. Another flag, FCCU_STAT[PhysicErrorPin], is accessible through the register interface. It mirrors the physical state of the FCCU_F[n] external pin's value, though this might differ from the logical state if a toggling protocol is used.

75.4.1.7.2 Failure handling

The FCCU is an autonomous module that is responsible for reacting to failure indicators. A different reaction can be configured for each failure source. Overall failure reaction time requires time for detecting, processing, and indicating the error. During this time, the MPC5746R could provide incorrect results to the system.

Failure sources include:

- All failure indication signals from modules within the MCU
- Control logic and signals monitored by the FCCU itself. See the "Fault Collection and Control Unit (FCCU)" chapter for details.
- Software-initiated failure indications. For example, software signals the FCCU that it has evidence of a failure. Keep in mind that software can also directly influence the state of the FCCU_F[n] pins.
- External failure input

Available failure reactions are:

- Assertion of an interrupt (maskable or non-maskable)
- Resetting the chip
- Changing the state of the failure indication pins, ERRORn
- Disabling the transmission capabilities of communication controllers (Ethernet, CAN) (note: possible only in conjunction with changing the state of the failure indication pins and the SMC bit of the corresponding pad's MSCR register)
- No reaction

Software can read the failure source that caused a fault, and can do so either before, or after, a functional reset (the condition indicators are not volatile). Software can also reset the failure, but the external failure indication will stay in failure mode for a configurable minimum time. If necessary, software can also reset the MCU.

75.4.1.8 Operational interference protection

Being a multi-master system, MPC5746R provides safety mechanisms to prevent non-safety masters from interfering with the operation of the Safety Core, as well as mechanisms to handle the concurrent execution of software with different (lower) ASIL. Interference freedom is guaranteed via a hierarchical memory protection schema including:

- MPUs
- PBRIDGEs
- Register protection.

There are two Memory Protection Unit levels included in the MPC5746R. The Core Memory Protection Unit (CMPU) is a mechanism included in each core to protect address ranges against access by software developed according to lower ASIL. It will typically be used by the operating system to ensure inter-task interference protection.

The second MPU level is provided by the System MPUs (SMPU) located in each XBAR. They will prevent access of different bus masters to address ranges and will typically be used by the safety application to prevent NoSaMos to access the application's safety-relevant resources.

Furthermore, the PBRIDGE can restrict read and write access to individual I/O modules based on the origin of the access and its state (user mode/supervisor mode).

Finally, the register protection included allows individual registers to be "locked" against any manipulation without unlocking.

These safety mechanisms are further described in "System Memory Protection Unit (SMPU)".

75.4.1.9 FCCU supervision (FOSU)

As the FCCU is a central component in reacting to errors, it is itself supervised even though an error in it can only cause a latent failure. This supervision is provided by the FOSU (FCCU Output Supervision Unit). The FOSU receives failure indications at the same time as the FCCU. Unless the respective failure is switched off, the FOSU will observe the outputs of the FCCU (IRQ, RESET, FCCU_F[n]). If the FCCU does not react—within a predefined interval—on one of those outputs to the incoming failure indication, the FOSU assumes the FCCU has failed and causes a reset.

The FOSU does not require any configuration by software.

75.4.2 Dual-core lockstep

The MPC5746R duplicates its safety-relevant processing elements and compares their operation in Lockstep mode (LSM). This Safety core consists of two cores, Checker Core_0 and Master Core_0, and as far as software is concerned they behave as one core. Main Core_0 is the main execution core of the pair, where Checker Core_0 follows the execution of the Master core in lockstep.

The processing elements which are replicated contain:

- Core
- Cache control
- Local memory control
- Core Memory Protection Unit (CMPU)
- Core System Bus Interface, including E2E ECC logic
- eDMA controller

Together each set of replicated elements forms a channel (for example, the Main channel and the Checker channel).

Elements that are not replicated are:

- Cache and local memory arrays
- INTC
- Interconnect (XBAR)
- RAM controller
- Flash memory controller

NOTE

This is not a complete list of all non-replicated elements. But the elements listed are sometimes replicated in similar lockstep architectures.

Equivalent operation of replicated resources is supervised by comparators on all functional signals leaving the channels for the rest of the MCU. Any operational deviations between the supervised signals will cause the FCCU to be notified of the discrepancy.

The Checker Core does not have a direct connection to the XBAR. All of the outputs of Checker Core_0 that target the XBAR (as well as any other non-duplicated resource, like local memories) will end in an RCCU for verification, and all the inputs to Checker Core_0 from the XBAR will be split off from the Main Core_0 XBAR inputs.

An abstract view of the implementation including delays and RCCUs is shown in the following figure, but it does not show the local memory.

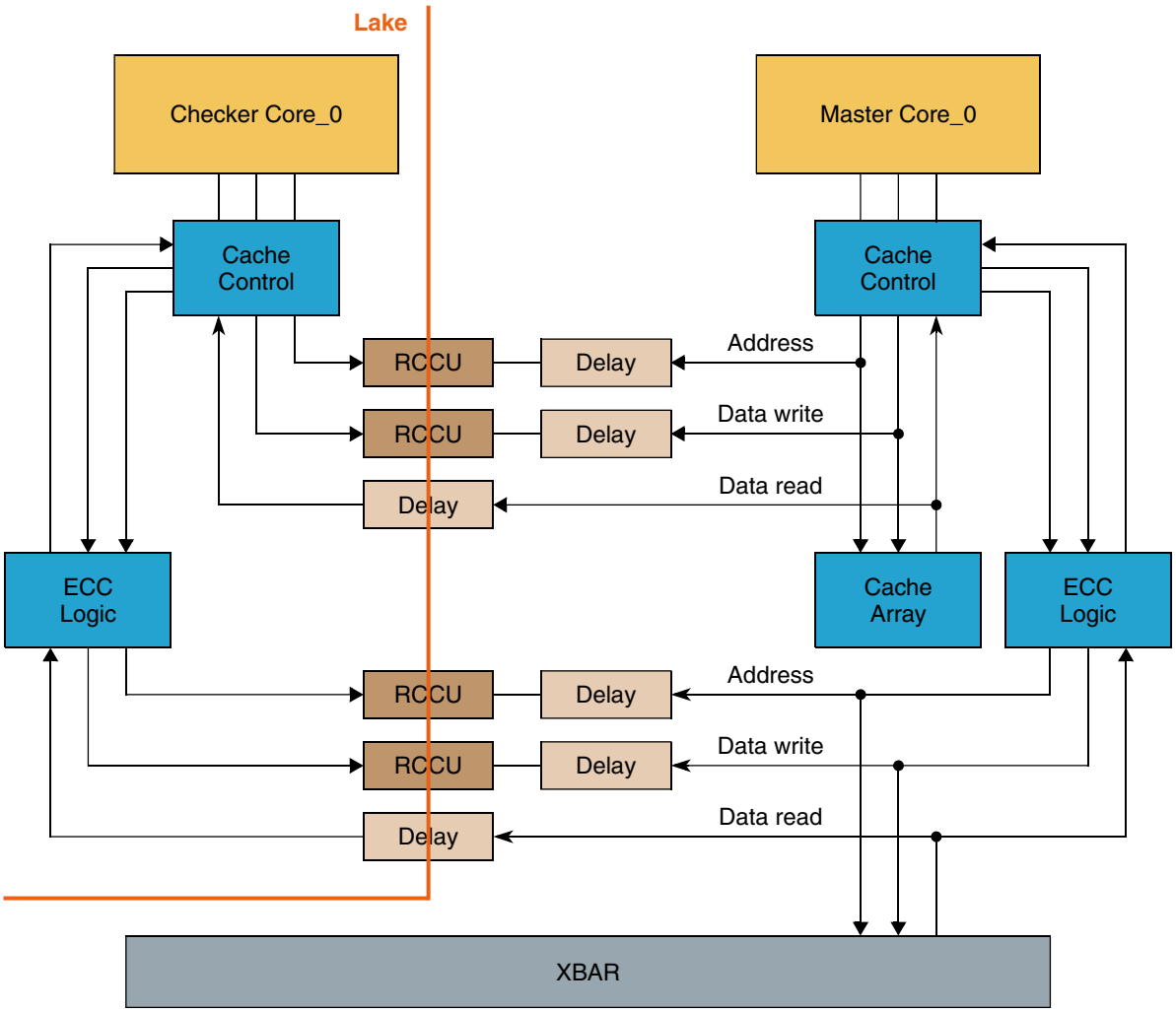


Figure 75-2. Dual core concept

A delay of two clock cycles exists between the Master and Checker channel to reduce their susceptibility to CCFs caused by power and clock disturbances. Outputs of the Master core (such as XBAR addresses) are registered and delayed before being forwarded to the RCCUs while inputs signals (such as XBAR read data) are registered and delayed before being fed to the Checker channel.

LSM can be disabled during boot by reprogramming the flash memory. LSM is not guaranteed when Core_0 is in debug mode; cores (main and checker) are working, but the RCCUs are disabled. If disabled, the Checker Core (Core_0 - Checker) and the RCCUs are constantly off. No dynamic switching is possible between lockstep on/off, a reset is needed to reestablish LSM. No Decoupled Parallel Mode (DPM) is available (Main Core_0 and Checker Core_0 cannot operate as two independent cores running different software). Main Core_0 is able to operate as an independent core if Checker Core_0 is disabled. The status of LSM for the Safety core can be verified by checking the

Core Status register (ME_CS) in the Mode Enable module (MC_ME). If both Main Core_0 and Checker Core_0 are on, or both off, the Safety Core is in LSM. See the "Core Status Register (ME_CS)" section in the "Mode Entry Module (MC_ME)" chapter.

75.4.3 Common Cause Failure measures

Various measures are included to prevent CCFs from endangering the effectiveness of the replication of the Safety Core or of the peripherals. These measures include physical separation of the components on the die, routing restrictions and supervision of clock, power, temperature, test and debug signals. In general these measures are independent from the software.

Also, there are several functional configuration registers throughout the MCU where, if they erroneously change, they can affect the execution of the MCU's safety function and, at the same time, disable the respective safety mechanism. These registers in particular are either protected against bit flips or those flips are detected by independent measures. These same registers are also protected against accidental software writes by employing as well the register protection safety feature.

75.4.4 ECC

Error correcting codes are used for end-to-end protection from cores to system storage as well as for individual protection of peripheral RAMs.

75.4.4.1 ECC for storage

All storage (System RAM, local memory, flash memory, peripheral RAM) used in normal operation is protected by ECC with SECDED (Single Error Correct and Double Error Detect). Exceptions to this are noted below:

- The caches in the cores are protected by EDC (Error Detection Code), which detects but does not correct errors.

Some memories, particularly system storage, use an ECC computed over data and address to detect data and addressing faults (no/wrong/multiple selection). In addition, these same memories include dedicated measures against addressing and control faults (such as address/control feedback). This is different for storage related to PeMos. PeMos in general use an ECC without address error protections and also do not include additional measures to detect them.

75.4.4.2 All-x words and ECC

There is a special case for legal ECC values in the MPC5746R. Memory entries that are all zeros (All-0) or all ones (All-1), including the ECC parity bits, are not legal for memory that is checked by ECC. The flash memory allows All-1, corresponding to the status of an erase block, as a valid codeword.

Memories that include addresses in the ECC calculation do not specifically protect against All-0 or All-1. This means that for some addresses All-0 or All-1 may be legal.

All-0 and All-1 memory content is indicated in different ways. For memories that do not include address into the ECC calculation, All-0 and All-1 will be uncorrectable errors. For all memories that include address into the ECC code-bit calculation, since the ECC checkbits depend on the address, it is not possible to generate an uncorrectable error indication for all the possible addresses. Therefore, an All-x content may result in a correctable error.

Notice that for flash memory, additional dedicated safety mechanisms exist to detect failures that have the potential of leading to an All-1 word (see the "Flash memory controller" chapter for more details on flash memory safety mechanisms).

75.4.4.3 ECC failure handling

Single-bit and double-bit errors (correctable and uncorrectable errors) are signaled to the FCCU unless filtered by the MEMU. The MEMU (see section "Memory Error Management Unit (MEMU)" in this chapter) may filter ECC error notification for known ECC error addresses (known permanent correctable errors). Actual implementation will signal errors, not to the FCCU, but to the MEMU which filters, then forwards unfiltered notifications in an aggregated manner to the FCCU.

75.4.4.4 Memory Error Management Unit (MEMU)

The MEMU is responsible for the collection and reporting of error events associated with ECC logic used on SRAM (system RAMs in this context are those RAMs with "System RAM" memory classification), peripheral RAM, and flash memory. When ECC error events occur, the MEMU receives an error signal that causes an event to be recorded, and possibly sets corresponding error flags that are reported to the FCCU.

The MEMU stores the addresses of ECC errors that have occurred in a table as follows:

- Uniqueness of the addresses in the table is ensured. For example, if an address is already stored in the MEMU's table, correctable errors at that address will no longer be reported to the FCCU.
- Software can read and modify the MEMU table and remove individual entries (by marking them as invalid). For example, this allows software to invalidate a new entry in the MEMU table and wait for a repeat occurrence of the ECC error indicating a permanent error in memory.
- If the MEMU table overflows, an additional signal (instead of the ECC error signal) is sent to the FCCU.

As the MEMU aggregates address information from several sources, it might not be able to process all simultaneously arriving reports. This is called a simultaneous overflow and is signaled to the FCCU as a buffer overflow (see "Memory Error Management Unit (MEMU)" chapter for details).

75.4.4.4.1 Interface to ECC units

The MEMU receives data according to the following interface per ECC unit connected:

- Whether the error is a Single-Bit-Error (SBE) or Uncorrectable Error (UCE) type
- Address of the memory where the error occurred
- A configuration specifying whether the ECC unit is attached to a vital memory (SRAM, local memory, and system RAM of Safety Core), to flash memory, or to a peripheral RAM

If an error is signaled, it is compared against all errors known for that storage:

NOTE

If a previously known error has been marked invalid by software, no comparison against the address will occur.

- If no entry of that address is already in the buffer, it will be added into the appropriate table depending on the SBE versus UCE indicator

NOTE

For this comparison the bit position will not be taken into account. This is in contrast to MBIST reporting.

- If there is no free entry left in that buffer, the overflow flag is set.

If a valid entry of that address is in the buffer, the following occurs:

- If the entry indicates an SBE in that address, and the error indicated is correctable, a new entry is added in the MEMU buffer.
- In all other cases, nothing is changed in the entry.

Chapter 76
Cyclic Redundancy Check (CRC) Unit

76.1 Introduction

The Cyclic Redundancy Check (CRC) computing unit is dedicated to the computation of CRC, off-loading the CPU. Each context has a separate CRC computation engine in order to allow the concurrent computation of the CRC of multiple data streams. The CRC computation is performed at speed without wait-state insertion. Bit-swap and bit-inversion operations can be applied on the final CRC signature. Each context can be configured with one of three hard-wired polynomials, normally used for most of the standard communication protocols. The data stream supports multiple data width (byte/halfword/word) formats.

76.2 Main features

- Single context is supported
- Zero-wait states during the CRC computation (pipeline scheme)
- Four hard-wired polynomials (two CRC-8, CRC-32 Ethernet, and CRC-16-CCITT)
- Support for byte, halfword, or word width of the input data stream

76.2.1 Standard features

- Peripheral bus interface
- CRC-8 VDA CAN
- CRC-16-CCITT

- CRC-32 Ethernet (see note below)
- CRC-8-H2F Autosar polynomial

NOTE

Although the CRC-32 Ethernet polynomial is used for CRC generation, the generation algorithm differs (MISR) from the one used by the Ethernet protocol (IEEE 802).

76.3 Block diagram

The top level diagram of the CRC module is given in the following figure. Refer to Main features for number of contexts in this module.

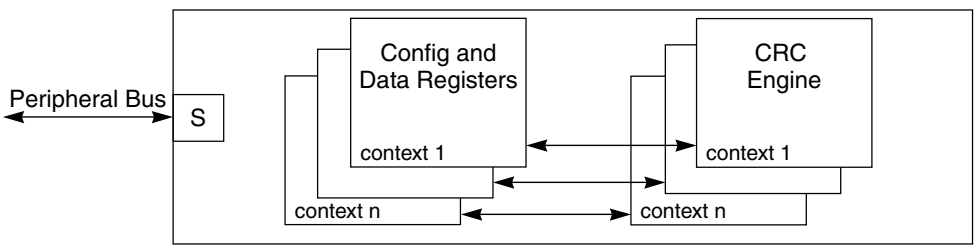


Figure 76-1. CRC top level diagram

76.4 External signal description

The CRC module does not generate any external signals.

76.4.1 Peripheral bus interface

The peripheral bus interface is a slave bus used for configuration and data streaming (CRC computation) purposes via CPU or DMA. The following bus operations (contiguous byte enables) are supported:

- Word (32 bits) data write/read operations to any registers
- Low and high halfword (16 bits, data[31:16] or data[15:0]) data write/read operations to any registers

- Byte (8 bits, data[31:24], data[23:16], data[15:8], or data[7:0]) data write/read operations to any registers
- Any other operation (free byte enables or other operations) must be avoided.

The CRC module generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral
- Any write/read operation different from byte/halfword/word (free byte enables or other operations) on each register

The registers of the CRC module are read/write accessible in each access mode:
- user
- supervisor

The following summarizes bus operation performance:

- Zero wait states (single bus cycle) for each write/read operation to the CRC_CFG and CRC_INP registers
- Zero wait states (single bus cycle) for each write operation to the CRC_ CSTAT register
- Double wait states (3 bus cycles) for each read operation to the CRC_ CSTAT or CRC_OUTP registers immediately following (next clock cycle) a write operation to the CRC_CSTAT, CRC_INP, or CRC_CFG registers belonging to the same context; in all the other cases, no wait states are inserted

76.5 CRC memory map and registers

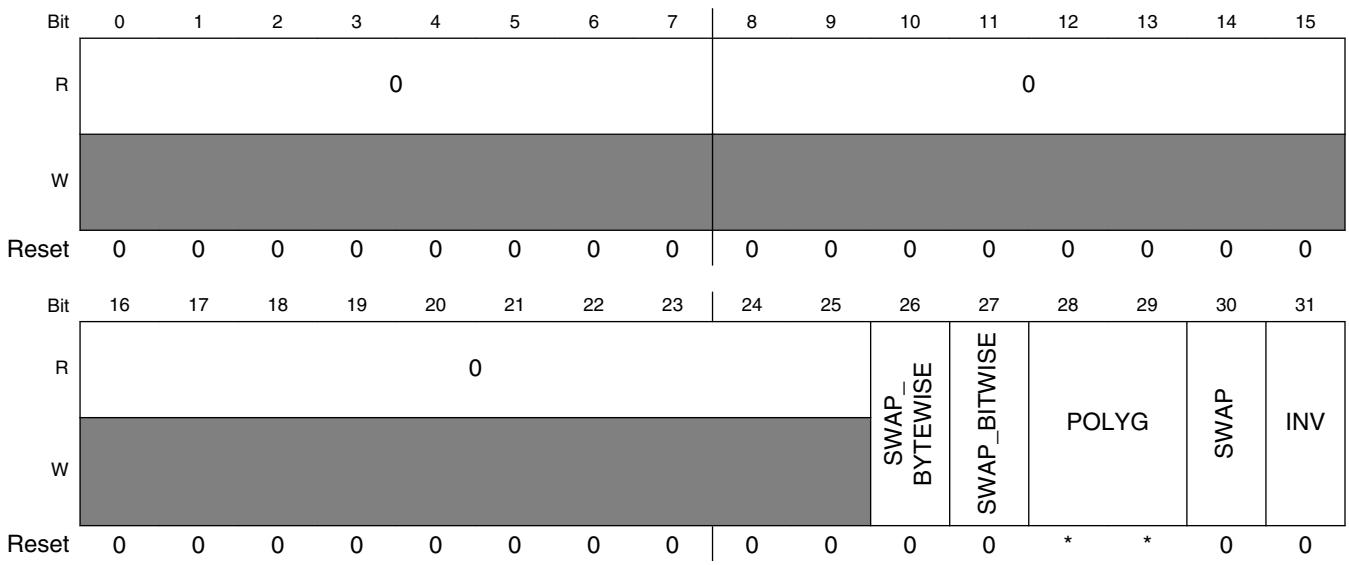
CRC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Configuration Register (CRC_CFG)	32	R/W	See section	76.5.1/3232
4	Input Register (CRC_INP)	32	R/W	0000_0000h	76.5.2/3233
8	Current Status Register (CRC_CSTAT)	32	R/W	FFFF_FFFFh	76.5.3/3234
C	Output Register (CRC_OUTP)	32	R	FFFF_FFFFh	76.5.4/3234

76.5.1 Configuration Register (CRC_CFG)

Access: User read/write.

Address: 0h base + 0h offset = 0h



* Notes:

- POLYG field: Reset value is a function of the number of contexts, N. When N%2 = 0, reset value = 01b. When N%2 = 1, reset value = 00b.

CRC_CFG field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 SWAP_ BYTEWISE	Swap CRC_INP byte-wise **NOTE:** INV and SWAP bits are set to 1 for CRC-32 polynomial calculations. 0 Do not swap 1 Perform byte-wise swap on CRC_INP input data internally for CRC-32 polynomial calculations.
27 SWAP_BITWISE	Swap CRC_INP bit-wise 0 Do not swap 1 Perform bit-wise swap on CRC_INP input data internally for CRC-8 and CRC-16 polynomial calculations.
28–29 POLYG	Polynomial selection

Table continues on the next page...

CRC_CFG field descriptions (continued)

Field	Description
	This bit can be written only during the configuration phase. 00 CRC-CCITT polynomial 01 CRC-32 polynomial 10 CRC-8 polynomial 11 CRC-8-H2F Autosar polynomial
30 SWAP	Swap selection This bit can be written only during the configuration phase. The swap operation is a bit-by-bit swapping of the content. 0 No swap selection applied on the CRC_OUTP content 1 Swap selection (MSB to LSB, LSB to MSB) applied on the CRC_OUTP content; in case of CRC-CCITT polynomial, the swap operation is applied on the 16 least-significant bits
31 INV	Inversion selection This bit can be written only during the configuration phase. The inversion operation is a complement (or negation) of the content. 0 No inversion selection applied on the CRC_OUTP content 1 Inversion selection (bit x bit) applied on the CRC_OUTP content

76.5.2 Input Register (CRC_INP)

Access: User read/write.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	INP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CRC_INP field descriptions

Field	Description
0–31 INP	Input data for the CRC computation This register can be written with word-, halfword- (high or low), or byte-wide writes in any sequence. Only the bits written are fed to the CRC engine. In case of halfword write operation, the bytes must be contiguous.

76.5.3 Current Status Register (CRC_CSTAT)

Access: User read/write.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CSTAT																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CRC_CSTAT field descriptions

Field	Description
0–31 CSTAT	CRC signature status This register includes the current status of the CRC signature. No bit swap and inversion are applied to this register. In the case of the CRC-CCITT polynomial, only the 16 least-significant bits are significant. The 16 most-significant bits are set to 0 during the computation. In the case of the CRC-8 polynomial, only the 8 least-significant bits are significant. The 24 most-significant bits are set to 0 during the computation. The CSTAT register can be written at byte, halfword, or word. This register can be written only during the configuration phase.

76.5.4 Output Register (CRC_OUTP)

Access: User read-only.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OUTP																															
W																																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CRC_OUTP field descriptions

Field	Description
0–31 OUTP	Final CRC signature This register includes the final signature corresponding to the CRC_CSTAT register value eventually swapped and inverted. In the case of the CRC-CCITT polynomial, only the 16 least-significant bits are significant. The 16 MSB bits are set to 0 during the computation. In the case of the CRC-8 polynomial, only the 8 least-significant bits are significant. The 24 most-significant bits are set to 0 during the computation.

76.6 Functional description

The CRC module supports the CRC computation for each context. Each context has its own complete set of registers, including the CRC engine. The data flow of each context can be interleaved. The data stream can be structured as a sequence of bytes, halfwords, or words. The input data sequence is provided, eventually mixing the data formats (byte, halfword, and word), writing to the input data register (CRC_INP).

The data stream is generally executed by *n* concurrent DMA data transfers (mem2mem) where *n* is less or equal to the number of contexts.

The standard generator polynomials are given in Figure 76-6, Figure 76-7, and Figure 76-8 for the CRC computation of each context. Two separate registers are available: CRC_INP for writing data and CRC_CSTAT for retrieving the (accumulated up to now) CRC.

$$x^8 + x^4 + x^3 + x^2 + 1$$

Figure 76-6. CRC8 VDA CAN

$$x^{16} + x^{12} + x^5 + 1$$

Figure 76-7. CRC-CCITT (x.25 protocol)

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Figure 76-8. CRC-32 (Ethernet protocol)

$$x^5 + x^3 + x^2 + x + 1$$

Figure 76-9. CRC-8-H2F (Autosar 4.0 protocol)

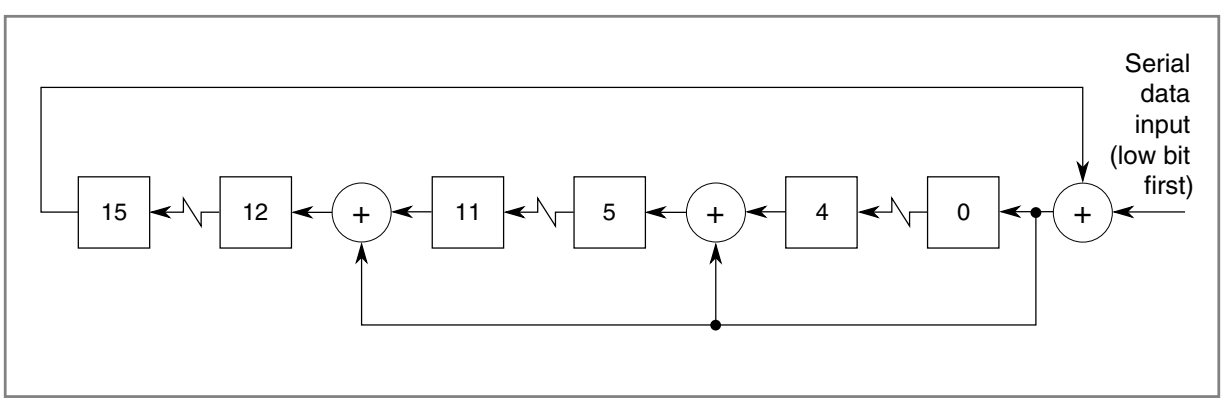


Figure 76-10. CRC-CCITT engine concept scheme

The initial seed value of the CRC can be programmed by initializing the CRC_CSTAT register. A diagram illustrating serial data loading of the CRC engine is shown in Figure 76-10 for the CRC-CCITT. Note that, conceptually, the least-significant, or "right-most," bit shown in Input Register (CRC_INP), is fed first into the engine. The actual implementation executes the CRC computation in a single clock cycle (parallel data loading). A pipeline scheme has been adopted to decouple the IPS bus interface from the CRC engine in order to allow the computation of the CRC at speed (zero wait states).

If the CRC signature is used for encapsulation in the data frame of a communication protocol (for example, SPI, etc.), a bit swap (high bit for low bit or low bit for high bit) and/or bit inversion of the final CRC signature can be applied to CRC_OUTP before transmitting the CRC.

Use of the CRC module is summarized in the flow chart given in Figure 76-11.

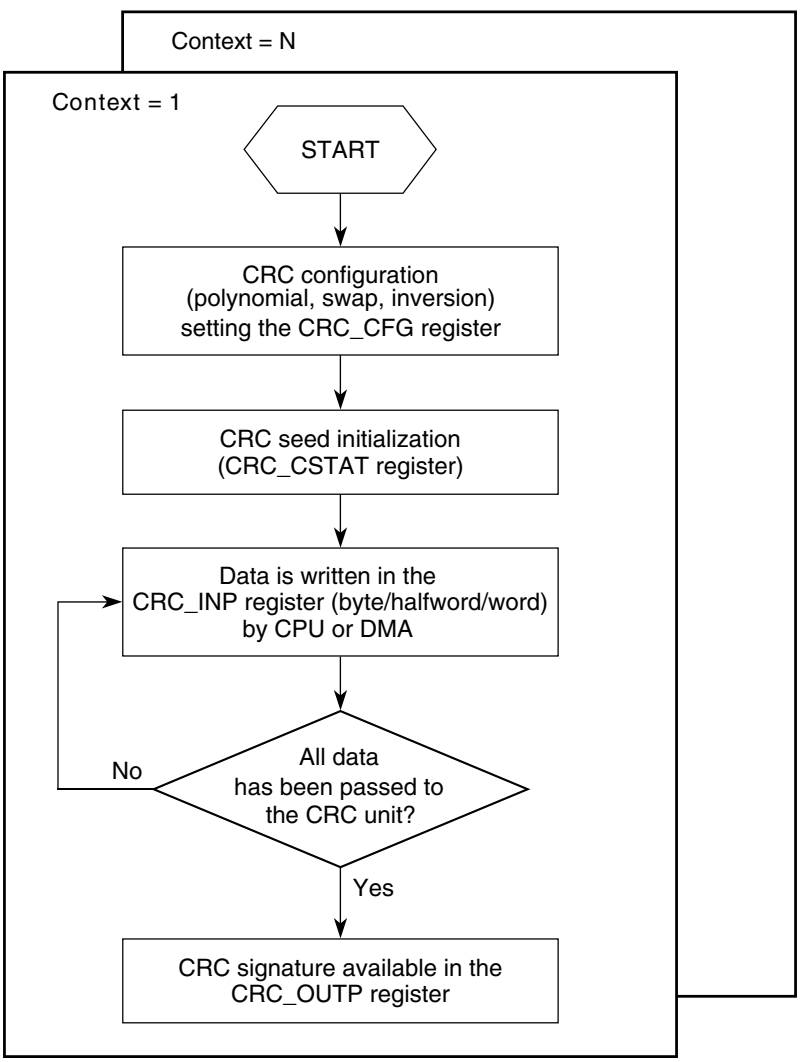


Figure 76-11. CRC computation flow

76.7 Use cases

76.7.1 Programming example

The number of contexts depends on the application. The overall number of contexts for the CRC peripheral depends on the number of peripherals that concurrently require intervention by the CRC module. Two main use cases are considered:

- Calculation of the CRC of the configuration registers during the process safety time
- Calculation of the CRC on the incoming/outcoming frames for the communication protocols (not protected with CRC by definition of the protocol itself) used as a safety-relevant peripheral

The signature of the configuration registers is computed correctly only if these registers do not contain any status bit.

Assuming that the DMA engine has *n* channels (greater than or equal to the number of contexts) configurable for the following types of data transfer—mem2mem, periph2mem, mem2periph—the following sequence, as shown in the following figure, is applied to manage the transmission data flow:

1. DMA/CRC module configuration (context x, channel x) by CPU
2. Payload transfer from the MEM to the CRC module (CRC_INP register) after MSB-to-LSB change (input mirroring) to calculate the CRC signature (phase1) by DMA (mem2mem data transfer, channel x)
3. CRC signature copy from the CRC module (CRC_OUTP register) to the MEM (phase 2) by CPU
4. Data block (payload + CRC) transfer from the MEM to the PERIPH module (for example, SPI Tx FIFO) (phase 3) by DMA (mem2periph data transfer, channel x)

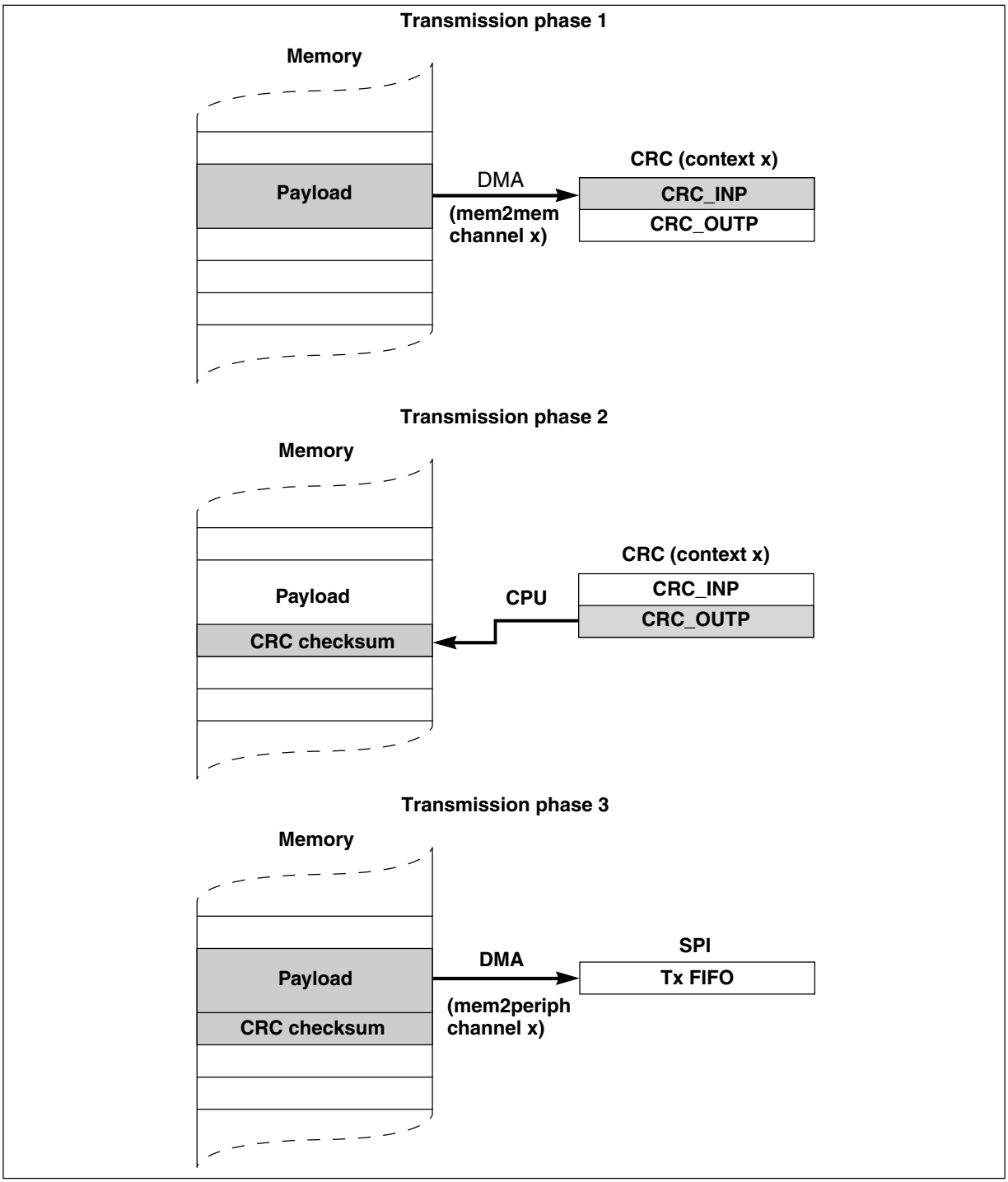


Figure 76-12. DMA-CRC transmission sequence

76.7.2 Register programming

- INV and/or SWAP (affecting the output only) can be applied to CRC-8, CRC-16, and CRC-32. Both can be applied together.

- SWAP_BITWISE (affecting the input only) can be applied for all polynomials if required by an application.
- SWAP_BYTEWISE (affecting the input only) can be applied for CRC-16 and CRC-32 polynomials only.
- SWAP_BITWISE has priority over SWAP_BYTEWISE when both are applied together.
- When generating CRC-32 for the Ethernet standard the user needs to set SWAP_BYTEWISE together with INV and SWAP.
- When generating CRC-16 the user needs to set SWAP_BITWISE bit.

Chapter 77
Memory Error Management Unit (MEMU)

77.1 Memory Error Management Unit (MEMU) chip-specific module information

77.1.1 MEMU system connections

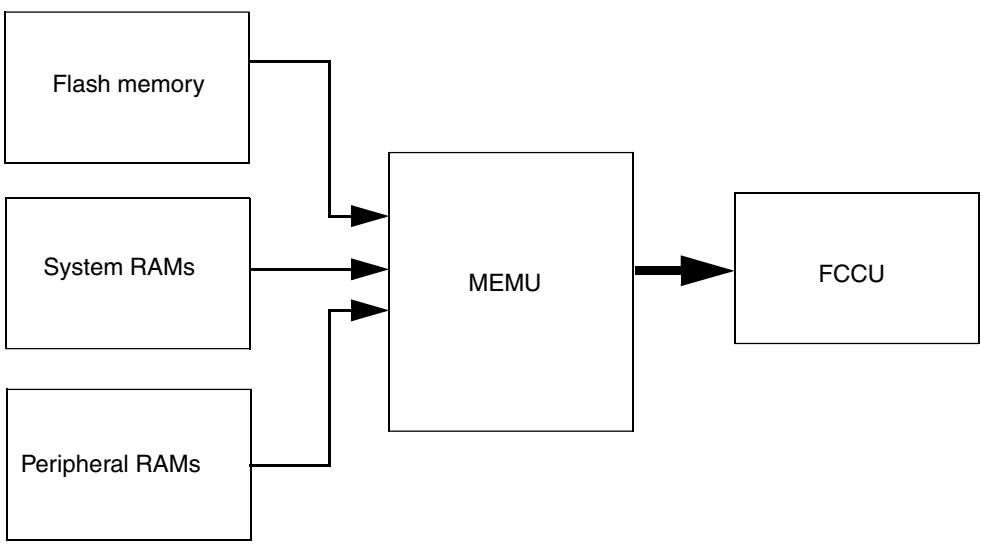


Figure 77-1. MEMU system connections

77.1.2 MEMU error sources

The following table shows the MEMU error sources for each instance configured on this chip.

Table 77-1. MEMU error sources

Instance	Value
Number of system RAM unique error sources	18
Number of peripheral RAM unique error sources	12
Number of flash memory unique error sources	2

77.1.3 Reporting table implementation

MEMU establishes the following categories of error reporting:

- Flash memory error reporting
- System RAM error reporting (for errors when System RAM is accessed by CPUs)
- Peripheral RAM error reporting (for errors in any RAM that is not CPU accessible as well as errors when System SRAM is accessed by peripherals connected to concentrators - Ethernet, SIPI, or eDMA)

Each category of error reporting has two tables associated with it, these are:

- Correctable error reporting table
- Uncorrectable error reporting table

Each entry in a reporting table corresponds to a unique error event. The number of entries supported by each table depends upon the error source.

The table below describes the number of entries in each reporting table for the different error sources.

Table 77-2. Number of entries in each reporting table

Error source	Number of entries in correctable error reporting table	Number of entries in un-correctable error reporting table
Flash memory	20	1
System RAM	10	1
Peripheral RAM	2	1

77.1.4 Concurrent overflow register (OFLW) implementation

Errors are reported to the MEMU from these categories:

- Flash memory
- System RAM
- Peripheral RAM

In an overflow condition, it is possible to identify the unique error source that caused the overflow. Corresponding error sources for the OFLW register details the corresponding error source to the respective bit-field in the Concurrent Overflow registers.

Positions not explicitly listed in the tables below are reserved.

Table 77-3. Corresponding error sources for system RAM OFLW0 register

Bit number	Error source
0	Core0 d-ahb error
1	Core0 i-ahb error
2	Core0 Data TCM error
3	Core0 Instruction TCM error
5	Core0 Instruction cache error
6	Core1 d-ahb error
7	Core1 i-ahb error
8	Core1 Data TCM error
9	Core1 Instruction TCM error
11	Core1 Instruction cache error
12	PRAM
13	PBRIDGE_0 Decorated storage
14	PBRIDGE_1 Decorated storage
15	PRAM decorated storage
16	Core0 backdoor TCM decorated storage
17	Core1 backdoor TCM decorated storage

Table 77-4. Corresponding error sources for peripheral RAM OFLW0 register

Bit number	Error source
0	DMA TCD RAM
2	2:1 concentrator
3	3:1 concentrator
4	PBRIDGE_0
5	PBRIDGE_1
6	FEC RIF RAM
7	FEC MIB RAM
8	CAN0
9	CAN1
10	CAN2
11	CAN3

Table 77-5. Corresponding error sources for flash memory OFLW0 register

Bit number	Error source
0	Flash Port0 ECC Error
1	Flash Port1 ECC Error

77.1.5 Uncorrectable error in flash memory can cause uncorrectable error in SRAM

Be aware of the following scenario where an uncorrectable error in flash memory can cause an uncorrectable error in SRAM to be reported. Each error triggers a separate FCCU response.

1. An uncorrectable error in flash memory occurs, and C55FMC_MCR[EER] sets to 1.
2. MEMU_FLASH_UNCERR_STS[VLD] sets to 1, and MEMU_ERR_FLAG[F_UCE] sets to 1. MEMU_FLASH_UNCERR_ADDR[ERR_ADD] records the address on which the error was detected.
3. The FCCU responds to a fault on NCF[29].

When the core register field E2ECTL0[DECCEN] is 1, these events follow:

1. The detected error in flash memory results in an exception/machine check.
2. MEMU_SYS_RAM_UNCERR_STS[VLD] sets to 1, and MEMU_ERR_FLAG[SR_UCE] sets to 1. MEMU_SYS_RAM_UNCERR_ADDR[ERR_ADD] records the address on which the error was detected.
3. The FCCU responds to a fault on NCF[23].

77.2 Introduction

The MEMU is responsible for collection and reporting of error events associated with ECC (Error Correction Code) logic used on:

- System RAM

- Peripheral RAM
- Flash memory

When any of the following events occur the MEMU receives an error signal which causes an event to be recorded and corresponding error flags to be set and reported to FCCU (Fault Collection and Control Unit).

- Correctable Error: It comprises the following:
  - Single Bit error in the data part that is detected via ECC for:
    - System RAM
    - Peripheral RAM
    - Flash memory
- Uncorrectable Error: It comprises the following:
  - Multiple bit error that is detected via ECC for:
    - System RAM
    - Peripheral RAM
    - Flash memory
  - Addressing errors and unused data bit errors detected by ECC logic.

The MEMU system connections are chip-specific; see the chip-specific MEMU information.

When multiple errors are indicated from various sources at same instant, an Overflow can be indicated by the MEMU to the FCCU. Overflow can also be indicated if the reporting table entries are full and a new unique error is reported by the system.

MEMU processes the errors based on whether they are correctable or uncorrectable (from ECC logic), independent of the source generating these errors.

The MEMU has multiple instances of the basic reporting block; one each for:

- System RAM ECC
- Peripheral RAM ECC
- Flash memory ECC

Each instance has two reporting tables. Correctable errors are reported in one table, and the uncorrectable errors are reported in the other.

See Design overview for the reporting block details.

77.3 Features

The MEMU has the following features:

- Supports up to 128 error sources per the following reporting blocks (the number of error sources are chip-specific; see the chip-specific MEMU information):
  - System RAM ECC
  - Peripheral RAM ECC
  - Flash memory ECC
- Support for error reporting from the following category of error sources (for ECC):
  - System RAM
  - Peripheral RAM
  - Flash memory
- Unique reported errors are logged into the module's reporting table which is accessible to CPU via memory mapped CPU register programming interface.
- MEMU handles overflow during error assertion and reports status accordingly.
- Unique errors are stored in correctable and non correctable section of the reporting table and corresponding indication is generated to the FCCU.
- CPU can program the known errors into the reporting table to avoid their re-reporting by MEMU.
- CPU can clear the reporting table errors.
- The reporting table for uncorrectable errors supports only 1 entry.

77.4 Block diagram

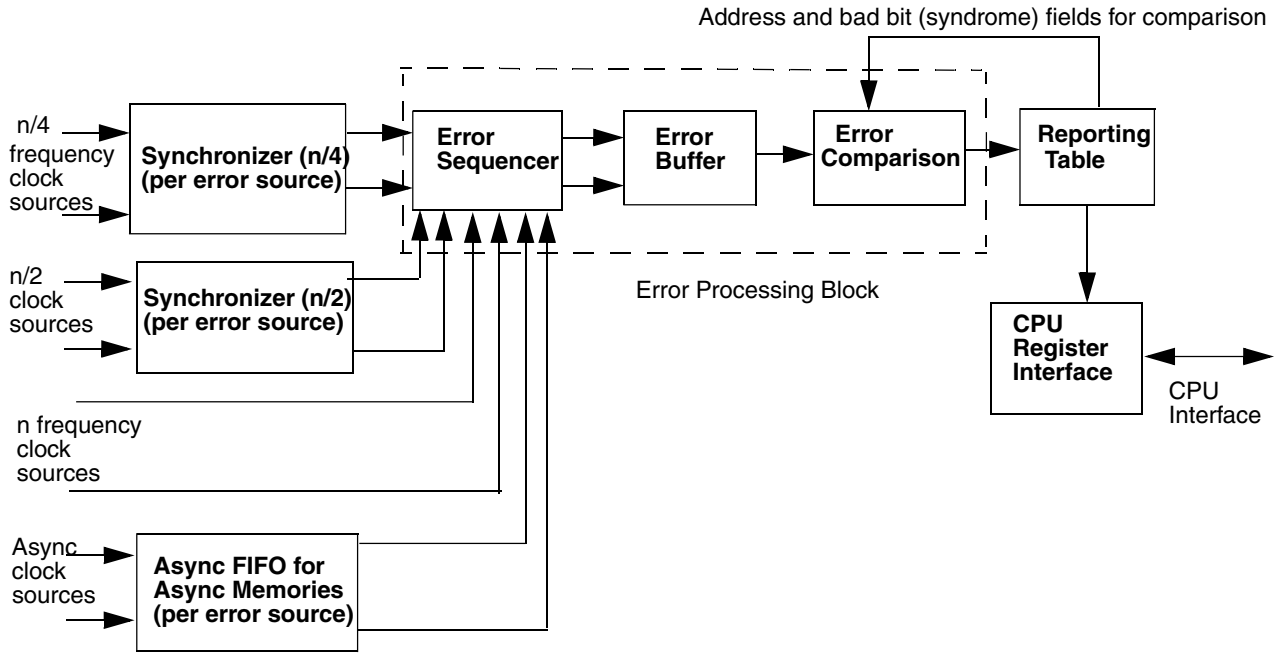


Figure 77-2. MEMU block diagram

77.5 Design overview

The MEMU should be enabled on power-on reset. The user software can write known error addresses into the reporting table to prevent reporting of those errors addresses to FCCU should they later be accessed in operation. All writes into the reporting table by the CPU that assert the Valid Bit (of the respective reporting table entry) will be indicated as a new error detected and will be reported to the FCCU. This feature can also be used as self-checking option of the MEMU and the MEMU-FCCU connection.

The MEMU design does not limit any particular sequence or errors. See Handling overflows (Multiple error reporting), for details on overflow behaviors.

The basic algorithm flow/architecture for error reporting in the MEMU is summarized below:

1. Synchronizer block

   For asynchronous inputs, a FIFO-based structure (per asynchronous channel) is implemented. All the error inputs (correctable error reported, uncorrectable error reported, error address and bad bits/syndrome) are concatenated and stored in the

FIFO. While storing the entry in the FIFO, a duplicity check is performed with the previously stored entry (the entry which was stored in the last cycle). If both the entries are duplicate the new entry is deleted and the pointers are rolled back to the old value, otherwise the new entry is stored. Whenever there is an overflow (FULL condition) in the Asynchronous FIFO, it is denoted by setting the Error Buffer Overflow or the EBO flag to the FCCU and the corresponding bit in the Concurrent Overflow Register is set.

For synchronous inputs, inputs are sampled whenever the corresponding clock synchronizing signal is asserted for a particular channel.

NOTE

On this chip, all error inputs are synchronous.

2. Error Processing block

   This block is used to process the incoming errors coming from the device.

   - Error Sequencer is used for sequencing the errors so that these are processed one per clock. In case of more than one errors detected by the sequencer logic, one is sent for the processing (error comparison block), the other is stored in the error buffer and the rest are marked as overflows in the Concurrent overflow register.
   - All the sequences of incoming errors are supported. However, in case of multiple errors being asserted in back to back clock cycles, MEMU will process the errors being forwarded by the sequencer logic and register the rest of the errors as Concurrent Overflows in the Concurrent Overflow Register.
   - While storing the error in the error buffer, uniqueness is determined by comparing the error address, correctable or uncorrectable error type and the bad bit/syndrome (depending on ECC logic) of the incoming error with the one stored in buffer. If the error buffer is full, and the incoming error to be stored in the error buffer is not yet in the error table, Error Buffer Overflow (EBO) is registered and the corresponding bit in the Concurrent Overflow Register is set., else the incoming error is dropped.
   - While doing the error buffer uniqueness check, the type of error i.e Correctable Error (CE) or Uncorrectable Error UCE) is also compared, hence even if the Error Address and the Bad bit/syndrome fields (Depending on whether the incoming error is generated from ECC logic) of the incoming error matches with the error buffer, if the type of errors (CE and UCE) differ they are treated as unique

- Determine the type and uniqueness of errors in the error comparison block and store them in the correct reporting table (correctable or uncorrectable)
- Generate signals to assert the necessary flags. Be aware that it takes several MEMU clock cycles (<=5 unless the error gets temporarily stored in the error buffer due to a collision) for an error report to cause a signal to be sent to the FCCU.

3. Reporting table

   This is a pre-defined table in the MEMU to store the details for the device. For the reporting table if the entry is unique it is stored in the reporting table. If the entry is not unique it is not stored and discarded.

   The MEMU asserts proper flags when error is stored in the reporting table or an overflow is asserted. Flags are also asserted when the software writes the VLD bit to the reporting table.

   The CPU can clear the flags signaling errors or overflows to the FCCU directly by writing 1 at the corresponding valid bit of the status register (i.e. ..._CERR_STS or ..._UNCERR_STS). The clearing of flags is totally independent from the status of VLD bits in the reporting table.

   The reporting table sizes are chip-specific; see the chip-specific MEMU information.

4. Register block

   The CPU register programming interface provides the memory-mapped registers necessary for the CPU to access the reporting table and other control and status registers for the modules. Flags to FCCU will be asserted on write by CPU to the valid bit or by MEMU itself if a unique error is stored. However, in the situation when both MEMU and the CPU are accessing the same Register location, a wait cycle will be signaled to the CPU and the CPU command will be completed in the next clock cycle.

   The software, in principle, is not supposed to write VLD to 1 if it is already 1.

   The CPU register interface provides the decoding of the peripheral signals to allow access to the reporting table and other registers in module.

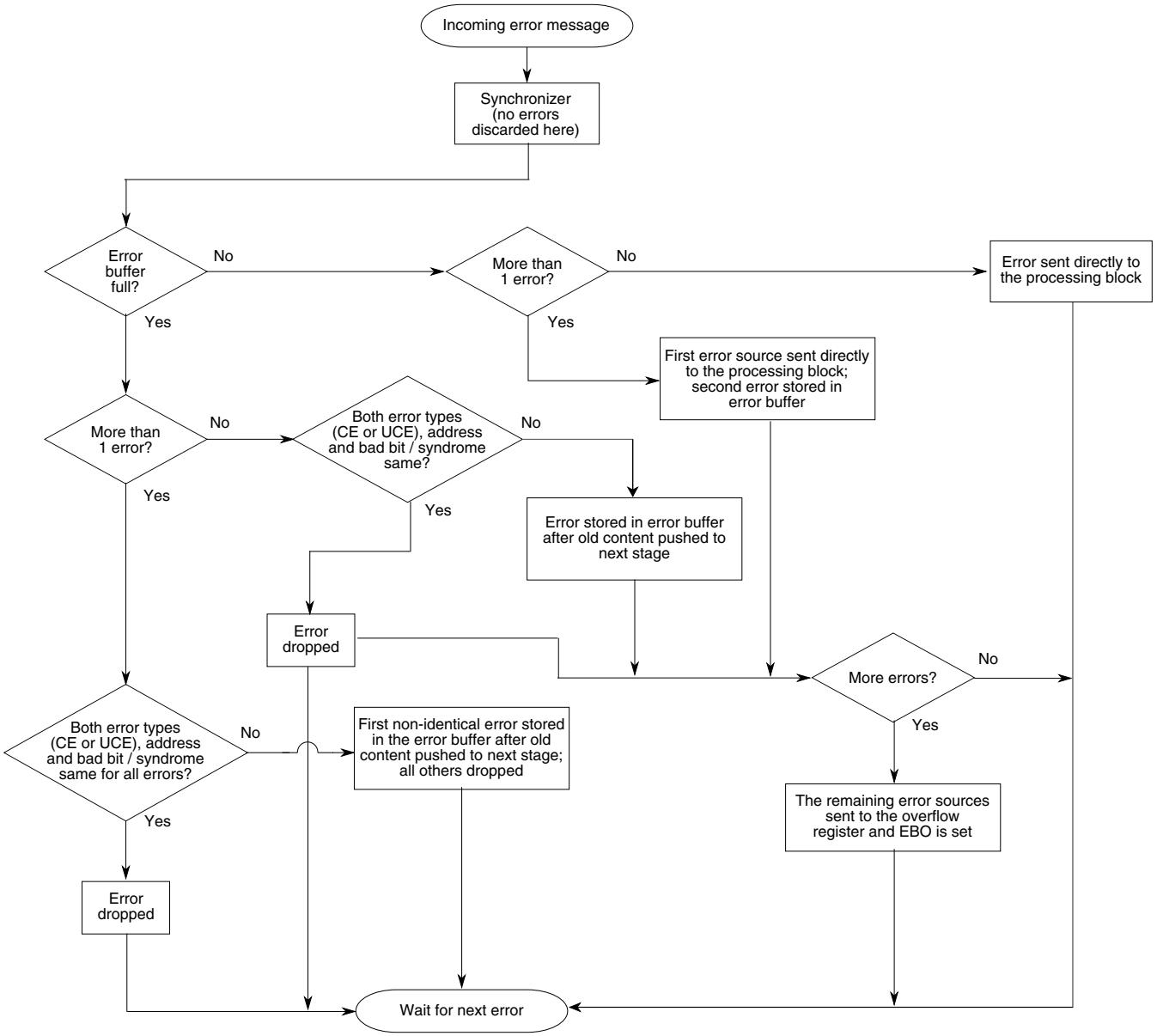


Figure 77-3. MEMU error buffer uniqueness check flow chart

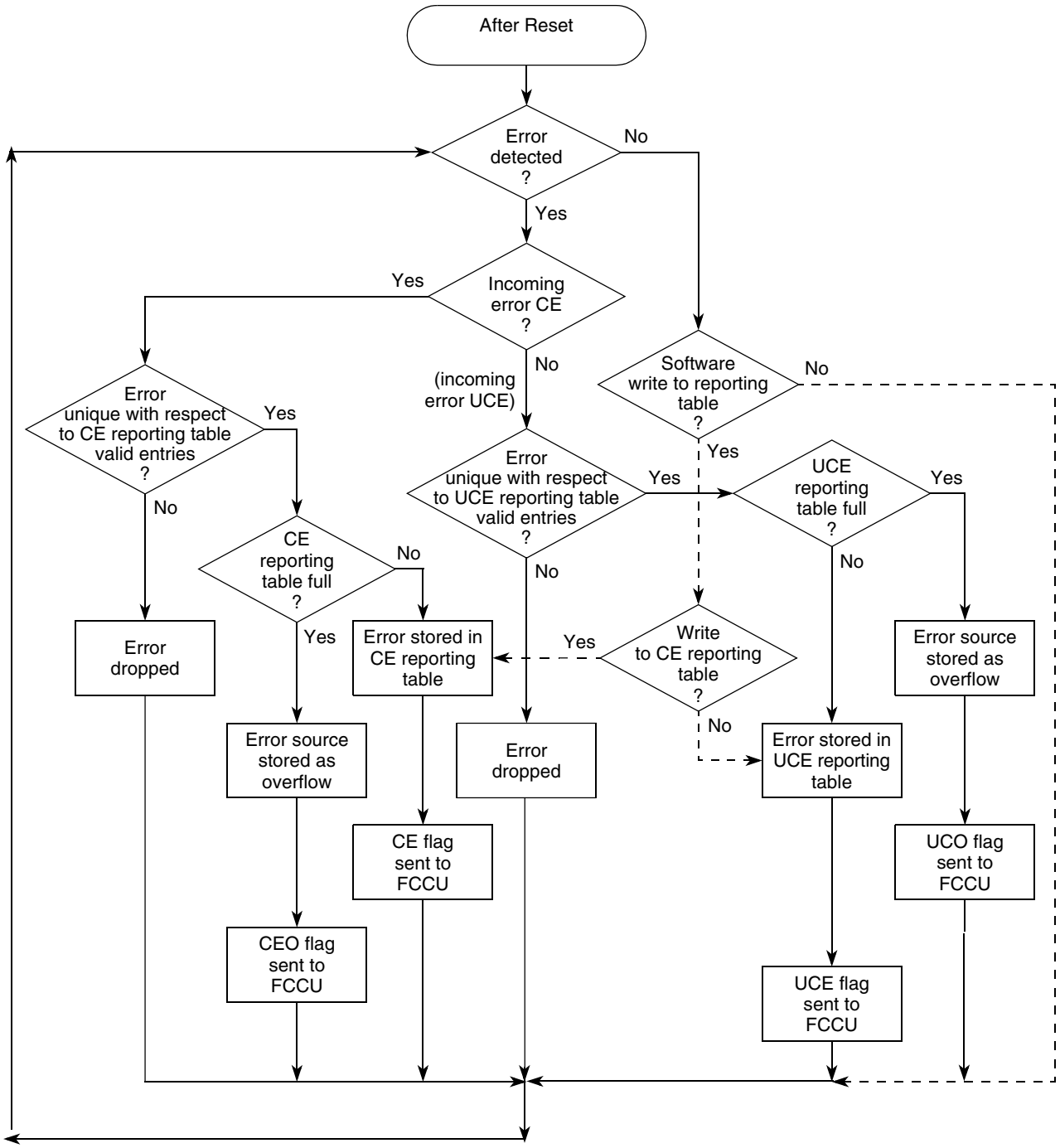


Figure 77-4. MEMU reporting table uniqueness check flow chart

77.6 External signal description

MEMU has no external signals.

77.7 Memory map and register definition

This section describes the registers on this module.

77.7.1 Overview

The memory map comprises 32-bit aligned registers that can be accessed via 8-bit,16-bit, or 32-bit accesses. Write access to reserved address locations will generate a transfer error. Read data from reserved locations will also generate a transfer error and the read data bus will return all 0's.

NOTE

The module does not check for correctness of the programmed values in registers. Software must ensure that the correct values are being written.

NOTE

The memory map in this section shows all the possible registers on this module and the associated offsets (which vary depending on the number of registers actually implemented). The actual availability, number and addresses of registers, and number of reported errors in the reporting table are chip-specific; see the Device Configuration chapter that describes how modules are configured and connected.

MEMU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control register (MEMU_CTRL)	32	R/W	0000_0000h	77.7.2/3256
4	Error flag register (MEMU_ERR_FLAG)	32	w1c	0000_0000h	77.7.3/3256
C	Debug register (MEMU_DEBUG)	32	R/W	0000_0000h	77.7.4/3260
20	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS0)	32	R/W	0000_0000h	77.7.5/3262
24	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR0)	32	R/W	0000_0000h	77.7.6/3263
28	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS1)	32	R/W	0000_0000h	77.7.5/3262
2C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR1)	32	R/W	0000_0000h	77.7.6/3263
30	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS2)	32	R/W	0000_0000h	77.7.5/3262

Table continues on the next page...

MEMU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
34	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR2)	32	R/W	0000_0000h	77.7.6/3263
38	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS3)	32	R/W	0000_0000h	77.7.5/3262
3C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR3)	32	R/W	0000_0000h	77.7.6/3263
40	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS4)	32	R/W	0000_0000h	77.7.5/3262
44	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR4)	32	R/W	0000_0000h	77.7.6/3263
48	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS5)	32	R/W	0000_0000h	77.7.5/3262
4C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR5)	32	R/W	0000_0000h	77.7.6/3263
50	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS6)	32	R/W	0000_0000h	77.7.5/3262
54	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR6)	32	R/W	0000_0000h	77.7.6/3263
58	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS7)	32	R/W	0000_0000h	77.7.5/3262
5C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR7)	32	R/W	0000_0000h	77.7.6/3263
60	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS8)	32	R/W	0000_0000h	77.7.5/3262
64	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR8)	32	R/W	0000_0000h	77.7.6/3263
68	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS9)	32	R/W	0000_0000h	77.7.5/3262
6C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR9)	32	R/W	0000_0000h	77.7.6/3263
70	System RAM uncorrectable error reporting table status register (MEMU_SYS_RAM_UNCERR_STS)	32	R/W	0000_0000h	77.7.7/3263
74	System RAM uncorrectable error reporting table address register (MEMU_SYS_RAM_UNCERR_ADDR)	32	R/W	0000_0000h	77.7.8/3264
78	System RAM concurrent overflow register (MEMU_SYS_RAM_OFLW0)	32	w1c	0000_0000h	77.7.9/3264
620	Peripheral RAM correctable error reporting table status register (MEMU_PERIPH_RAM_CERR_STS0)	32	R/W	0000_0000h	77.7.10/ 3265
624	Peripheral RAM correctable error reporting table address register (MEMU_PERIPH_RAM_CERR_ADDR0)	32	R/W	0000_0000h	77.7.11/ 3266
628	Peripheral RAM correctable error reporting table status register (MEMU_PERIPH_RAM_CERR_STS1)	32	R/W	0000_0000h	77.7.10/ 3265
62C	Peripheral RAM correctable error reporting table address register (MEMU_PERIPH_RAM_CERR_ADDR1)	32	R/W	0000_0000h	77.7.11/ 3266

Table continues on the next page...

MEMU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
630	Peripheral RAM uncorrectable error reporting table status register (MEMU_PERIPH_RAM_UNCERR_STS)	32	R/W	0000_0000h	77.7.12/ 3266
634	Peripheral RAM uncorrectable error reporting table address register (MEMU_PERIPH_RAM_UNCERR_ADDR)	32	R/W	0000_0000h	77.7.13/ 3267
638	Peripheral RAM concurrent overflow register (MEMU_PERIPH_RAM_OFLW0)	32	w1c	0000_0000h	77.7.14/ 3267
C20	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS0)	32	R/W	0000_0000h	77.7.15/ 3268
C24	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR0)	32	R/W	0000_0000h	77.7.16/ 3269
C28	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS1)	32	R/W	0000_0000h	77.7.15/ 3268
C2C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR1)	32	R/W	0000_0000h	77.7.16/ 3269
C30	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS2)	32	R/W	0000_0000h	77.7.15/ 3268
C34	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR2)	32	R/W	0000_0000h	77.7.16/ 3269
C38	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS3)	32	R/W	0000_0000h	77.7.15/ 3268
C3C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR3)	32	R/W	0000_0000h	77.7.16/ 3269
C40	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS4)	32	R/W	0000_0000h	77.7.15/ 3268
C44	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR4)	32	R/W	0000_0000h	77.7.16/ 3269
C48	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS5)	32	R/W	0000_0000h	77.7.15/ 3268
C4C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR5)	32	R/W	0000_0000h	77.7.16/ 3269
C50	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS6)	32	R/W	0000_0000h	77.7.15/ 3268
C54	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR6)	32	R/W	0000_0000h	77.7.16/ 3269
C58	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS7)	32	R/W	0000_0000h	77.7.15/ 3268
C5C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR7)	32	R/W	0000_0000h	77.7.16/ 3269
C60	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS8)	32	R/W	0000_0000h	77.7.15/ 3268
C64	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR8)	32	R/W	0000_0000h	77.7.16/ 3269
C68	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS9)	32	R/W	0000_0000h	77.7.15/ 3268

Table continues on the next page...

MEMU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C6C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR9)	32	R/W	0000_0000h	77.7.16/ 3269
C70	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS10)	32	R/W	0000_0000h	77.7.15/ 3268
C74	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR10)	32	R/W	0000_0000h	77.7.16/ 3269
C78	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS11)	32	R/W	0000_0000h	77.7.15/ 3268
C7C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR11)	32	R/W	0000_0000h	77.7.16/ 3269
C80	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS12)	32	R/W	0000_0000h	77.7.15/ 3268
C84	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR12)	32	R/W	0000_0000h	77.7.16/ 3269
C88	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS13)	32	R/W	0000_0000h	77.7.15/ 3268
C8C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR13)	32	R/W	0000_0000h	77.7.16/ 3269
C90	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS14)	32	R/W	0000_0000h	77.7.15/ 3268
C94	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR14)	32	R/W	0000_0000h	77.7.16/ 3269
C98	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS15)	32	R/W	0000_0000h	77.7.15/ 3268
C9C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR15)	32	R/W	0000_0000h	77.7.16/ 3269
CA0	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS16)	32	R/W	0000_0000h	77.7.15/ 3268
CA4	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR16)	32	R/W	0000_0000h	77.7.16/ 3269
CA8	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS17)	32	R/W	0000_0000h	77.7.15/ 3268
CAC	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR17)	32	R/W	0000_0000h	77.7.16/ 3269
CB0	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS18)	32	R/W	0000_0000h	77.7.15/ 3268
CB4	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR18)	32	R/W	0000_0000h	77.7.16/ 3269
CB8	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS19)	32	R/W	0000_0000h	77.7.15/ 3268
CBC	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR19)	32	R/W	0000_0000h	77.7.16/ 3269
CC0	Flash memory uncorrectable error reporting table status register (MEMU_FLASH_UNCERR_STS)	32	R/W	0000_0000h	77.7.17/ 3269

Table continues on the next page...

MEMU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
CC4	Flash memory uncorrectable error reporting table address register (MEMU_FLASH_UNCERR_ADDR)	32	R/W	0000_0000h	77.7.18/ 3270
CC8	Flash memory concurrent overflow register (MEMU_FLASH_OFLW0)	32	w1c	0000_0000h	77.7.19/ 3270

77.7.2 Control register (MEMU_CTRL)

Access: User read/write

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SWR	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_CTRL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SWR	Software Reset bit. Setting this will clear status flags and overflow register. However, the reporting table registers are not cleared. This bit will auto clear on next clock cycle. 0 No reset. 1 Reset asserted.
17–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

77.7.3 Error flag register (MEMU_ERR_FLAG)

Bits in the ERR_FLAG register indicate:

- A new entry in an error reporting table has been made (indicated by PR_CE, PR_UCE, F_CE, F_UCE, SR_CE, and SR_UCE)

- An overflow condition has been encountered when attempting to make a new entry in an error reporting table (indicated by SR_UCO, SR_CEO, F_UCO,F_CEO, PR_UCO, and PR_CEO)
- An overflow condition has been encountered in the internal error processing buffer of a MEMU reporting block.

Individual flags can be cleared by writing a '1'. This also resets the corresponding error indication to FCCU. Flags will not automatically reset if entries are removed from the reporting table. They have to be explicitly cleared by SW.

Bits SR_UCO, SR_CEO, F_UCO, F_CEO, PR_UCO, and PR_CEO are asserted when an overflow in the uncorrectable error reporting table is detected.

Access: User read/write

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0											PR_CE	PR_UCE	PR_CEO	PR_UCO	PR_EBO
W												w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			F_CE	F_UCE	F_CEO	F_UCO	F_EBO	0			SR_CE	SR_UCE	SR_CEO	SR_UCO	SR_EBO
W				w1c	w1c	w1c	w1c	w1c				w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_ERR_FLAG field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 PR_CE	Peripheral RAM ECC Correctable Error detect flag. Indicates that a new and unique Peripheral RAM ECC correctable error is detected. Asserted when a new entry is inserted into the correctable error reporting table (either by CPU or module).

Table continues on the next page...

MEMU_ERR_FLAG field descriptions (continued)

Field	Description
	0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
12 PR_UCE	Peripheral RAM ECC Uncorrectable Error Detect flag. Asserted when a new Peripheral RAM ECC uncorrectable error was detected or a new entry is inserted into the uncorrectable error reporting table (either by CPU or module). 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
13 PR_CEO	Peripheral RAM ECC Correctable error Overflow flag. Asserted when the Peripheral RAM ECC correctable error reporting table is full and a new entry is attempted to be made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.
14 PR_UCO	Peripheral RAM ECC Uncorrectable error Overflow flag. Asserted when the Peripheral RAM ECC uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
15 PR_EBO	Peripheral RAM ECC Error buffer Overflow flag. Asserted when the internal error processing buffer of Peripheral RAM ECC MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for processing by MEMU. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to 1 if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 F_CE	Flash ECC Correctable Error Detect flag. Indicates that a new and unique Flash ECC correctable error was detected. Asserted when a new entry to the correctable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
20 F_UCE	Flash ECC Uncorrectable Error Detect flag. Asserted when: • a new Flash ECC uncorrectable error was detected. • a new entry to the uncorrectable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
21 F_CEO	Flash ECC Correctable Error Overflow flag. Asserted when the Flash ECC correctable error reporting table is full and a new entry is made to this table.

Table continues on the next page...

MEMU_ERR_FLAG field descriptions (continued)

Field	Description
	0 No Overflow. 1 Overflow in the correctable error reporting table detected.
22 F_UCO	Flash ECC Uncorrectable Error Overflow flag. Asserted when the Flash ECC uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
23 F_EBO	Flash ECC Error buffer Overflow. Flag Asserted when the internal error processing buffer of Flash ECC MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for MEMU to process them all. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to 1 if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 SR_CE	System RAM ECC Correctable Error detect flag. Indicates that a new and unique System RAM ECCcorrectable error is detected. Asserted when a new entry to the correctable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
28 SR_UCE	System RAM ECC Uncorrectable Error Detect flag. Asserted when: • A new System RAM ECC uncorrectable error is detected • A new entry to the uncorrectable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
29 SR_CEO	System RAM ECC Correctable error Overflow flag. Asserted when the System RAM ECC correctable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.
30 SR_UCO	System RAM ECC Uncorrectable error Overflow flag. Asserted when the System RAM ECC uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
31 SR_EBO	System RAM ECC Error buffer Overflow.

Table continues on the next page...

MEMU_ERR_FLAG field descriptions (continued)

Field	Description
	Flag asserted when the internal error processing buffer of System RAM ECC MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for processing by MEMU. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to 1 if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.

77.7.4 Debug register (MEMU_DEBUG)

This register is used to check the connections between MEMU & FCCU.

The error output of the MEMU towards FCCU shall stay set if forced until reset by SW via the MEMU in the normal way.

Access: User read/write

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0											FR_PR_CE	FR_PR_UCE	FR_PR_CEO	FR_PR_UCO	FR_PR_EBO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			FR_F_CE	FR_F_UCE	FR_F_CEO	FR_F_UCO	FR_F_EBO	0			FR_SR_CE	FR_SR_UCE	FR_SR_CEO	FR_SR_UCO	FR_SR_EBO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_DEBUG field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 FR_PR_CE	Force Peripheral RAM Correctable Error detect flag. 0 Forcing is disabled. 1 Forces PR_CE flag going towards FCCU to 1.

Table continues on the next page...

MEMU_DEBUG field descriptions (continued)

Field	Description
12 FR_PR_UCE	Force Peripheral RAM Uncorrectable Error detect flag. 0 Forcing is disabled. 1 Forces PR_UCE flag going towards FCCU to 1.
13 FR_PR_CEO	Force Peripheral RAM Correctable Error overflow flag 0 Forcing is disabled. 1 Forces PR_CEO flag going towards FCCU to 1.
14 FR_PR_UCO	Forces Peripheral RAM Uncorrectable Error overflow flag. 0 Forcing is disabled. 1 Forces PR_UCO flag going towards FCCU to 1.
15 FR_PR_EBO	Forces Peripheral RAM Error Buffer Overflow Flag. 0 Forcing is disabled. 1 Forces PR_EBO flag going towards FCCU to 1.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 FR_F_CE	Forces Flash Correctable Error detect flag 0 Forcing is disabled. 1 Forces F_CE flag going towards FCCU to 1.
20 FR_F_UCE	Forces Flash Uncorrectable Error detect flag 0 Forcing is disabled. 1 Forces F_UCE flag going towards FCCU to 1.
21 FR_F_CEO	Forces Flash Correctable Error overflow flag 0 Forcing is disabled. 1 Forces F_CEO flag going towards FCCU to 1.
22 FR_F_UCO	Forces Flash Uncorrectable Error overflow flag 0 Forcing is disabled. 1 Forces F_UCO flag going towards FCCU to 1.
23 FR_F_EBO	Forces Flash Error buffer Overflow flag 0 Forcing is disabled. 1 Forces F_EBO flag going towards FCCU to 1.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 FR_SR_CE	Forces System RAM Correctable Error detect flag 0 Forcing is disabled. 1 Forces SR_CE flag going towards FCCU to 1.
28 FR_SR_UCE	Forces System RAM Uncorrectable Error detect flag 0 Forcing is disabled. 1 Forces SR_UCE flag going towards FCCU to 1.

Table continues on the next page...

MEMU_DEBUG field descriptions (continued)

Field	Description
29 FR_SR_CEO	Forces System RAM Correctable Error overflow flag 0 Forcing is disabled. 1 Forces SR_CEO flag going towards FCCU to 1.
30 FR_SR_UCO	Forces System RAM Uncorrectable Error overflow flag 0 Forcing is disabled. 1 Forces SR_UCO flag going towards FCCU to 1.
31 FR_SR_EBO	Forces System RAM Error buffer Overflow Flag 0 Forcing is disabled. 1 Forces SR_EBO flag going towards FCCU to 1.

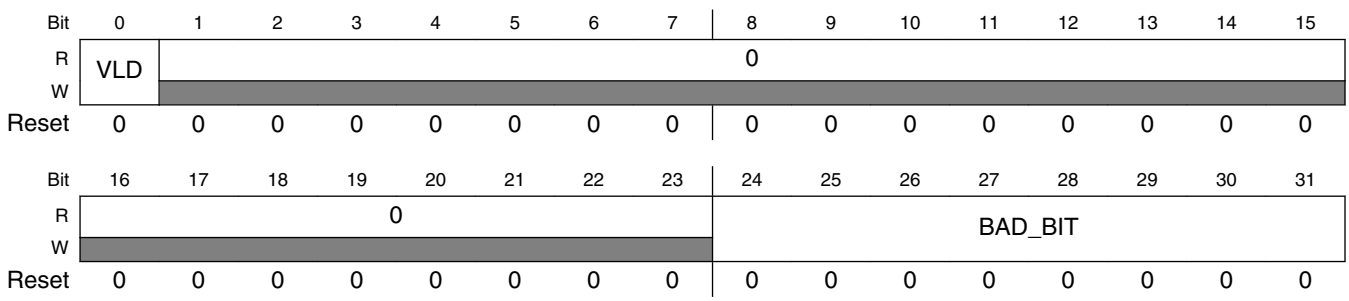
77.7.5 System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 20h offset + (8d × i), where i=0d to 9d



MEMU_SYS_RAM_CERR_STS*n* field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 BAD_BIT	Bad bit field. Indicates the bit position in RAM where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFF.

Table continues on the next page...

MEMU_SYS_RAM_CERR_STS*n* field descriptions (continued)

Field	Description
	Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

77.7.6 System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR*n*)

Access: User read/write

Address: 0h base + 24h offset + (8d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_SYS_RAM_CERR_ADDR*n* field descriptions

Field	Description
0–31 ERR_ADD	Error address field Indicates the address on which the error was detected.

77.7.7 System RAM uncorrectable error reporting table status register (MEMU_SYS_RAM_UNCERR_STS)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 70h offset = 70h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_SYS_RAM_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

77.7.8 System RAM uncorrectable error reporting table address register (MEMU_SYS_RAM_UNCERR_ADDR)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 74h offset = 74h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_SYS_RAM_UNCERR_ADDR field descriptions

Field	Description
0–31 ERR_ADD	Error address field. Indicates the address on which the error was detected. This field must be read-only when its corresponding valid bit (SYS_RAM_UNCERR_STS[VLD]) = 0.

77.7.9 System RAM concurrent overflow register (MEMU_SYS_RAM_OFLW*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 78h offset + (4d × i), where i=0d to 0d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_SYS_RAM_OFLW*n* field descriptions

Field	Description
0–31 OFLW	Overflow Bit. Each bit corresponds to an error source and is asserted when any of the conditions mentioned in Handling overflows (Multiple error reporting) occurs.

77.7.10 Peripheral RAM correctable error reporting table status register (MEMU_PERIPH_RAM_CERR_STS*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 620h offset + (8d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								BAD_BIT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

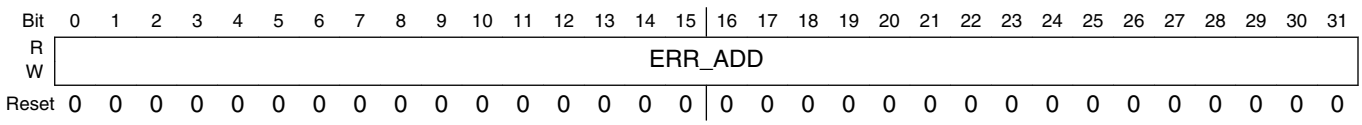
MEMU_PERIPH_RAM_CERR_STS*n* field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 BAD_BIT	Bad bit field. Indicates the bit position in RAM where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFF. Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

77.7.11 Peripheral RAM correctable error reporting table address register (MEMU_PERIPH_RAM_CERR_ADDR*n*)

Access: User read/write

Address: 0h base + 624h offset + (8d × i), where i=0d to 1d



MEMU_PERIPH_RAM_CERR_ADDR*n* field descriptions

Field	Description
0–31 ERR_ADD	Error address field Indicates the address on which the error was detected.

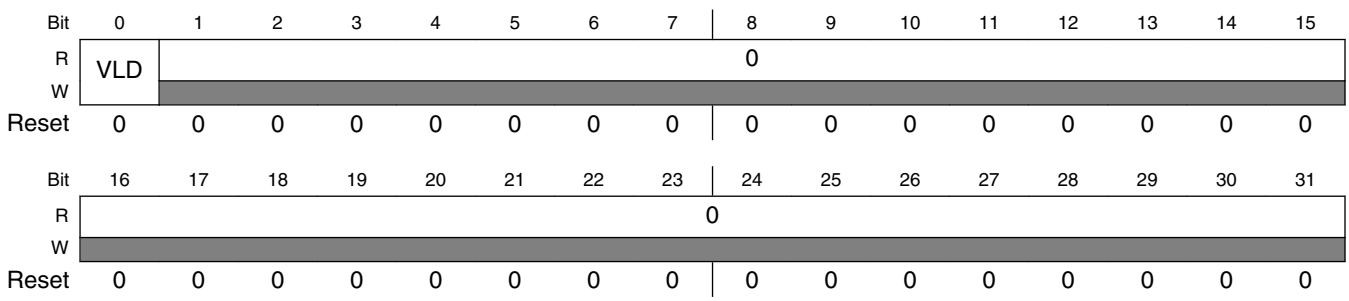
77.7.12 Peripheral RAM uncorrectable error reporting table status register (MEMU_PERIPH_RAM_UNCERR_STS)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 630h offset = 630h



MEMU_PERIPH_RAM_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted.

Table continues on the next page...

MEMU_PERIPH_RAM_UNCERR_STS field descriptions (continued)

Field	Description
	0 Entry in table is invalid. 1 Entry in table is valid.
1–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

77.7.13 Peripheral RAM uncorrectable error reporting table address register (MEMU_PERIPH_RAM_UNCERR_ADDR)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 634h offset = 634h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_PERIPH_RAM_UNCERR_ADDR field descriptions

Field	Description
0–31 ERR_ADD	Error address field. Indicates the address on which the error was detected. This field must be read-only when its corresponding valid bit (PERIPH_RAM_UNCERR_STS[VLD]) = 0.

77.7.14 Peripheral RAM concurrent overflow register (MEMU_PERIPH_RAM_OFLW*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 638h offset + (4d × i), where i=0d to 0d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_PERIPH_RAM_OFLW*n* field descriptions

Field	Description
0–31 OFLW	Overflow Bit. Each bit corresponds to an error source and is asserted when any of the conditions mentioned in Handling overflows (Multiple error reporting) occurs.

77.7.15 Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + C20h offset + (8d × i), where i=0d to 19d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD								0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								BAD_BIT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_FLASH_CERR_STS*n* field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid . Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 BAD_BIT	Bad bit field. Indicates the bit position in flash memory where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFF. Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

77.7.16 Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR*n*)

Access: User read/write

Address: 0h base + C24h offset + (8d × i), where i=0d to 19d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_FLASH_CERR_ADDR*n* field descriptions

Field	Description
0–31 ERR_ADD	Error address field Indicates the address on which the error was detected.

77.7.17 Flash memory uncorrectable error reporting table status register (MEMU_FLASH_UNCERR_STS)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + CC0h offset = CC0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	VLD	0														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_FLASH_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted.

Table continues on the next page...

MEMU_FLASH_UNCERR_STS field descriptions (continued)

Field	Description
	0 Entry in table is invalid. 1 Entry in table is valid.
1–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

77.7.18 Flash memory uncorrectable error reporting table address register (MEMU_FLASH_UNCERR_ADDR)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + CC4h offset = CC4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_FLASH_UNCERR_ADDR field descriptions

Field	Description
0–31 ERR_ADD	Error address field. Indicates the address on which the error was detected. This field must be read-only when its corresponding valid bit (FLASH_UNCERR_STS[VLD]) = 0.

77.7.19 Flash memory concurrent overflow register (MEMU_FLASH_OFLW*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + CC8h offset + (4d × i), where i=0d to 0d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_FLASH_OFLW*n* field descriptions

Field	Description
0–31 OFLW	Overflow Bit. Each bit corresponds to an error source and is asserted when any of the conditions mentioned in Handling overflows (Multiple error reporting) occurs.

77.8 Functional description

In this section, the term “bad bit” and “syndrome” are equivalent, and are used to indicate the position of the bit in error.

77.8.1 Initializing MEMU

MEMU is always enabled and can be configured by user software for known faults which the system/application has collected over a period of time. The MEMU logic will not update the reporting table corresponding to the programmed values which have the VLD bit SET.

The user software can write into the reporting table and must make VLD bit ='1' for the programmed entry to be valid. Any write by the CPU that asserts the VLD bit will cause a corresponding flag to FCCU.

The software can program the table with known error addresses by setting the valid bit and storing the corresponding error address. The sequence of programming to prevent a collision of reporting table entries written by the HW and the SW is as follows:

1. Check that the VLD bit is not set,
2. Write an invalid address/bad bit information,
3. Set VLD bit,
4. Check that the address still has an invalid value,
5. Write address/bad bit information to desired address.

If the FCCU is programmed to cause an IRQ on new MEMU table entries, steps 3 to 5 should be executed with interrupts disabled to prevent any fault handling SW to read the invalid address written in step 2). Note that in the case the software is in between the 3rd and the 5th step, and the MEMU writes a fault with the same address that Software wants to program, both the faults will be stored in the reporting table (i.e. duplicate entries).

77.8.2 Reading the reporting table

The software at any time can read the reporting table via the software programming interface. It should read the entries with the valid bit set and take necessary action. To invalidate any entry, the valid bit must be cleared. Reading an entry which does not have the VLD bit set will return invalid data.

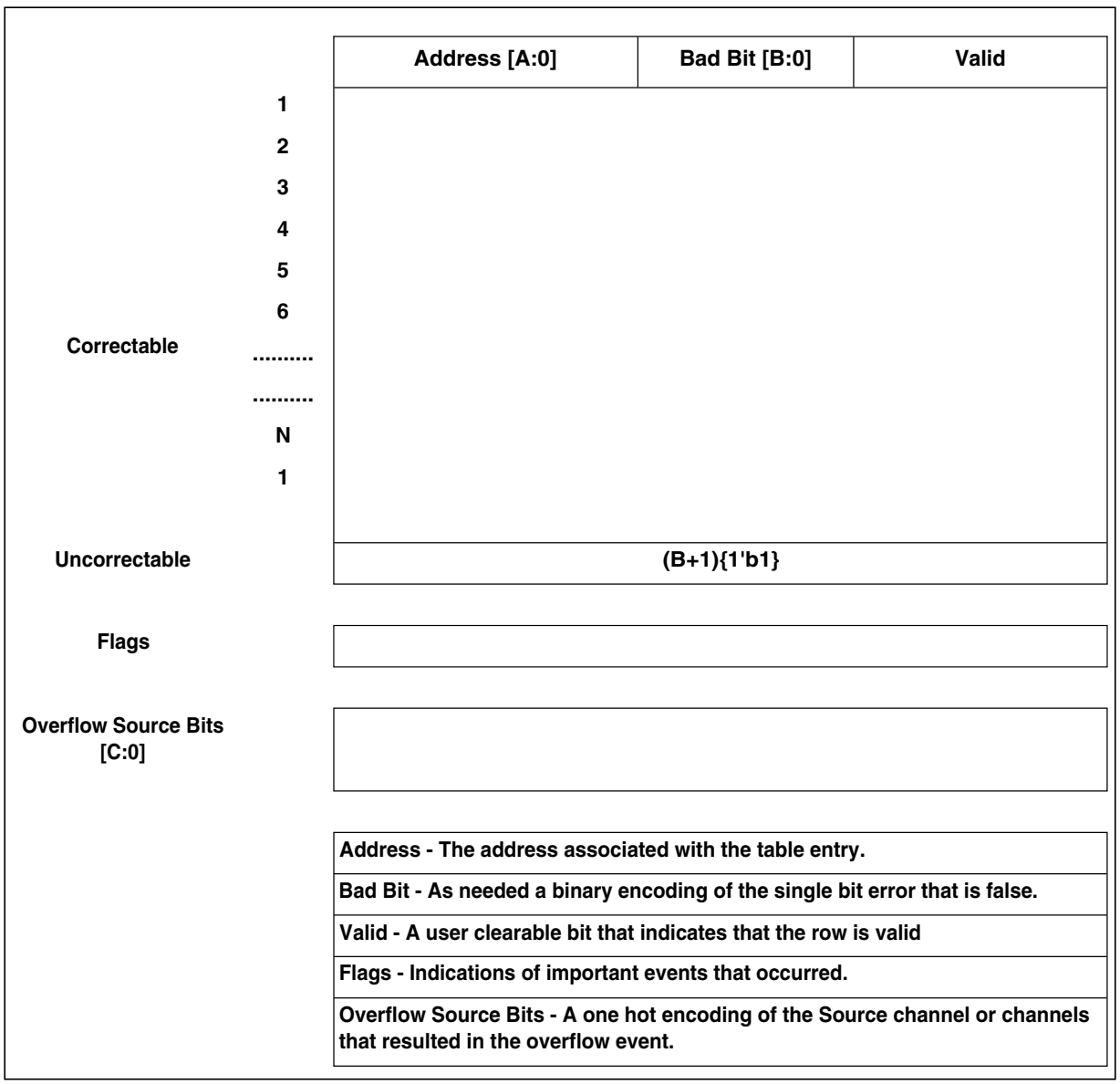


Figure 77-90. Generic representation of the reporting table of one MEMU reporting block

77.8.3 Handling overflows (Multiple error reporting)

When a bit in the overflow registers (either error buffer or reporting table though the overflow registers are same) is asserted it can indicate the occurrence of one of the following conditions:

1. Fault buffer overflow–More than two memories reporting a fault at the same instant or the input ASYNC FIFOs reach the FULL level (overflow)
    - More than 2 faults detected at same time. This would lead to overflow in the fault buffer. MEMU can process only one fault at a time while storing the other in the fault buffer. So this would be indicated as an overflow. The uniqueness check in the input buffer does not "guarantee" that an ECC-supervised memory with only one fault will not be marked as overflow source. If that one fault occurs together with two (or more) additional faults at the same cycle it can get flagged as an overflow.
2. Correctable/Uncorrectable Fault Overflow–Overflow in correctable and uncorrectable reporting table occurs only when a unique entry is to be stored but the tables are already full (all entries have valid bit set).
3. The Overflows will be stored in the bit-wise order i.e., if the fault bit 31 reports a fault, the bit 31 of the overflow register will be set to '1'.

Chapter 78
Fault Collection and Control Unit (FCCU)

78.1 FCCU chip-specific module information

This section describes the implementation of this module in the device.

78.1.1 FCCU fault recovery after NMI reaction

When an NMI (Non-Maskable Interrupt) is configured as the internal FCCU reaction for a specific NCF (fault state), the application software should not clear the core's MCSR[NMI] bit field during the execution of the NMI interrupt service routine, unless the FCCU fault source which generated the respective NMI reaction is cleared.

For details on MCSR[NMI] bit field clearing, see Machine Check Syndrome Register (MCSR). For details on FCCU Fault recovery see Fault recovery.

78.1.2 FCCU chip-specific register reset values

The following table shows FCCU registers whose reset values are specific to this chip.

Table 78-1. FCCU chip-specific register reset values

Register	Reset value
FCCU_CFG	0x00000008
FCCU_NCF_CFG0	0xFFFFFFFF
FCCU_NCF_CFG1	0xFFFFFFFF
FCCU_NCF_CFG2	0xFFFFFFFF
FCCU_NCFS_CFG0	0x00002000
FCCU_NCFS_CFG1	0x00000080
FCCU_NCFS_CFG2	0x00000000
FCCU_NCFS_CFG3	0x00000000

Table continues on the next page...

Table 78-1. FCCU chip-specific register reset values (continued)

Register	Reset value
FCCU_NCFS_CFG4	0x00000000
FCCU_NCFS_CFG5	0x00000000
FCCU_NCF_E0	0x00080040
FCCU_NCF_E1	0x00000000
FCCU_NCF_E2	0x00000000
FCCU_NCF_TOE0	0x00000000
FCCU_NCF_TOE1	0x00000000
FCCU_NCF_TOE2	0x00000000
FCCU_NCF_TO	0x0000_FFFF
FCCU_CFG_TO	0x0000_0006

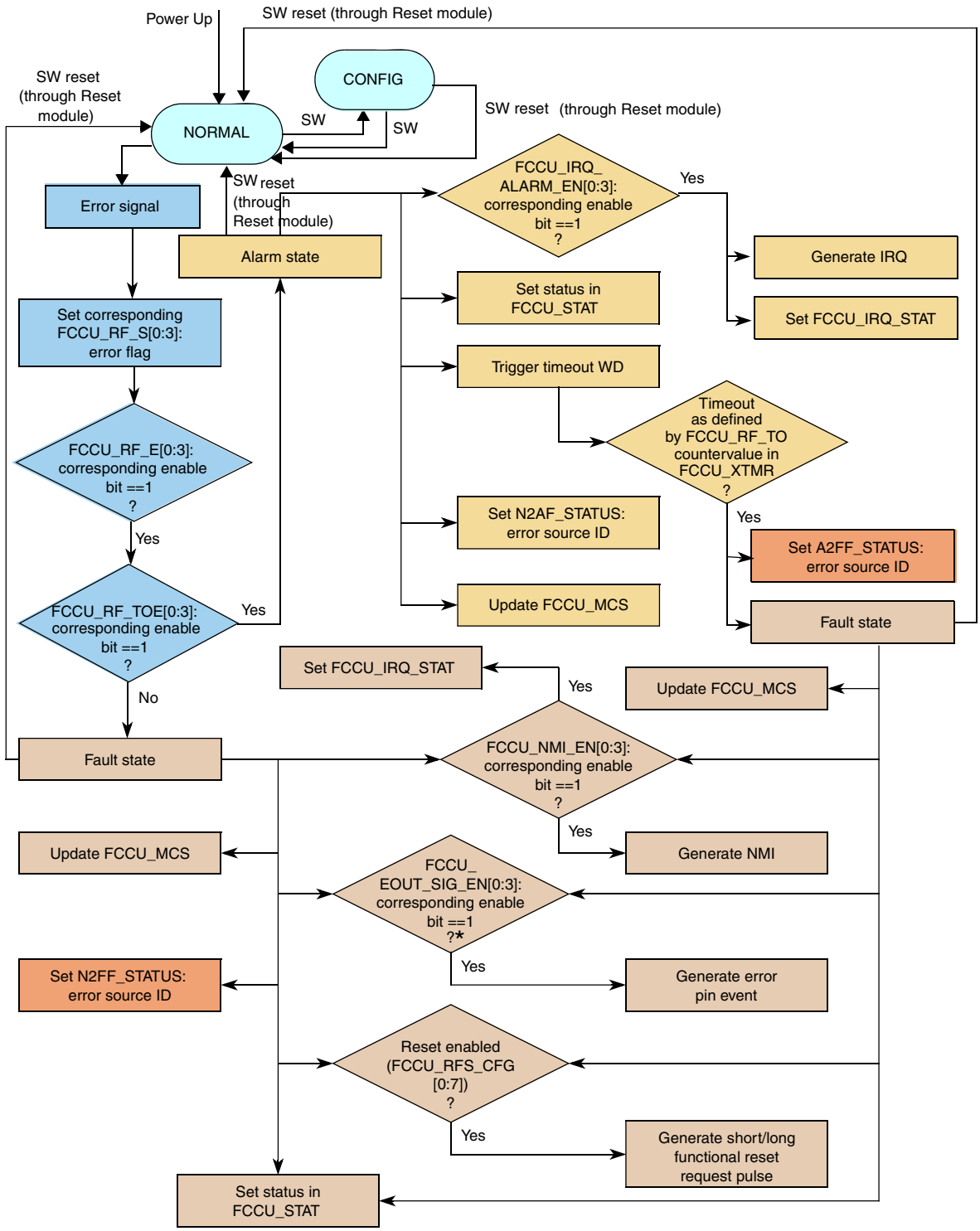
78.1.3 Available FCCU output signals

The following table shows the available FCCU output signals implemented on this chip.

Table 78-2. FCCU output signals

FCCU output signal name	Chip-top signal name
EOUT[0]	ERROR0
EOUT[1]	ERROR1

78.1.4 Error signal flow diagram



* This condition must be satisfied only when FCCU is configured for Bistable fault-output mode (FCCU_CFG[FOM]).

Figure 78-1. Error signal flow

78.1.5 FCCU failure inputs from other modules

The following table shows the different failure input signals of the FCCU, how they can be tested, and their default configuration after reset.

NOTE

All the faults shown in the following table require a long functional reset.

NOTE

FCCU Fault Channels 51, 52, 69 through 84, 92, and 93 may be asserted during the execution of MBIST and LBIST. Before running MBIST or LBIST, these channels should be configured to NO REACTION. If any of these channels become asserted during MBIST or LBIST testing, the fault should simply be cleared and no further action should be taken.

Table 78-3. FCCU Fault Inputs

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
0	No Reaction	Temp sensor	Using PMC_TS_FIR register to set the fault at source + Via FCCU Fake bit	Temperature Sensor Event Triggered by PMC_ESR_TD[TEMP1_2, TEMP1_0, TEMP0_2, TEMP0_0] and PMC_TS_FIR[TSNS_FAULT] . For details, see the descriptions of the PMC_ESR_TD and PMC_TS_FIR registers.
1	No Reaction	PMC	Using PMC FIR register to set the fault at source + Via FCCU Fake bit	LVD event to FCCU Triggerred by PMC_LVD_HVD_EVENT_STATUS[LVD_PMC, LVD_SAR_ADC, LVD_IO, LVD_FLASH, LVD_CORE_HOT, LVD_CORE_COLD] and PMC_FIR[LVD_FAULT]. For details, see the descriptions of the PMC_LVD_HVD_EVENT_STATUS and PMC_FIR registers.
2	No Reaction	PMC	Using PMC FIR register to set the fault at source + Via FCCU Fake bit	HVD Event to FCCU Triggered by PMC_LVD_HVD_EVENT_STATUS[HVD_FLASH, HVD_HV, HVD_CORE]. For details, see the descriptions of the PMC_LVD_HVD_EVENT_STATUS register.

Table continues on the next page...

Table 78-3. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
3	No Reaction	PMC	Using PMC FIR register to set the fault at source + Via FCCU Fake bit	PMC Self Test Failure Event PMC hard/soft self-test failed: LVD/HVD **NOTE:** When this occurs, single LVD or HVD self-tests should be executed to identify which detection mechanism failed in the SOC: LVD or HVD. For details, see the description of the PMC_SELF_TEST_UM_VD_REG register as well as the flag bits associated to NCF3 (PMC_SELF_TEST_UM_VD_REG [ST_RESULT]). Self test bit definition indicates all valid values for PMC_SELF_TEST_UM_VD_REG[VD_ST_CTRL] while executing single self-tests. LVD-HVD self test mechanism provides details about the available PMC self-tests
4	No Reaction	SSCM	write wrong data to address + Via FCCU Fake bit to set the fault at error path	See SSCM_STATUS[CERS] field description.
5	No Reaction	SSCM	write wrong data to address + Via FCCU Fake bit to set the fault at error path	See SSCM_STATUS[CER] field description.
6	long reset	STCU	SW check of BIST MISR + Via FCCU Fake bit to set the fault at source	Critical Fault of STCU. Set when STCU2_ERR_STAT[UFSF] is set.
7	No Reaction	STCU	SW check of BIST MISR + Via FCCU Fake bit to set fault at source	Non critical Fault of STCU. Set when STCU2_ERR_STAT[RFSF] is set.
8	No Reaction	STCU	STCU activation during user application run Via FCCU Fake bit to set the fault at source	Critical Fault in case LBIST or MBIST control signals go to wrong condition during User application. This fault is set when the self-test was triggered without STCU2_RUN[RUN] bit programming.
9	No Reaction	Checker Shell	Via FCCU Fake bit	out of lockstep fault detected for miscellaneous safety core output
10	No Reaction	Checker Shell	Via FCCU Fake bit to set the fault at error source	out of lockstep fault detected for safety Core dtcm array interface signal
11	No Reaction	Checker Shell	Via FCCU Fake bit to set the fault at error source	out of lockstep fault detected for safety Core I Cache array interface signals
12	Reserved	Reserved		Reserved
13	No Reaction	Checker Shell	Via FCCU Fake bit to set the fault at error source	out of lockstep fault detected for safety Core Amba interface signals
14	No Reaction	DMA Checker Shell	Via FCCU Fake bit to set the fault at error source	out of lockstep fault detected for DMA

Table continues on the next page...

Table 78-3. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
15	No Reaction	DMC Checker	Via FCCU Fake bit to set the fault at error source	out of lockstep fault detected for core0 TCM DMC
16	RESERVED	RESERVED		RESERVED
17	No Reaction	SWT3	Emulate fault using functionality + Via FCCU Fake bit to set the fault at error path	Software watchdog timer reset request
18	RESERVED	RESERVED		RESERVED
19	Long Reset	SWT1	Emulate fault using functionality (SW test by not triggering SWT during startup) + Via FCCU Fake bit to set the fault at error path	Software watchdog timer reset request
20	No Reaction	SWT0	Emulate fault using functionality (SW test by not triggering SWT during startup) + Via FCCU Fake bit to set the fault at error path	Software watchdog timer interrupt
21	No Reaction	SWT0	Emulate fault using functionality (SW test by not triggering SWT during startup) +Via FCCU Fake bit to set the fault at error path	Software watchdog timer reset request
22	No Reaction	MEMU	Injection of RAM ECC error via ECC bit access and later reading + Via FCCU Fake bit to set the fault at error path+ LBIST	System RAM Correctable Error detected
23	No Reaction	MEMU	Injection of RAM ECC error via ECC bit access and later reading + Via FCCU Fake bit to set the fault at error path + LBIST	System RAM Un-Correctable Error detected
24	No Reaction	MEMU	Via FCCU Fake bit to set the fault at error path + LBIST	System RAM error Overflow
25	No Reaction	MEMU + eTPU Memory Error Correction Interrupt	Via FCCU Fake bit to set the fault at error path + LBIST	Peripheral RAM Correctable error detected. Set eTPU_MESR_MECR[MECIE] to enable this error.
26	No Reaction	MEMU + eTPU Global Exception	Via FCCU Fake bit to set the fault at error path + LBIST	Peripheral RAM UN-Correctable error detected. Set eTPU_MESR_MECR[MECIE] to enable this error.
27	No Reaction	MEMU	Via FCCU Fake bit to set the fault at error path + LBIST	Periph RAM error Overflow

Table continues on the next page...

Table 78-3. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
28	No Reaction	MEMU	Reading preprogrammed correctable ECC error from UTEST flash memory + Via FCCU Fake bit to set the fault at error path + LBSIT	Flash Correctable Error detected NCF correlated to C55FMC_MCR[SBC] bit **NOTE:** From the MEMU perspective this is indicated by MEMU_FLASH_CERR_STSn. For details, see the description of the C55FMC_MCR and MEMU_FLASH_CERR_STSn registers.
29	No Reaction	MEMU	Reading preprogrammed uncorrectable ECC error from UTEST flash memory + Via FCCU Fake bit to set the fault at error path + LBIST	Flash Un-Correctable Error detected NCF correlated to C55FMC_MCR[EER] bit **NOTE:** From the MEMU perspective this is indicated by MEMU_FLASH_UNCERR_STSn. For details, see the description of the C55FMC_MCR and MEMU_FLASH_UNCERR_STS registers.
30	No Reaction	MEMU	Via FCCU Fake bit to set the fault at error path + LBIST	Flash Error Overflow For details on conditions which trigger this NCF, see section Handling overflows (Multiple error reporting). **NOTE:** From the MEMU perspective this is indicated by MEMU_FLASH_OFLWn. For details, see the description of the MEMU_FLASH_OFLWn register.
31	RESERVED	RESERVED		RESERVED
32	No Reaction	PLL0	Forced loss by PLL reconfiguration + Via FCCU Fake bit to set the fault at error path	Loss of Lock detected for NON-FM PLL **NOTE:** This source should be configured to generate a reset instead of an interrupt if the source is used as the system clock.
33	No Reaction	PLL1	Forced loss by PLL reconfiguration + Via FCCU Fake bit to set the fault at error path	Loss of Lock detected for FM PLL **NOTE:** This source should be configured to generate a reset instead of an interrupt if the source is used as the system clock.
34	No Reaction	CMU_PLL	Test by misconfiguration + Via FCCU Fake bit to set the fault at error path	OLR Event from crystal clock monitor circuit of cmu_pll **NOTE:** This source should be configured to generate a reset instead of an interrupt if the source is used as the system clock.

Table continues on the next page...

Table 78-3. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
35	No Reaction	CMU_PLL	Test by misconfiguration + Via FCCU Fake bit to set the fault at error path	FLL/FHH event from PLL Clock Monitor circuit. This fault is set when CMU_0 detects an FHH or FLL event **NOTE:** To identify which CMU module raised this NCF, read the CMU_ISR[FHHI/FLLI] bit fields from all relevant CMU instances. For details, see the description of the CMU_ISR register. For available CMU's, see sections Clock input sources and CMU registers and field availability .
36	No Reaction	System clock CMU	Test by misconfiguration + Via FCCU Fake bit to set the fault at error path	FLL/FHH event from System clock monitor circuit. This fault is set when one of CMU_1 to CMU_3 detects an FHH or FLL event. **NOTE:** To identify which CMU module raised this NCF, read the CMU_ISR[FHHI/FLLI] bit fields from all relevant CMU instances. For details, see the description of the CMU_ISR register. For available CMU's, see sections Clock input sources and CMU registers and field availability .
37	No Reaction	Other CMU	Emulate fault using functionality + Via FCCU Fake bit to set the fault at error path	FLL/FHH event from other clock monitor circuit. This fault is set when one of CMU_4 to CMU_12 detects an FHH or FLL event. **NOTE:** To identify which CMU module raised this NCF, read the CMU_ISR[FHHI/FLLI] bit fields from all relevant CMU instances. For details, see the description of the CMU_ISR register. For available CMU's, see sections Clock input sources and CMU registers and field availability .
38	No Reaction	Periph AXBS	Using XBIC_EIR register to set the fault at source+Via FCCU Fake bit to set the fault at error path	Peripheral Crossbar alarm to FCCU
39	No Reaction	Comp AXBS	Using XBIC_EIR register to set the fault at source+Via FCCU Fake bit to set the fault at error path	Computational Crossbar alarm to FCCU
40	No Reaction	Flash	Via FCCU Fake bit to set the fault at error source + LBIST	alarm indicating the flash detected an error in the ECC correction logic through an EDC

Table continues on the next page...

Table 78-3. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
41	No Reaction	Flash Controller	Via FCCU Fake bit to set the fault at error path + LBIST	alarm indicating the flash controller detected an error in the address ECC manipulation logic through an EDC
42	No Reaction	Flash	Via FCCU Fake bit to set the fault at error path + LBIST	alarm indicating the Flash Array Controller detected an Encode Address Error, such as multiple address lines selected or no address lines selected **NOTE:** Address Encode Errors detected within Flash Array [FCCU channel 42] will also assert the P-Flash [FCCU channel 43]
43	No Reaction	Flash Controller	Via FCCU Fake bit to set the fault at error path + LBIST	alarm indicating the flash controller detected a transaction monitor mismatch when compared to the flash safety feedback outputs
44	No Reaction	Flash Controller	Via FCCU Fake bit to set the fault at error path + LBIST	alarm indicating the flash controller detected a transaction monitor mismatch when compared to the on-chip calibration RAM safety feedback outputs
45	No Reaction	Flash Controller	Via FCCU Fake bit to set the fault at error path + LBIST	alarm indicating the flash controller detected a transaction monitor mismatch in the pseudo-replicated calibration evaluation hardware
46	RESERVED	RESERVED		RESERVED
47	No Reaction	Platform Gasket	Via FCCU Fake bit to set the fault at error path	2:1 bridging gasket transaction monitor alarm
48	No Reaction	Platform Gasket	Via FCCU Fake bit to set the fault at error path	1:2 bridging gasket transaction monitor alarm
49	No Reaction	SMPU	Via FCCU Fake bit to set the fault at error path + LBIST	Computational MPU alarm to the fault collection unit An access error has happened due to restrictions specified by the MPU region descriptors. This is in addition to the MPU responding to the requesting master with an error response.
50	No Reaction	SMPU	Via FCCU Fake bit to set the fault at error path + LBIST	Peripheral MPU alarm to the fault collection unit An access error has happened due to restrictions specified by the MPU region descriptors. This is in addition to the MPU responding to the requesting master with an error response.

Table continues on the next page...

Table 78-3. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
51	No Reaction	Decorated memory Controller	No fake faults supported	AIPS decorated storage transaction monitor error: this monitor basically ensure that the Decorated Memory Controller is not modifying regular AHB transactions when a non-decorated storage instruction is performed. If a non-decorated storage instruction is executed, the DMC should pass the transaction through unaltered.
52	No Reaction	Decorated memory Controller	No fake faults supported	AIPS decorated storage transaction monitor error: this monitor basically ensure that the Decorated Memory Controller is not modifying regular AHB transactions when a non-decorated storage instruction is performed. If a non-decorated storage instruction is executed, the DMC should pass the transaction through unaltered.
53	No Reaction	PRAM Controller	Via FCCU Fake bit to set the fault at error path	PRAM Controller/AHB bus transaction monitor error output
54	RESERVED	RESERVED		RESERVED
55	No Reaction	Core	Willfully cause an appropriate exception (for example, access a non-existing address) + Via FCCU Fake bit to set the fault at error path	machine check has occurred
56	No Reaction	Core	Willfully cause an appropriate exception (for example, access a non-existing address) + Via FCCU Fake bit to set the fault at error path	machine check has occurred
57	No Reaction	SOC	Disabling Checker core while safety core is on+ Via FCCU Fake bit to set the fault at error path	Lock Step Disabled
58	No Reaction	RGM	RGM “safe mode” request + Via FCCU Fake bit to set the fault at error path + LBIST	Safe Mode request from RGM
59	RESERVED	RESERVED		-
60	No Reaction	IOMUX	External activation of EIN. External entity has to pull the error input low. + Via FCCU Fake bit to set the fault at error path	Error Input
61	RESERVED	RESERVED		RESERVED

Table continues on the next page...

Table 78-3. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
62	No Reaction	DCI	Via FCCU Fake bit to set the fault at error path + LBIST	Debug Fault Condition to MCU • If any JTAG_aux_tms is high from DCI • If Jtag_rst_b is low from DCI • If JTAG mode is high from DCI • If nex_rst_b is high from DCI
63	No Reaction	SAR ADC0/1/2/3	VIA SAR-ADC STCR2[SERR] bit to set the fault at error source + Via FCCU Fake bit to set the fault at error path	Non Critical fault from ALL SARs
64	No Reaction	SAR ADC0/1/2/3	VIA SAR-ADC STCR2[SERR] bit to set the fault at error source + Via FCCU Fake bit to set the fault at error path	Critical Fault from All SARs
69	No Reaction	CoreA Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data0 feedback check alarm
70	No Reaction	CoreA Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data1 feedback check alarm
71	No Reaction	CoreA Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data2 feedback check alarm
72	No Reaction	CoreA Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data3 feedback check alarm
73	No Reaction	CoreA Instruction Tag	Via FCCU Fake bit to set the fault at error path	IC tag feedback check alarm
74	No Reaction	CoreA DMEM Controller	Via FCCU Fake bit to set the fault at error path	DTCM0 feedback check alarm
75	No Reaction	CoreA DMEM Controller	Via FCCU Fake bit to set the fault at error path	DTCM1 feedback check alarm
76	No Reaction	CoreA IMEM Controller	Via FCCU Fake bit to set the fault at error path	ITCM feedback check alarm
77	No Reaction	CoreB Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data0 feedback check alarm
78	No Reaction	CoreB Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data1 feedback check alarm
79	No Reaction	CoreB Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data2 feedback check alarm
80	No Reaction	CoreB Instruction cache	Via FCCU Fake bit to set the fault at error path	IC data3 feedback check alarm

Table continues on the next page...

Table 78-3. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
81	No Reaction	CoreB Instruction Tag	Via FCCU Fake bit to set the fault at error path	IC tag feedback check alarm
82	No Reaction	CoreB DMEM Controller	Via FCCU Fake bit to set the fault at error path	DTCM0 feedback check alarm
83	No Reaction	CoreB DMEM Controller	Via FCCU Fake bit to set the fault at error path	DTCM1 feedback check alarm
84	No Reaction	CoreB IMEM Controller	Via FCCU Fake bit to set the fault at error path	ITCM feedback check alarm
85	No Reaction	CoreA/0 interrupt controller	This alarm can be triggered by enabling the INTC_MMR and setting the INTC_TIMER value to be larger than the INTC_LAT value (which itself needs to be non-zero). + Via FCCU Fake bit to set the fault at error path	INTC monitor error
86	No Reaction	CoreB/1 Interrupt controller	This alarm can be triggered by enabling the INTC_MMR and setting the INTC_TIMER value to be larger than the INTC_LAT value (which itself needs to be non-zero). + Via FCCU Fake bit to set the fault at error path	INTC monitor error
87	No Reaction	Decorated memory Controller	Via FCCU Fake bit to set the fault at error path	PRAM decorated storage transaction monitor error
88	No Reaction	Computational SMPU	Via FCCU Fake bit to set the fault at error path + LBIST	MPU alarm to the fault collection unit An internal error has happened in the operation of the MPU. Example: MPU erroneously blocked transactions without a real permissions violation will assert this NCF.
89	No Reaction	peripheral SMPU	Via FCCU Fake bit to set the fault at error path + LBIST	MPU transaction monitor error output An internal error has happened in the operation of the MPU. Example: MPU erroneously blocked transactions without a real permissions violation will assert this NCF.
90	No Reaction	Core0/1 IMEM MBIST	Via FCCU Fake bit to set the fault at error path	Mbist multi bit error of IMEM (Core0 or Core1)
91	No Reaction	Flash	Via FCCU Fake bit to set the fault at error path + LBIST	Flash Read Voltage error or reference current error
92	No Reaction	Core0/1 DMEM MBIST	Via FCCU Fake bit to set the fault at error path	Mbist multi bit error of DMEM (Core0 or Core1)
93	No Reaction	Core0/1 ICACHE/TAG MBIST	Via FCCU Fake bit to set the fault at error path	Mbist multi bit error of ICACHE(Core0 or Core1)

Table continues on the next page...

Table 78-3. FCCU Fault Inputs (continued)

NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
94	No Reaction	SRAM MBIST	Via FCCU Fake bit to set the fault at error path	Mbist Multibit error of SRAM
95	No reaction	All Peripheral RAM MBIST	Via FCCU Fake bit to set the fault at error path	Mbist multi bit error from all peripheral RAM

78.1.6 Lockstep Status

RCCU status is evaluated in FCCU_NCF[57] channel.

In debug mode, the cores do not work in lockstep mode and the RCCUs are disabled. The actual enablement/disablement state of the RCCUs (including any automatic disablement by debug) can be read by software.

78.1.7 FOSU Count Value

The fixed preprogrammed value for FOSU_Count is 0x1F40.

78.1.8 FCCU_CFG[FCCU_SET_AFTER_RESET]

FCCU_CFG[FCCU_SET_AFTER_RESET] is not used to control EOUT pads on this chip and hence there will be no effect of writing to this bit. FCCU EOUT pads OBE are controlled either via SIUL or DCF records.

78.2 Introduction

The Fault Collection and Control Unit (FCCU) offers a hardware channel to collect faults and to place the device into a safe state when a failure in the device is detected. No CPU intervention is requested for collection and control operation.

The FCCU offers a systematic approach to fault collection and control. The distinctive features of the module are:

- Redundant collection of fault information from safety relevant modules on the device
- Collection of test results
- Configurable fault control
- Configurable internal reactions for each NCF:
  - No reaction
  - IRQ
  - Short functional reset
  - Long functional reset
  - NMI
- External reactions via configurable output pins
- Watchdog timer for the reconfiguration phase
- The FCCU's configuration is lockable:
  - Permanently locked until next reset or transiently locked until a specific key is written. FCCU gets transiently locked again if an invalid key is written into FCCU_TRANS_LOCK register (that is, other than 0xBC). Key to lock the FCCU permanently is 0xFF, written in the FCCU_PERMNT_LOCK register.
- Self checking capabilities: FSM redundancy and parity check for the configuration register
- One of the error out pins is high to indicate operational (OK) state (in bi-stable operation mode). The above is not true in case the error out protocol is configured to be a toggling protocol, for example, dual rail and time switching.
- After power on, the EOUT signals have high impedance.[1] They show operational state only on software request.
- In case of a failure event or on software request for Error pin indication, the pin(s) are set to faulty state for a minimum time T_min, even if SW tries to release it before (for the case of error pin configured in Bi-stable mode only).
- The overall fault detection, processing and indication time is less than 10 ms.
- The actual maximum time is 5 CLKSAFE cycles**.**

1. Actual value depends on device specific setting at pad level.

The FCCU circuitry is checked at the start-up by the self-checking procedure. The FCCU is operative with the default configuration immediately after the completion of the self-checking procedure. Internal (short or long functional reset request pulse, interrupt request) and external (EOUT signaling) reactions are statically defined or programmable. The default configuration can be modified only in the CONFIG state. FCCU is designed to function when CLKSYS is faster than the CLKSAFE clocks.

78.2.1 Acronyms and abbreviations

Term	Description
CF	Critical fault
CPU	Central processing unit
EOUT	Error out
FOSU	FCCU output supervision unit
FSM	Finite state machine
INTC	Interrupt controller
intf	Interface
IRQ	Interrupt request
MC_ME	Mode entry module
NCF	Noncritical fault
NMI	Nonmaskable interrupt
SoC	System-on-chip
STCU	Self-test control unit
SW	Software
WKPU	Wake-up unit

78.3 Main features

- Management of non-critical faults
- HW or SW fault recovery management
- Fault detection and collection
- Fault injection (fake faults)
- External reaction (fault state): EOUT signaling. Error indication via the pin(s) is controlled by the FCCU.
- Internal chip reactions (alarm state): interrupt request
- Internal chip reactions (fault state):

- long functional reset request pulse
- short functional reset request pulse
- NMI

- Bi-Stable, Dual-Rail and Time Switching output protocols on EOUT
- Internal (to the FCCU) watchdog timer for the reconfiguration phase
- Configuration lock
- Self-checking capabilities: FSM redundancy and parity check for the configuration registers

78.4 Block diagram

The following figure is a top-level diagram of the FCCU module.

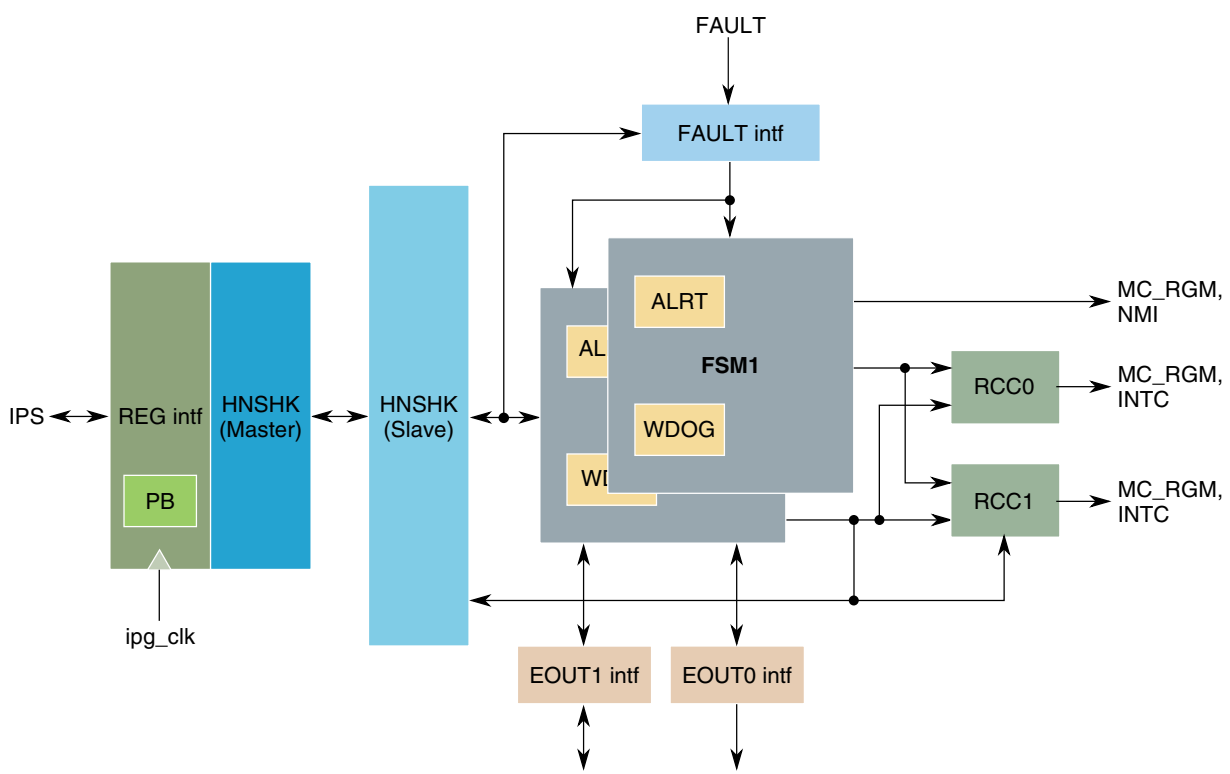


Figure 78-2. FCCU block diagram

This table describes the FCCU submodules.

Table 78-4. FCCU submodules

Submodule	Description
REG intf	Includes the register file, the IPS bus interface, the IRQ interface and the parity block (PB) for the configuration registers
HNSHK blocks (master and slave blocks)	Includes the FSM's ability to support the handshake between the REG interface and the FSM unit due to the usage of 2 asynchronous clocks (IPS system clock and RC oscillator clock)
FSM unit	Implements the main functions of the FCCU. The FSM also includes the: • Watchdog timer (WDG) • Alarm timer (ALRT)
FAULT intf	Implements the interface for the fault conditioning and management
EOUTx units	Implement the output stage to manage the EOUT interfaces
RCCx units	Implement the redundancy control checker to monitor the FSMx unit state and its configuration

78.5 Signal description

78.5.1 Signals

This table describes the signals on the boundary of FCCU.

Signal	Direction	Function	Active level	Clock
Clock and reset				
CLKSYS	Input	System Clock—Supplies the primary clock.	Rising edge	—
CLKSAFE0	Input	Safe Clock 0—Supplies one of two 16 MHz clocks for use by various redundant submodules. This clock and CLKSAFE1 are often referred to as the same clock (CLKSAFE).	Rising edge	—
CLKSAFE1	Input	Safe Clock 1—Supplies one of two 16 MHz clocks for use by various redundant submodules. This clock and CLKSAFE0 are often referred to as the same clock (CLKSAFE).	Rising edge	—
EOUT interface				
EOUT[1:0]	Output	Error Output—Indicate FCCU's condition (Faulty, Nonfaulty, or Configuration) to off-chip logic.	Programmable	CLKSAFE0, CLKSAFE1
FOSU interface				

Table continues on the next page...

Signal	Direction	Function	Active level	Clock
FIF	Output	FCCU In Fault—Indicates to the FOSU module that FCCU is recognizing a fault so that FOSU doesn't mistakenly generate a destructive chip reset. FCCU asserts and then deasserts this signal in two cases: *Case 1* When the EOUT signals are programmed to be always low (FCCU_SET_CLEAR), FCCU asserts this signal until the EOUT signals are programmed not to be always low. *Case 2* When the EOUT signals are programmed not to be always low (FCCU_SET_CLEAR), FCCU asserts this signal when all of the following are true: • FCCU enters Fault state as the result of a fault. • If the fault is a noncritical fault and the EOUT signals are in Bistable fault-output mode (FOM), the EOUT signals are enabled for the fault (EOUTENx). FCCU deasserts this signal when FCCU leaves Fault state.	High	CLKSAFE0, CLKSAFE1
WKPU interface				
NMIWKPU	Input	WKPU Nonmaskable Interrupt Request—Receives nonmaskable interrupt requests from the WKPU module.	Low	CLKSYS
WAKEWKPU	Output	Wake WKPU—Wakes the WKPU module to bring the chip out of dormant state when FCCU enters Fault state as the result of a noncritical fault for which NMI is enabled as the reaction (FCCU_NMI_ENa[NMIENx]).	High	CLKSAFE0, CLKSAFE1
Interrupt interface				
NMIOUT	Output	Nonmaskable Interrupt Output—Sends a nonmaskable interrupt request to the processor core or cores when FCCU enters Fault state as the result of a noncritical fault for which NMI is enabled as the reaction (FCCU_NMI_ENa[NMIENx]).	Low	CLKSYS

78.6 Functional description

78.6.1 Definitions

In general, the following definitions are applicable for the fault management:

- HW recoverable fault: the fault indication is an edge-triggered and level-sensitive signal that remains asserted until the fault cause is deasserted. That is, if logical 0 on the fault signal indicates fault, then the status flags are valid as long as the fault line stays at 0. The status is automatically cleared when the fault signal goes to 1. Typically the fault signal is latched external to the FCCU in the module where the fault occurred. The FCCU state transitions are consequently executed on the state changes of the input fault signal. No SW intervention in the FCCU is required to recover the fault condition.
- SW recoverable fault: the fault indication is a signal asserted without a defined time duration. The fault signal is latched in the FCCU. The fault recovery is executed following a SW recovery procedure (status/flag register clearing).

HW recoverable is an option to exclude the handling of error source/s by FCCU management SW, in case it is known that the fault is recoverable by itself when the fault condition gets corrected.

These type of chip resets reset the FCCU:
- Power-on
- Destructive
- Long functional, when initiated by the assertion of the chip reset pin (RESET_B)

These types of chip resets don’t reset the FCCU:
- Long functional, when initiated by anything other than the assertion of the chip reset pin (RESET_B)
- Short functional reset

For more information on each type of reset, see the MC_RGM and reset chapters.

78.6.2 FSM description

FCCU functionality is depicted by the finite state machine (FSM) state diagram.

Basically four states are identified with the following meaning:

- Configuration: Used only to modify the configuration of FCCU from its default. A subset of the FCCU registers, dedicated to define the FCCU configuration (global configuration, reactions to fault, timeout, noncritical fault masking) can be accessed in write mode only in Configuration state.

  Configuration state is accessible only in Normal state and if the configuration is not locked. The permanent configuration lock can be disabled by a reset which also resets the FCCU., The transient lock register is unlocked by writing BCh into it. FCCU gets transiently locked again if an invalid key is written into FCCU_TRANS_LOCK register (that is, other than BCh). Key to lock FCCU permanently is FFh, written in the FCCU_PERMNT_LOCK register.

  After the release of reset the state of the transient lock will be locked. The state of the permanent lock will be unlocked.

  The locking feature only restricts the FSM movement into Configuration state. Once the user enters Configuration state and then tries to lock the configuration, the locking of configuration will be effective after FCCU moves to Normal state, it will not be effective in the current Configuration state.

  The Configuration to Normal state transition can be executed by SW or automatically following a timeout condition of the watchdog. In case the timeout information and the SW request to mode change to Normal appears at the same time, watchdog timeout has the priority and hence the Configuration registers (those that are writable only in Configuration state) are reset to their default values. The movement to Normal is made.

  The incoming faults, occurring during the configuration phase (Configuration state) are latched in order to process them when FCCU is moved into Normal state, according to the new configuration.

  All pending faults that occur during Configuration state result in both of the following:

  - Highest-priority state transition
  - Interrupt generation (NMI or alarm IRQ)

  If the state transition occurs, it gives the reset reaction corresponding to the worst case based on all the faults (pending or nonpending faults) that occurred during Configuration state.

- Normal: This is FCCU's operating state when no faults are occurring. It is also the default state on the reset exit. Following state transitions occur on one of the following events:

    - unmasked noncritical faults with the timeout disabled → FCCU moves to Fault state
    - unmasked noncritical faults with the timeout enabled → FCCU moves to Alarm state
    - masked noncritical faults → FCCU stays in Normal state
- Alarm: FCCU moves into Alarm state when an unmasked noncritical fault occurs and the timeout is enabled. The transition to Alarm state goes along with an interrupt alarm, if enabled. By definition, this fault may be recovered within a programmable timeout period, before it generates a transition to Fault state. The timeout is reinitialized if FCCU enters Normal state. The timeout restarts following the recovery from Fault state.
- Fault: FCCU moves into Fault state when one of the following condition occurs:
    - timeout related to a noncritical fault when FCCU is in Alarm state
    - unmasked noncritical faults with the timeout disabled

The transition from Normal/Alarm to Fault state goes along with the generation of:

- NMI interrupt (optional)
- EOUT signaling (optional)
- SW option: Soft reaction (Short functional reset request pulse if configured)
- SW option: Hard reaction (Long functional reset request pulse if configured)
- Non Maskable Interrupt (NMI) is routed only to the Safety Core

After moving to Fault state, if there is either a previous pending fault or a new fault for which NMI is enabled, NMI generation takes place.

Multiple faults can occur at the same time. Consider the case where there is a fault for which Alarm state is enabled and IRQ is programmed along with a fault for which Alarm state is disabled. In that case, the FSM moves to Fault state, but with the generation of alarm IRQ for the fault which has IRQ enabled.

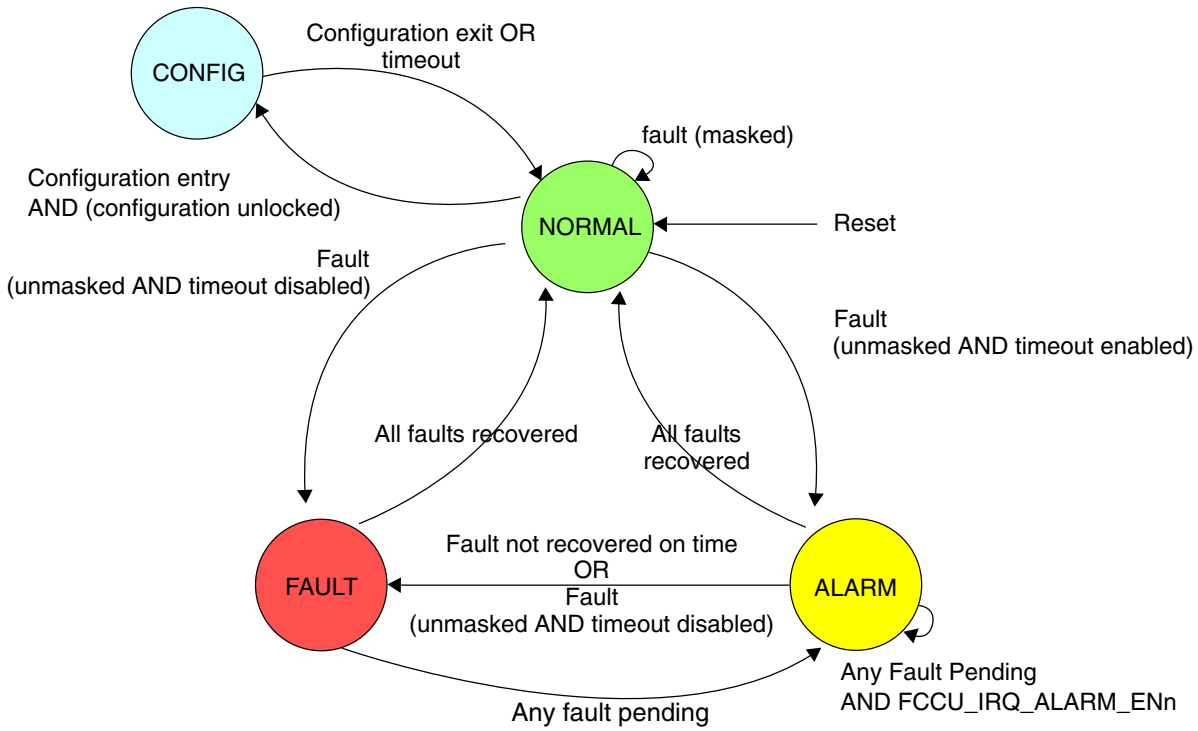


Figure 78-3. FCCU state diagram

78.6.3 Self-checking capabilities

The FCCU includes some features to support self-checking capabilities of the main FSM in running mode. The FSM unit is duplicated and its state (internal state and relevant outputs) is checked by 2 RCCx units (cycle accurate). The following checks (per CLKSAFE cycle) are provided by the RCCx units:

- all the FSM outputs and its internal state for the redundant FSM instances are checked to detect runtime faults on the FSM outputs and its internal state
- parity bits (computed at byte level) on the configuration registers are checked to detect run time faults on the configuration inputs of the FSM
- parity bits (computed at byte level) on the interface used to clear the FCCU_CF_Sx FCCU_UF_Sx and FCCU_RF_Sx status registers
- EOUT protocol state in dual-rail, time-switching and bi-stable mode
- common mode signals (handshake interface signals activated only in CONFIG state) congruence with the FSM state

In case of failure each RCCx unit provides the following reaction:

- an interrupt request to the INTC module.

The RCCx state is freeze in a SC freeze status register (FCCU_SCFS).

To guarantee the self checking capabilities through the FAULT interface, each critical/ unrecoverable fault source must be duplicated on a couple of CF UF inputs.

Two separate clock inputs (CLKSAFE0 and CLKSAFE1) are routed to the redundant modules (FSMx, RCCx, EOUTx, FAULT-if modules). An off-module monitor of the CLKSAFE clocks and an off-module reaction (functional reset to the MC) is required to cover a potential fault on the CLKSAFE clocks.

Self checking capabilities cover the HW reaction of the FCCU due to the assertion of an external fault. The IPS bus interface and the handshake modules are not covered by self-checking capabilities. Any operation executed by SW (configuration, fault recovery) must be cross-verified by congruence checks via SW (registers read following a write/ clear operation, internal FCCU state, ..).

A typical sequence with self checking failure and related reaction is given in Figure 78-4.

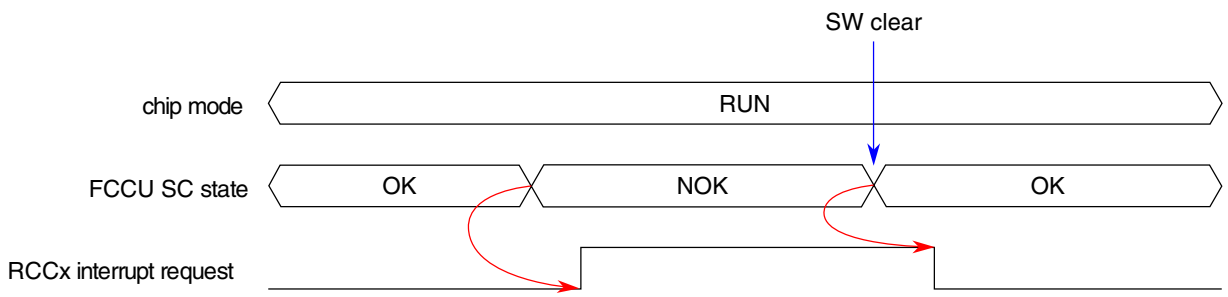


Figure 78-4. Self-checking reaction (interrupt)

For more information about how to enable or disable FCCU Self-checking, see Configuration (FCCU_CFG) and SC Freeze Status (FCCU_SCFS).

78.6.4 Fault priority scheme and nesting

The FAULT state has a higher priority than the ALARM state in case of concurrent fault events (non-critical) that occur in the NORMAL state.

The ALARM to FAULT state transition occurs if a non-critical fault (unmasked and with time-out disabled) is asserted in the ALARM state.

The ALARM to NORMAL state transition occurs only if all the non-critical faults (including the faults that have been collected after the entry in the ALARM state) have been cleared (SW or HW recovery); otherwise the FCCU will remain in the ALARM state.

The FAULT to NORMAL state transition occurs only if all the non-critical faults (including the faults that have been collected after the entry in the FAULT/ALARM state) have been cleared (SW or HW recovery); otherwise the FCCU will move to the ALARM state (if any non-critical fault is still pending and the time-out is not elapsed).

In general, no fault nesting is supported except for the non-critical faults that cause an ALARM to FAULT state transition. In this case the NCF timer is stopped until the FAULT state is recovered. If FCCU is in ALARM state and another fault comes, which has its alarm timeout enabled, then the alarm timer shall not reload and shall not start again.

78.6.5 Fault recovery

The following timing diagrams describe the main use cases of the FCCU in terms of fault events and related recovery.

A typical sequence related to a non-critical FAULT management (ALARM state), see Figure 78-5 and Figure 78-6, is as follows:

- non-critical fault assertion
- FCCU state transition (automatic): NORMAL → ALARM
  - alarm interrupt request (if enabled)
  - time-out running
- system state: RUN
- alarm interrupt management: FAULT recovery (by SW)
  - FCCU state transition ALARM → NORMAL

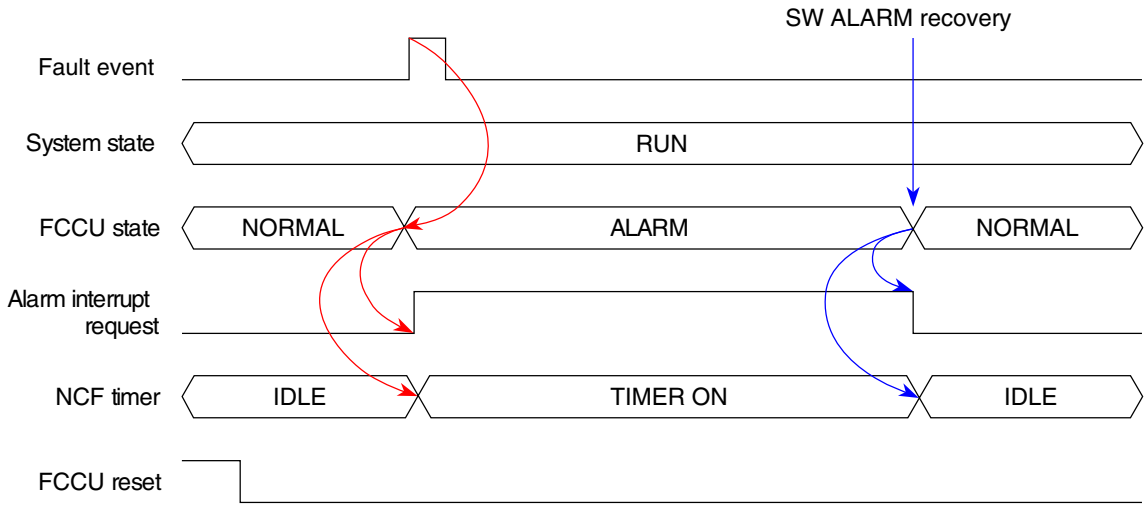


Figure 78-5. Non-critical FAULT (ALARM state) recovery (a)

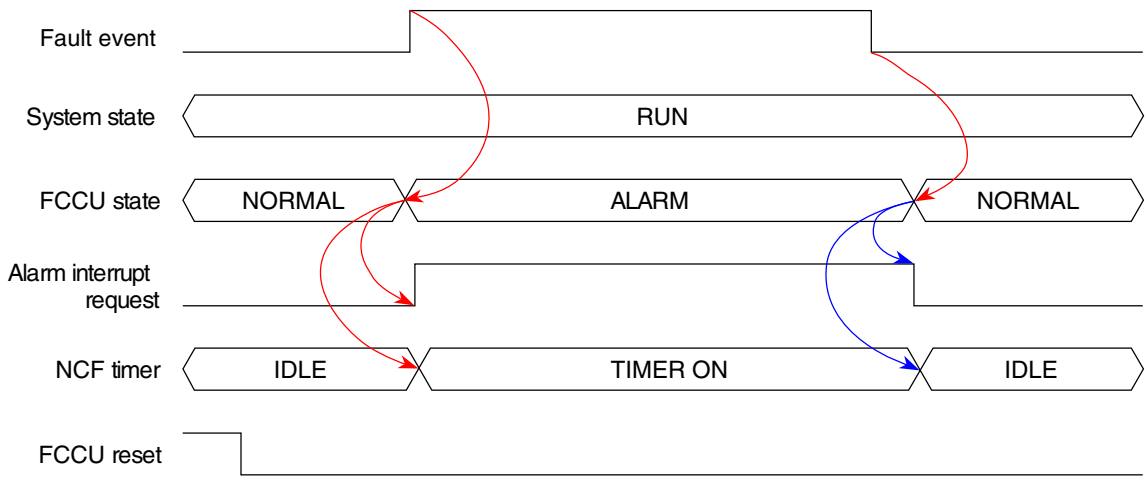


Figure 78-6. Non-critical FAULT (ALARM state) recovery (b)

A typical sequence related to a FAULT management (ALARM → FAULT state), see Figure 78-7, is as follows:

- non-critical fault assertion
- FCCU state transition (automatic): NORMAL → ALARM
  - alarm interrupt request (if enabled)
  - time-out running
- FCCU state transition (following the time-out trigger): ALARM → FAULT
  - NMI assertion (if enabled)
- system state transition: RUN → SAFE
- NMI interrupt management (if enabled)

- FAULT recovery (by SW): FCCU state transition FAULT → NORMAL
- system state transition: SAFE → RUN

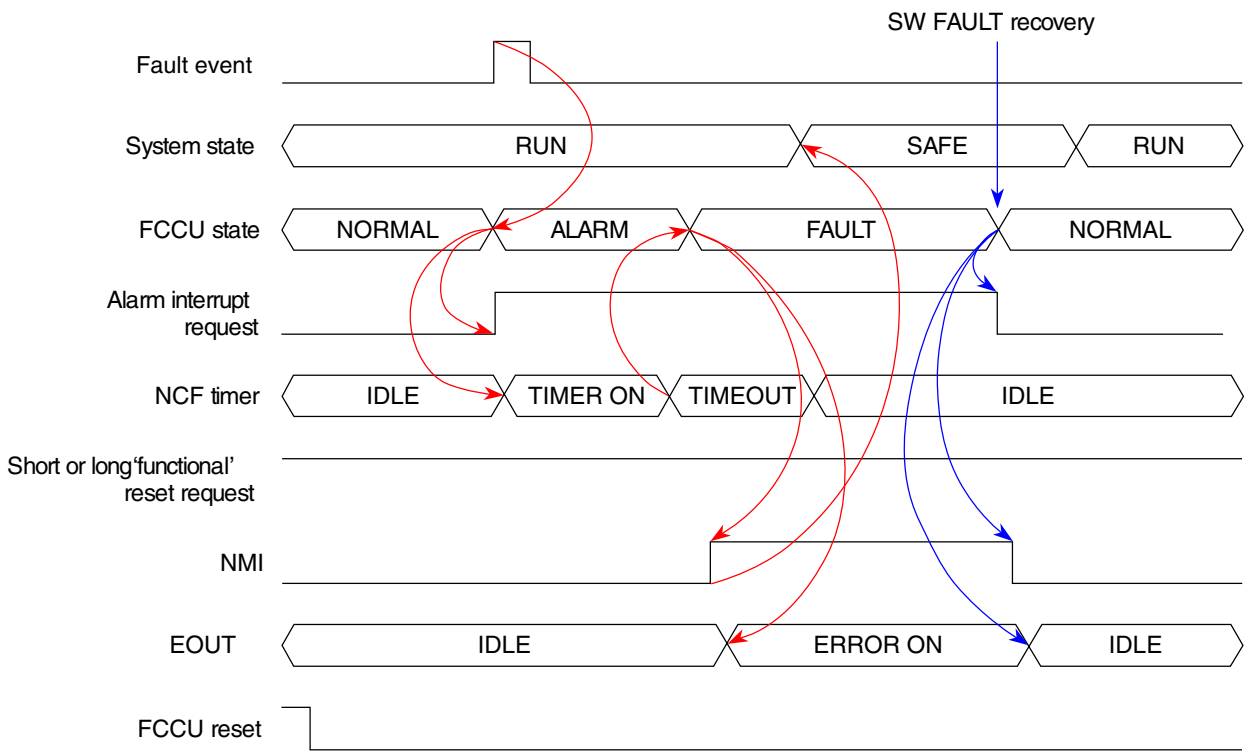


Figure 78-7. FAULT (ALARM → FAULT state) recovery

78.6.6 NMI/WKPU interface

NMIOUT is connected to all processor cores on the chip but is masked when any one of the following is true:

- An FCCU asynchronous reset is in progress.
- The chip is in RESET mode.
- The chip has automatically entered DRUN mode from RESET mode and software has not yet requested another mode.

This figure shows the logical scheme.

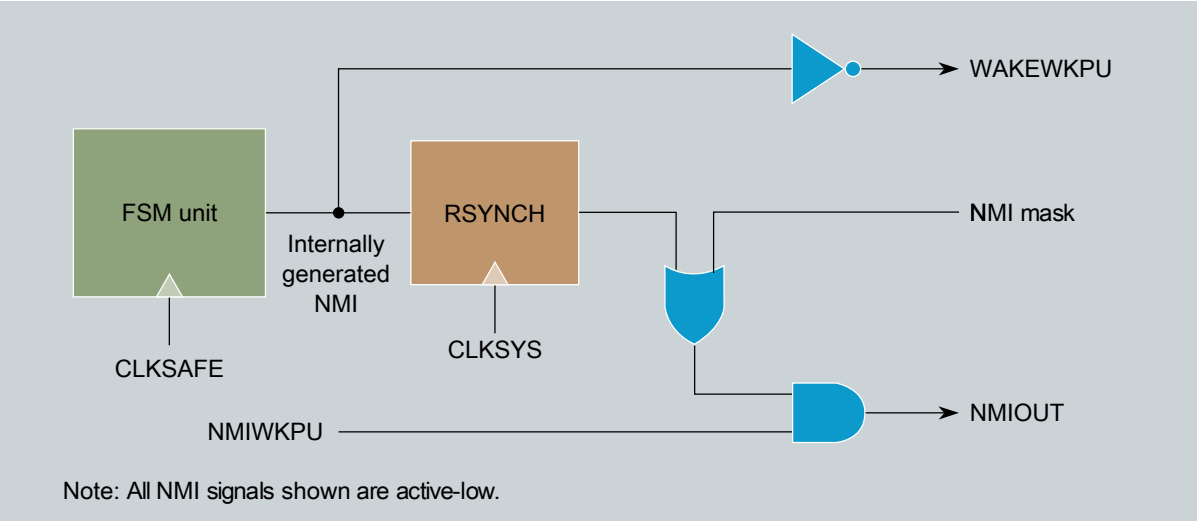


Figure 78-8. NMI/WKPU scheme

78.6.7 STCU interface

The STCU interface includes:

- A set of signals resulting from the self-checking procedure connected externally at the FCCU.

  The STCU fault signals are processed by the FCCU when the SoC is re-booted following the self-testing procedure. The STCU includes a status register that stores the self-testing results (flags).

- a mask that inhibits the EOUT dummy signaling until the STCU self-checking procedure has been completed.

During the self-testing procedure, depending on the STCU results, three cases are applicable:

1. STCU completes the self-testing procedure successfully. The SoC reboots and the FCCU is responsible for fault collection and indication.
2. STCU completes the self-testing procedure with low severity failures. The FCCU records the fault condition but does not, by default, trigger a reset cycle.
3. STCU completes the self-testing procedure with serious failures. The FCCU triggers a reset cycle in response to this fault input from the STCU. Because the fault condition survives reset, new reset cycles are subsequently triggered until the reset escalation feature takes control, freezing the chip in the reset state until a POR. This feature is optionally programmable using MC_RGM.

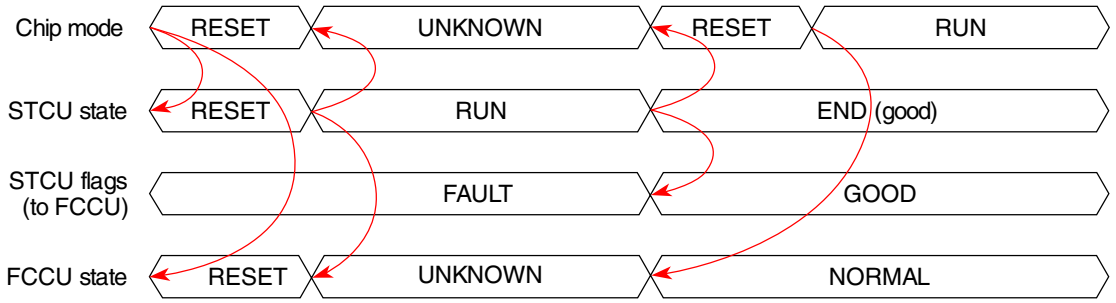


Figure 78-9. STCU-FCCU (case 1)

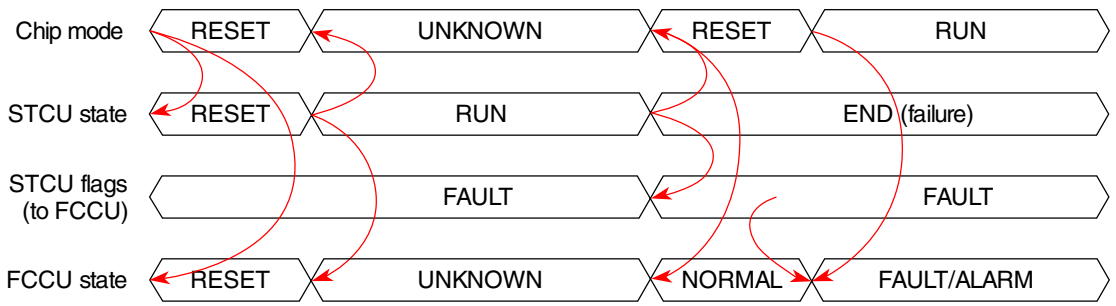


Figure 78-10. STCU-FCCU (case 2)

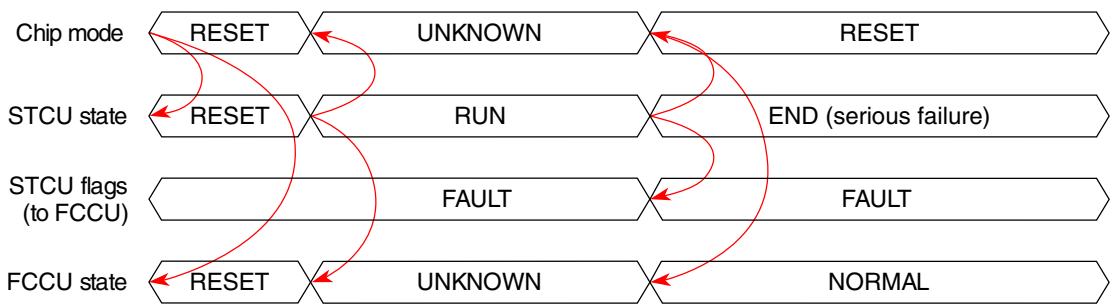


Figure 78-11. STCU-FCCU (case 3)

78.6.8 EOUT interface

The FCCU provides up to two bidirectional signals (EOUT[1:0]) as a failure indication to the external world. For information on the availability and names of these FCCU signals on the boundary of this chip, see the chip-specific FCCU information.

Different fault-output modes (protocols) for the fault-output (EOUT) interface are supported (FCCU_CFG[FOM]):

- Dual-Rail
- Time-Switching

- Bistable
- Test

You can further configure the fault-output modes using the following attributes:

Attribute	Field	Setting used in the example diagrams and tables that follow
Switching mode	FCCU_CFG[SM]	Slow
Polarity selection	FCCU_CFG[PS]	For the faulty indication, EOUT1 is high, and EOUT0 is low.

The EOUT frequency is generated by dividing the CLKSAFE frequency by a fixed factor of 2^{18}.

With a CLKSAFE frequency of 16 MHz, this drives a signal of 61 Hz on EOUT.

The external monitor of the EOUT protocol should oversample the EOUT signals in order to synchronize periodically the external clock (used by the monitor) and CLKSAFE (which detects the edge transition of the EOUT protocol) in Dual-Rail or Time-Switching fault-output mode.

In case of a failure event or on software request for EOUT indication, the signal(s) are set to the faulty state for a minimum time T_min, even if software tries to release it before. If software configures the error pins to OK(1), and if a fault comes trying to drive the pin to NOK(0), then priority is given to the fault indication and the error signals indicate NOK, such as an incoming fault is not masked when software has set the error signal to high. During the T_min by a non-software fault, the FCCU FSM moves independently of this signal state (low), and as soon as the timer expires, the pin behavior is dictated by the state in which the FSM finds itself in, and it is not possible to set the signals to OK by software moving FCCU to Configuration state, as long as this timer is running. No software intervention is needed to bring the signal from the low state.

Software can bring the pin back to OK state by clearing the faults and waiting for the T_min interval to expire, after which the FCCU automatically enters Normal state and the error signal indicates OK.

In case another failure event happens within T_min after a first one, the T_min counter is restarted.

These resets influence the state of the failure indication pins:

- Power on reset
- Destructive reset (including when an FOSU timeout occurs due to FCCU failing to react)
- Chip pin reset (RESET_B)

Other resets should not influence the state of the failure indication pins.

NOTE

The transition from Fault to Configuration state is not possible but is shown to display the behavior in all four states—reset, normal, error, and configuration.

NOTE

FCCU remains in Normal state when a fault is disabled; therefore, NA (not applicable) appears in the following table.

NOTE

The following table is valid only after software has enabled the EOUT signals (FCCU_CFG[FCCU_SET_AFTER_RESET].

Table 78-5. FCCU error pin behavior on state transitions

Fault signaling state	Transition from Normal to Fault state			Normal+Configuration states			Transition from Normal to Alarm state		
	with Fault disabled | enabled						with Fault disabled | enabled		
	Internal error pin	Error pin enable	Error pad	Internal error pin	Error pin enable	Error pad	Internal error pin	Error pin enable	Error pad
Disabled	NA | OK	NA | 1	NA | OK	OK	1	OK	NA | OK	NA | 1	NA | OK
Enabled	NA | NOK	NA | 1	NA | NOK	OK	1	OK	NA | OK	NA | 1	NA | OK

NOTE

The first time FCCU is in Configuration state, the EOUT pads are in a high-impedance state; thereafter, their states reflect the programmed EOUT protocol.

78.6.8.1 Dual-rail protocol

Dual-rail encoding is an alternate method for encoding bits. In contrast with classical encoding, where each signal carries a single-bit value, dual-rail encoded circuits use two wires to carry each bit. The encoding scheme is given in Table 78-6 and the related timing diagram is given in Figure 78-12 and Figure 78-13.

Table 78-6. Dual-rail encoding

Logical state	Dual-rail encoding (output pins EOUT[1:0])	Note
non-faulty	10	toggling
non-faulty	01	

Table continues on the next page...

Table 78-6. Dual-rail encoding (continued)

Logical state	Dual-rail encoding (output pins EOUT[1:0])	Note
faulty	00	toggling
faulty	11	
reset	Hi impedence[1]	no toggling
configuration	same as NORMAL	toggling

1. Final value depends on the SoC setting at pad level.

As long as FCCU is in NORMAL or ALARM state, output will show "non-faulty" signal. Output pins EOUT[0:1] will toggle between 01 and 10 with a given frequency of 61 Hz.

In the RESET phase the output pins are set as "high impedance".

Note

Figure 78-12 and Figure 78-13 are formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

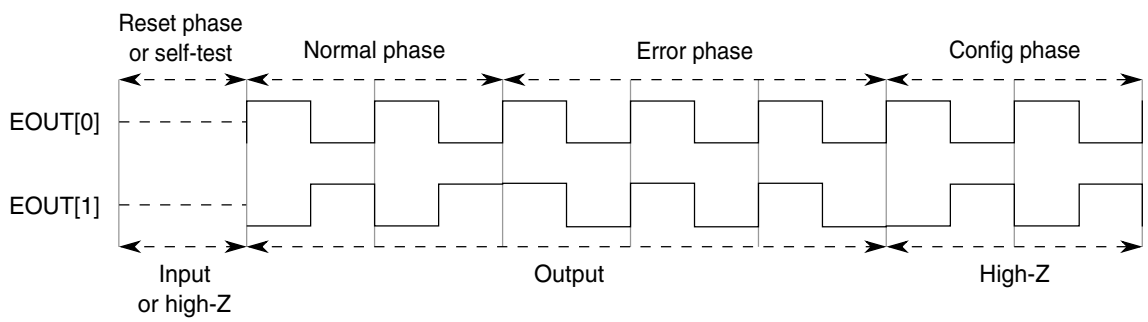


Figure 78-12. Dual-rail protocol (slow switching mode)

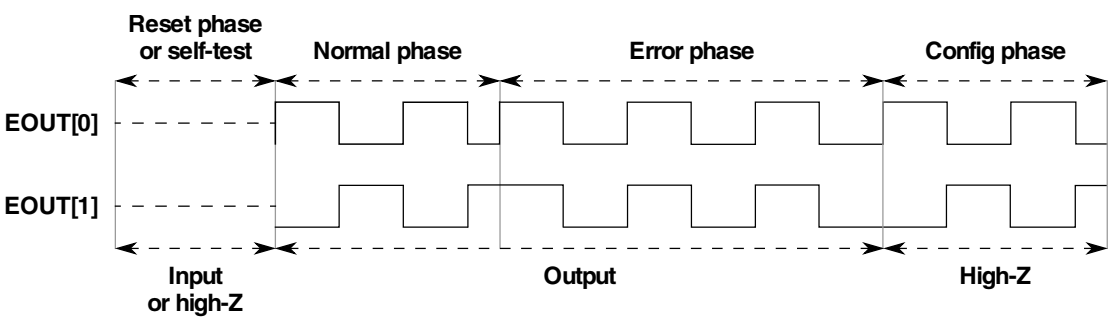


Figure 78-13. Dual-rail protocol (fast switching mode)

78.6.8.2 Time-switching protocol

The encoding scheme is given in Table 78-7 and the related timing diagram is given in Figure 78-14.

Table 78-7. Time-switching encoding

Logical state	Time-switching encoding (output pins EOUT[1:0])	Note
non-faulty	10	toggling
non-faulty	01	
faulty	10	no toggling
reset	high-Z[1]	no toggling
configuration	same as NORMAL	toggling

1. Final value depends on device specific settings at pad level.

As long as FCCU is in NORMAL or ALARM state, outputs will show "non-faulty" signal. Output pins #0, #1 will toggle between 01 and 10 with frequency of 61 Hz.

In the FAULT state, the output pin EOUT[0] is set as low.

In Time Switching mode the second output (EOUT[1]) is the inverted signal of first output (EOUT[0]).

In the RESET phase the output pins are set as "high impedance".[2]

Note

Figure 78-14 is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

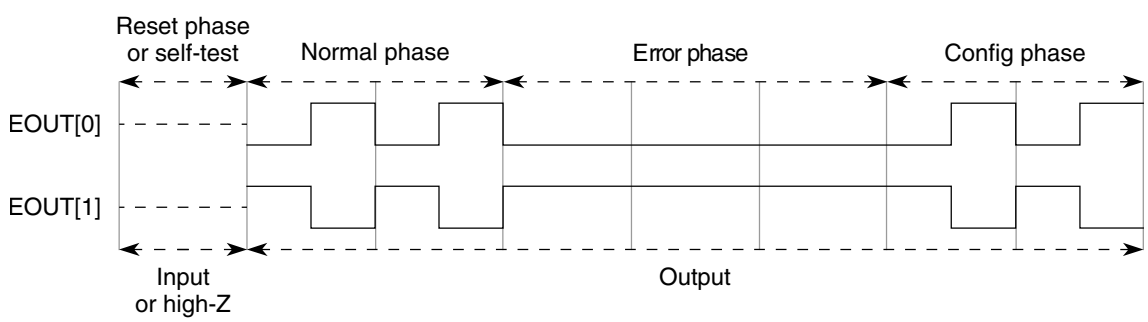


Figure 78-14. Time-switching protocol

2. Actual value depends on device specific settings at pad level.

78.6.8.3 Bi-stable protocol

The encoding scheme is given in Table 78-8 and the related timing diagram is given in Figure 78-15.

Table 78-8. Bi-stable encoding

Logical state	Bi-stable encoding (output pins EOUT[1:0])	Note
non-faulty	01	no toggling
faulty	10	no toggling
reset	high-Z[1]	no toggling
configuration	same as NORMAL	no toggling

1. Final value depends on device specific settings at pad level.

In the FAULT state, the faulty logical state is indicated. In NORMAL or ALARM state, "no-faulty" state is indicated. In Bi-stable mode the second output (EOUT[1]) is the inverted signal of first output (EOUT[0]).

In the RESET phase the output pins are set as "high impedance".[3]

Note

Figure 78-15 is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

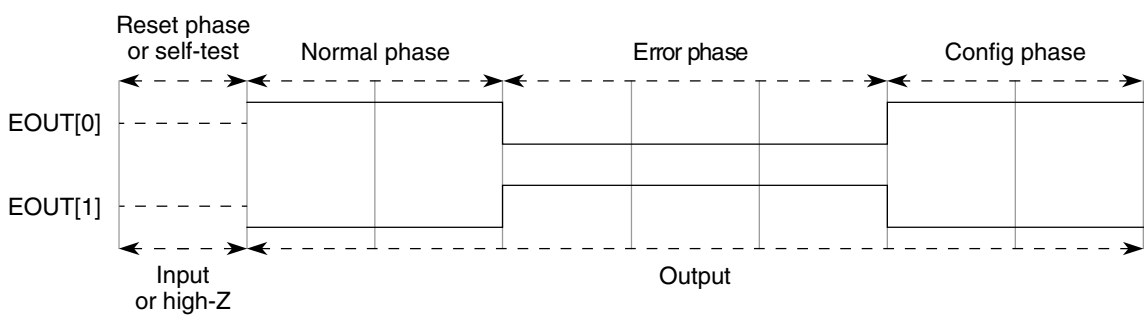


Figure 78-15. Bi-stable protocol

78.6.9 Error signal flow

See the chip-specific FCCU information for the diagram of the signal flow.

3. Actual value depends on device specific settings at pad level.

78.7 Register descriptions

The FCCU registers are listed in the table below. Any address offset not explicitly mentioned in this table is reserved.

The FCCU supports word (32-bit), half-word (16-bit), and byte (8-bit) read and write accesses.

Follow these register-access guidelines:

- Don't read from or write to any addresses that are not shown in the following table. Doing so may result in a transfer error.
- Don't write to any of the configuration registers unless FCCU is in Configuration state. Doing so results in a transfer error.
- Don't write to FCCU_TRANS_LOCK or FCCU_PERMNT_LOCK unless your code runs in Supervisor mode. Doing so results in a transfer error.

All the registers accessible in write mode only in CONFIG state are referred as configuration registers. These configuration registers return to the default value after configuration watchdog timer expires. These are the registers protected by FCCU_TRANS_LOCK and FCCU_PERMNT_LOCK registers. The configuration register setting has effect only when the FCCU state exits from the CONFIG state.

For each possible NCF failure source a different reaction shall be configurable through the use of NMI, IRQ, long/short reset selection registers as well as no reaction by disabling the former registers. It is not possible for a single event upset to switch off all reactions on failures as implementation is per fault source (but it will be possible to switch them all off by SW if intended). Failures themselves are not able to disable all reactions and indications.

The FCCU is not reset by short or long functional resets.

FCCU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control (FCCU_CTRL)	32	R/W	0000_00C0h	78.7.1/3310
4	Control Key (FCCU_CTRLK)	32	W	0000_0000h	78.7.2/3313
8	Configuration (FCCU_CFG)	32	R/W	See section	78.7.3/3314
1C	Noncritical Fault Configuration (FCCU_NCF_CFG0)	32	R/W	See section	78.7.4/3317
20	Noncritical Fault Configuration (FCCU_NCF_CFG1)	32	R/W	See section	78.7.4/3317
24	Noncritical Fault Configuration (FCCU_NCF_CFG2)	32	R/W	See section	78.7.4/3317

Table continues on the next page...

FCCU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4C	Noncritical Fault State Configuration (FCCU_NCFS_CFG0)	32	R/W	See section	78.7.5/3318
50	Noncritical Fault State Configuration (FCCU_NCFS_CFG1)	32	R/W	See section	78.7.5/3318
54	Noncritical Fault State Configuration (FCCU_NCFS_CFG2)	32	R/W	See section	78.7.5/3318
58	Noncritical Fault State Configuration (FCCU_NCFS_CFG3)	32	R/W	See section	78.7.5/3318
5C	Noncritical Fault State Configuration (FCCU_NCFS_CFG4)	32	R/W	See section	78.7.5/3318
60	Noncritical Fault State Configuration (FCCU_NCFS_CFG5)	32	R/W	See section	78.7.5/3318
80	Noncritical Fault Status (FCCU_NCF_S0)	32	R/W	0000_0000h	78.7.6/3319
84	Noncritical Fault Status (FCCU_NCF_S1)	32	R/W	0000_0000h	78.7.6/3319
88	Noncritical Fault Status (FCCU_NCF_S2)	32	R/W	0000_0000h	78.7.6/3319
90	Noncritical Fault Key (FCCU_NCFK)	32	W	0000_0000h	78.7.7/3321
94	Noncritical Fault Enable (FCCU_NCF_E0)	32	R/W	See section	78.7.8/3321
98	Noncritical Fault Enable (FCCU_NCF_E1)	32	R/W	See section	78.7.8/3321
9C	Noncritical Fault Enable (FCCU_NCF_E2)	32	R/W	See section	78.7.8/3321
A4	Noncritical Fault Timeout Enable (FCCU_NCF_TOE0)	32	R/W	See section	78.7.9/3323
A8	Noncritical Fault Timeout Enable (FCCU_NCF_TOE1)	32	R/W	See section	78.7.9/3323
AC	Noncritical Fault Timeout Enable (FCCU_NCF_TOE2)	32	R/W	See section	78.7.9/3323
B4	Noncritical Fault Timeout (FCCU_NCF_TO)	32	R/W	See section	78.7.10/ 3324
B8	Configuration-State Timeout (FCCU_CFG_TO)	32	R/W	See section	78.7.11/ 3324
BC	IO Control (FCCU_EINOUT)	32	R/W	0000_0000h	78.7.12/ 3325
C0	Status (FCCU_STAT)	32	R	0000_0010h	78.7.13/ 3327
C4	NA Freeze Status (FCCU_N2AF_STATUS)	32	R	0000_0000h	78.7.14/ 3329
C8	AF Freeze Status (FCCU_A2FF_STATUS)	32	R/W	0000_0000h	78.7.15/ 3330
CC	NF Freeze Status (FCCU_N2FF_STATUS)	32	R/W	0000_0000h	78.7.16/ 3331
D0	FA Freeze Status (FCCU_F2A_STATUS)	32	R	0000_0000h	78.7.17/ 3332
D4	SC Freeze Status (FCCU_SCFS)	32	R/W	0000_0000h	78.7.18/ 3333
DC	Noncritical Fault Fake (FCCU_NCFF)	32	R/W	0000_0000h	78.7.19/ 3335
E0	IRQ Status (FCCU_IRQ_STAT)	32	R/W	0000_0000h	78.7.20/ 3335
E4	IRQ Enable (FCCU_IRQ_EN)	32	R/W	0000_0000h	78.7.21/ 3337
E8	XTMR (FCCU_XTMR)	32	R	0000_0000h	78.7.22/ 3338

Table continues on the next page...

FCCU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
EC	Mode Controller Status (FCCU_MCS)	32	R	See section	78.7.23/ 3339
F0	Transient Lock (FCCU_TRANS_LOCK)	32	W	0000_0000h	78.7.24/ 3341
F4	Permanent Lock (FCCU_PERMNT_LOCK)	32	W	0000_0000h	78.7.25/ 3341
F8	Delta T (FCCU_DELTA_T)	32	R/W	0000_0000h	78.7.26/ 3342
FC	IRQ Alarm Enable (FCCU_IRQ_ALARM_EN0)	32	R/W	0000_0000h	78.7.27/ 3343
100	IRQ Alarm Enable (FCCU_IRQ_ALARM_EN1)	32	R/W	0000_0000h	78.7.27/ 3343
104	IRQ Alarm Enable (FCCU_IRQ_ALARM_EN2)	32	R/W	0000_0000h	78.7.27/ 3343
10C	NMI Enable (FCCU_NMI_EN0)	32	R/W	0000_0000h	78.7.28/ 3344
110	NMI Enable (FCCU_NMI_EN1)	32	R/W	0000_0000h	78.7.28/ 3344
114	NMI Enable (FCCU_NMI_EN2)	32	R/W	0000_0000h	78.7.28/ 3344
11C	Fault-Output (EOUT) Signaling Enable (FCCU_EOUT_SIG_EN0)	32	R/W	0000_0000h	78.7.29/ 3345
120	Fault-Output (EOUT) Signaling Enable (FCCU_EOUT_SIG_EN1)	32	R/W	0000_0000h	78.7.29/ 3345
124	Fault-Output (EOUT) Signaling Enable (FCCU_EOUT_SIG_EN2)	32	R/W	0000_0000h	78.7.29/ 3345

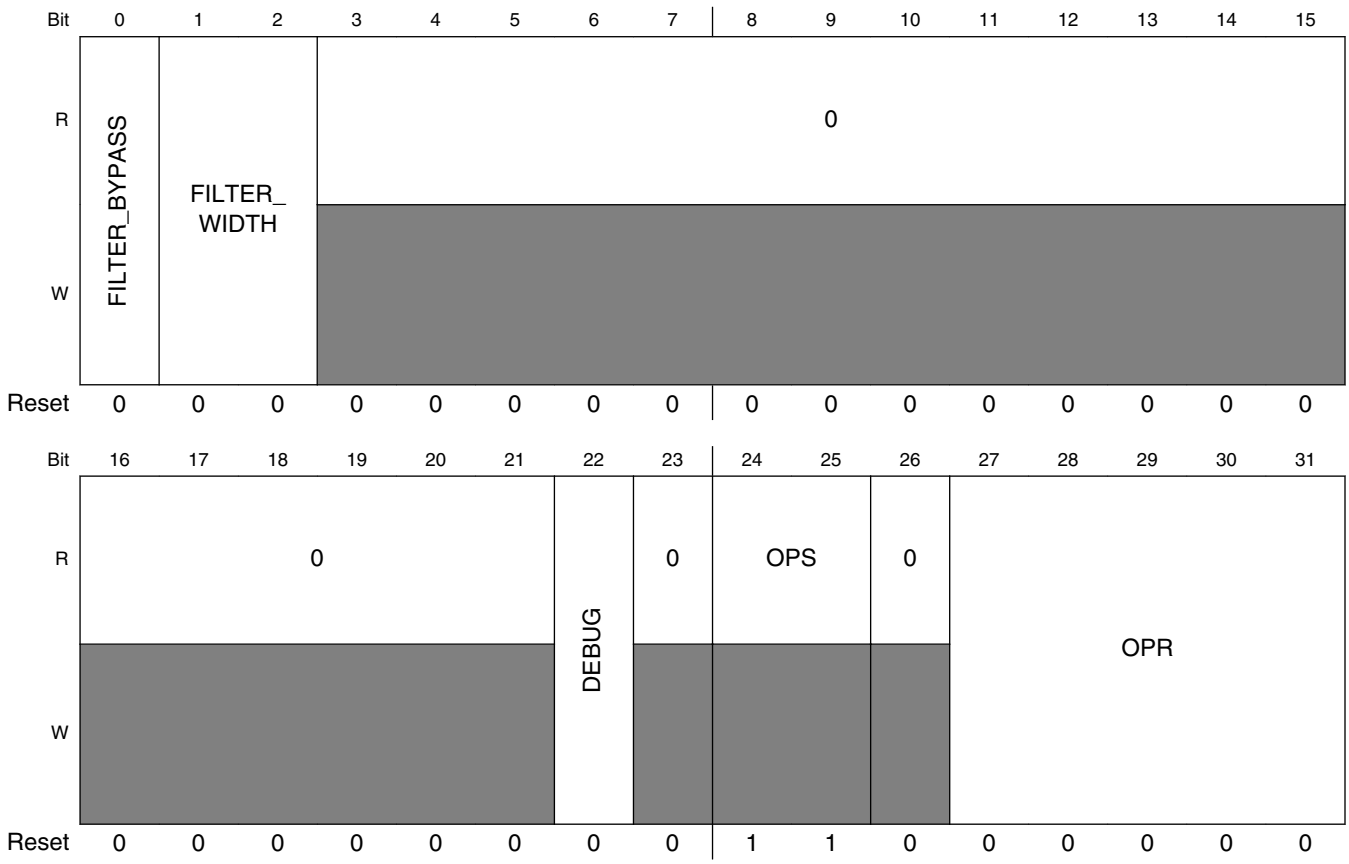
78.7.1 Control (FCCU_CTRL)

The FCCU_CTRL register allows the following operations to be executed:

- moving the FCCU state from the NORMAL state into the CONFIG state
- moving the FCCU state from the CONFIG state into the NORMAL state
- reading or to clear the NCF status register
- reading the FCCU FSM status register
- reading or to clear the FCCU freeze registers
- reading the ALARM timer
- reading the Watchdog timer

Some critical operations require a key as defined in the FCCU_CTRLK register.

Address: 0h base + 0h offset = 0h



FCCU_CTRL field descriptions

Field	Description
0 FILTER_ BYPASS	Filter bypass 0 glitch filter not bypassed 1 glitch filter bypassed
1–2 FILTER_WIDTH	Filter width 00 filters glitches up to 50 µs 01 filters glitches up to 75 µs 10 filters glitches up to 100 µs 11 filters glitches up to 100 µs
3–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 DEBUG	Debug Mode Enable Specifies whether Debug mode is enabled. If so, FCCU enters Debug mode when the Debug signal is asserted. When FCCU enters Debug mode, it halts operation and remains in the state it was in before it entered Debug mode. **NOTE:** FOSU doesn't halt when FCCU enters Debug mode. Therefore, FOSU can still cause a reset if a fault occurs while FCCU is in Debug mode.

Table continues on the next page...

FCCU_CTRL field descriptions (continued)

Field	Description
	0 Disabled 1 Enabled
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 OPS	Operation status. This bit can be read and cleared (via OP15 operation) by the software. 00 Idle 01 In progress 10 Aborted 11 Successful
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 OPR	Operation run. The SW application must not modify the OPR field (any write operation will be ignored) until the completion of the operation. Once the operation has been completed, the operation status (OPS) is set and the OPR field is automatically cleared (OPR = 000). The opcode OP12 must not be programmed. The OPR field is automatically set to OP12 when the FCCU_NCF_S registers are cleared by a write-clear operation into the related register. The opcode OP14 must not be programmed. The OPR field is automatically set to OP14 when the timeout occurs (FCCU_CFG_TO) during the configuration procedure => the FCCU state is automatically forced in NORMAL mode setting the default configuration. In this phase any write operation to the FCCU configuration registers is inhibited. In the case of OP15, the operation is successful immediately and OPS/OPR are cleared instantly. The operations OP21-OP30 are forbidden. In case of execution, they return an ABORT response without any side effect. The ABORT response occurs in the following cases: • wrong access (missing or wrong key) to the FCCU_NCF_S register (clear operation OP12) • wrong access (missing or wrong key) to the FCCU_CTRL register (OP1, OP2 operation) • OP1 (CONFIG command) execution when FCCU state /= NORMAL or configuration locked 00000 No operation [OP0] 00001 Set the FCCU into the CONFIG state [OP1] 00010 Set the FCCU into the NORMAL state [OP2] 00011 Read the FCCU state (see the FCCU_STAT register) [OP3] 00100 Read the FCCU frozen status flags (see the N2AF_STATUS register) [OP4] 00101 Read the FCCU frozen status flags (see the A2FF_STATUS register) [OP5] 00110 Read the FCCU frozen status flags (see the N2FF_STATUS register) [OP6] 00111 Read the FCCU frozen status flags (see the F2AF_STATUS register) [OP7] 01000 Read the FCCU frozen status flags (see the FCCU_SCFS register) [OP8] 01001 Reserved 01010 Read the NCF status register (see the FCCU_NCF_S register) [OP10] 01011 Reserved 01100 NCF status clear operation in progress (see the FCCU_NCF_S register) [OP12] 01101 Clear the freeze status registers (see the freeze registers) [OP13]

Table continues on the next page...

FCCU_CTRL field descriptions (continued)

Field	Description
	01110 CONFIG to NORMAL FCCU state (configuration timeout) in progress [OP14]
	01111 Clear the operation status (OPS=Idle) [OP15]
	10000 Reserved
	10001 Read the ALARM timer (see the FCCU_XTMR register) [OP17]
	10010 Reserved
	10011 Read the CFG timer (see the FCCU_XTMR register) [OP19]
	10100 Reserved
	10101-11110 Forbidden
	11111 Reserved

78.7.2 Control Key (FCCU_CTRLK)

The FCCU_CTRLK register implements the key access for the operations OP1, OP2 according to the following sequence:

1. Write the key into the FCCU_CTRLK register.
2. Write the FCCU_CTRL register (operations OP1 or OP2).

NOTE

The FCCU_CTRLK and FCCU_CTRL registers must be written with consecutive instructions. Both registers must be written as 32-bit values. Do not use read-modify-write instructions, such as bit field instructions, to modify these registers.

The FCCU_CTRLK register is not readable, a 0000_0000h value is always returned in case of read operation. The key must be written by a word (32 bits) data write operation.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	CTRLK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_CTRLK field descriptions

Field	Description
0–31 CTRLK	Control register key. = 913756AFh: Key for the operation OP1 = 825A132Bh: Key for the operation OP2

FCCU_CTRLK field descriptions (continued)

Field	Description

78.7.3 Configuration (FCCU_CFG)

Defines the global configuration for FCCU.

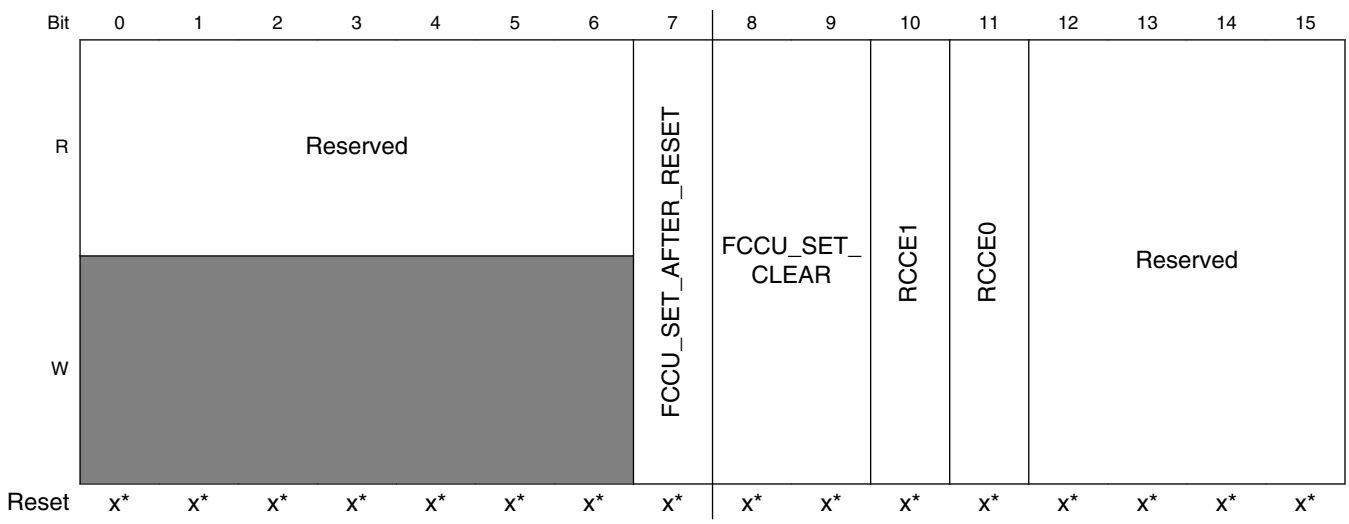
NOTE

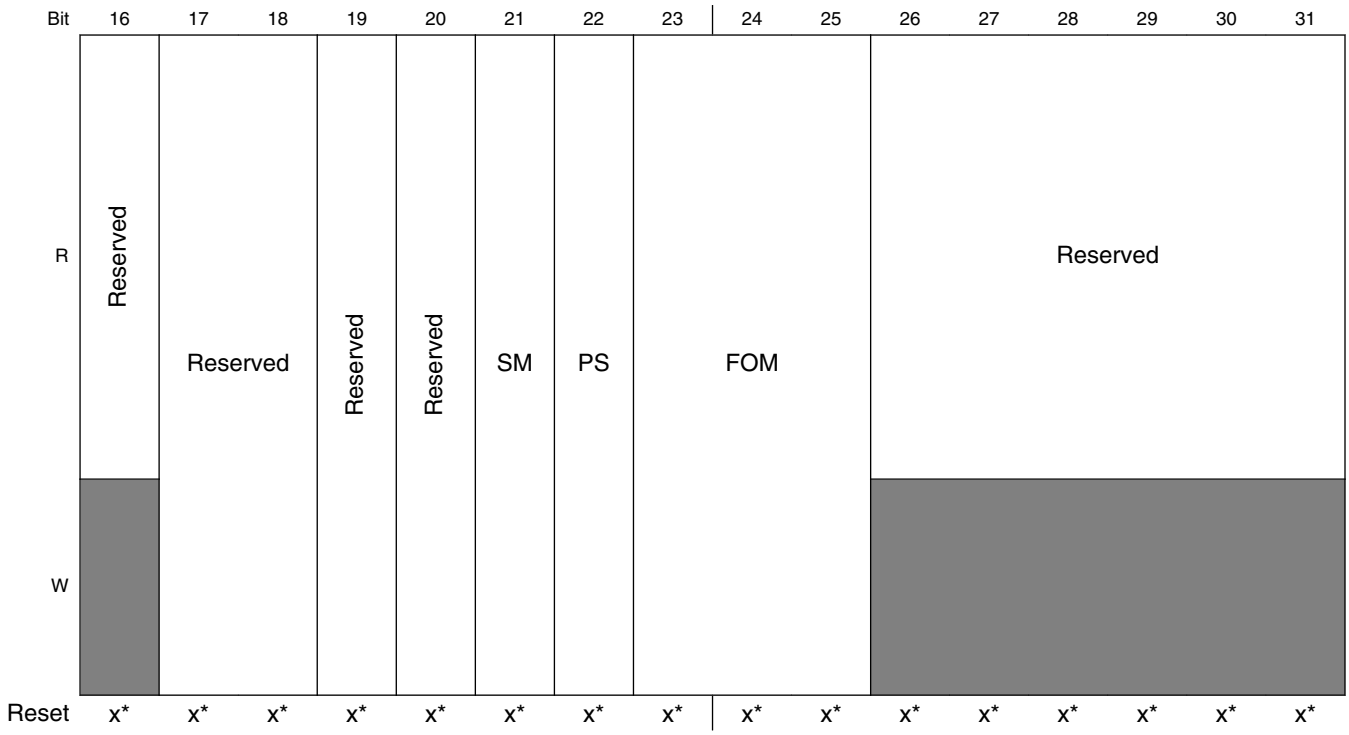
This register is writable only when FCCU is in Configuration state.

NOTE

If you specify a new value for any of the fields in this register that affect the EOUT signals while the EOUT fault-indication timer is running (FCCU is indicating a fault on the EOUT signals), FCCU doesn't use the new settings you specified until after the EOUT fault-indication timer expires (FCCU stops indicating a fault on the EOUT signals).

Address: 0h base + 8h offset = 8h





* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_CFG field descriptions

Field	Description
0–6 Reserved	This field is reserved.
7 FCCU_SET_ AFTER_RESET	Fault-Output (EOUT) Activate For fault-output (EOUT) signaling, controls whether the EOUT signals are active. 0 Inactive (the EOUT signals are in a high-impedance state) 1 Active (the EOUT signals indicate FCCU's condition)
8–9 FCCU_SET_ CLEAR	Fault-Output (EOUT) Control Applies only when the EOUT signals are active (FCCU_SET_AFTER_RESET). Controls whether the fault-output (EOUT) signals are controlled by FCCU's finite state machine (FSM). 00 Controlled by the FSM 01 Always low 10 Controlled by the FSM 11 High until a fault occurs on a channel, regardless of whether that fault is disabled; thereafter, controlled by the FSM. Note: FCCU ignores an attempt to write this value if the fault-output (EOUT) timer is already running.
10 RCCE1	Redundancy Control Checker 1 Enable For the self-checking feature, controls whether FCCU's redundancy control checker 1 (RCC1) is enabled.

Table continues on the next page...

FCCU_CFG field descriptions (continued)

Field	Description
	NOTE: • Always enable any RCCs in a separate register access after activating the EOUT signals (FCCU_SET_AFTER_RESET). • If only one RCC is enabled and a self-check fails, both RCCs assert an interrupt. 0 Disabled 1 Enabled
11 RCCE0	Redundancy Control Checker 0 Enable For the self-checking feature, controls whether FCCU's redundancy control checker 0 (RCC0) is enabled. **NOTE:** • Always enable any RCCs in a separate register access after activating the EOUT signals (FCCU_SET_AFTER_RESET). • If only one RCC is enabled and a self-check fails, both RCCs assert an interrupt. 0 Disabled 1 Enabled
12–15 Reserved	This field is reserved. Always write the reset value to this field.
16 Reserved	This field is reserved.
17–18 Reserved	This field is reserved. Always write the reset value to this field.
19 Reserved	This field is reserved. Always write the reset value to this field.
20 Reserved	This field is reserved. Always write the reset value to this field.
21 SM	Fault-Output (EOUT) Switching Mode For fault-output (EOUT) signaling, controls how quickly the signals switch between faulty and nonfaulty indication. Applies only to Dual-Rail and Time-Switching fault-output modes. 0 Slow: No EOUT frequency violation during the FCCU state transition (Normal to Fault or vice versa, and Configuration to Normal). The indication transition occurs after a maximum delay equal to the duration of the semiperiod of the EOUT frequency. 1 Fast: The indication transition (Normal to Fault or vice versa, and Configuration to Normal) occurs immediately. A pulse with the minimum duration corresponding to the CLKSAFE period can occur in this mode. It implies a frequency violation of the EOUT protocol.
22 PS	Fault-Output (EOUT) Polarity Selection For fault-output (EOUT) signaling, controls the polarity of the signals for fault-output-mode indications that hold the signals low or high (versus toggling them or placing them in a high-impedance state). Applies only to Time-Switching fault-output mode (for faulty and configuration indications) and Bistable fault-output mode (for all indications). 0 For the faulty indication, EOUT1 is high, and EOUT0 is low. 1 For the faulty indication, EOUT1 is low, and EOUT0 is high.
23–25 FOM	Fault-Output (EOUT) Mode For fault-output (EOUT) signaling, controls the protocol of the signaling. **NOTE:** In the test modes, a simple double-stage resynchronization stage is used to resynchronize the EOUT input/outputs with CLKSAFE.

Table continues on the next page...

FCCU_CFG field descriptions (continued)

Field	Description
	000 Dual-Rail (default state) [EOUT[1:0]= outputs] 001 Time-Switching [EOUT[1:0]= output to be used] 010 Bistable 011 Reserved 100 Reserved 101 Test 0 (controlled by the FCCU_EINOUT register; EOUT1 is an output; EOUT0 is an input) 110 Test 1 (controlled by the FCCU_EINOUT register; EOUT1 and EOUT0 are both outputs) 111 Test 2 (controlled by the FCCU_EINOUT register; EOUT1 is an input; EOUT0 is an output)
26–31 Reserved	This field is reserved. Always write the reset value to this field.

78.7.4 Noncritical Fault Configuration (FCCU_NCF_CFG*n*)

Contains the configuration of noncritical faults in terms of fault-recovery management. The configuration depends on the type of signaling of a fault event. Hardware-recoverable faults should be configured only if a previous latching stage captures and holds the physical fault; otherwise, the fault can be lost. All other faults should be configured as software faults.

NOTE

These registers are writable only when FCCU is in Configuration state.

This table shows the noncritical-fault configuration-register channels.

Table 78-14. Noncritical-fault configuration-register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
1Ch	FCCU_NCF_CFG0	NCFC[31:0]
20h	FCCU_NCF_CFG1	NCFC[63:32]

Address: 0h base + 1Ch offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCFCx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_CFG*n* field descriptions

Field	Description
0–31 NCFCx	Noncritical Fault Configuration. Defines the fault recovery mode. Hardware-recoverable faults are self-recovered (status flag clearing and related) if the root cause (input fault) has been removed. Software-recoverable faults are recovered (status flag clearing) by software clearing of the related status flag. See Table 78-14 for register offset to channel number relationship. 0 Hardware-recoverable fault 1 Software-recoverable fault

78.7.5 Noncritical Fault State Configuration (FCCU_NCFS_CFG*n*)

The FCCU_NCFS_CFGx registers contain the configuration of each non-critical fault in terms of fault reaction (short or long functional reset request pulse) when it is the root cause for the FAULT state transition.

NOTE

These registers are writable only when the FCCU is in the CONFIG state.

NOTE

Do not configure the same channel for both a RESET reaction in the FCCU_NCFS_CFGn register and an NMI reaction in the FCCU_NMI_ENn register at the same time.

Table 78-22 shows FCCU configuration register channels.

Table 78-22. FCCU NCFS configuration register channels

Address offset	Register name	Channel range (x)
4Ch	FCCU_NCFS_CFG0	NCFSCx[15:0]
50h	FCCU_NCFS_CFG1	NCFSCx[31:16]
54h	FCCU_NCFS_CFG2	NCFSCx[47:32]
58h	FCCU_NCFS_CFG3	NCFSCx[63:48]
5Ch	FCCU_NCFS_CFG4	NCFSCx[79:64]
60h	FCCU_NCFS_CFG5	NCFSCx[95:80]

Address: 0h base + 4Ch offset + (4d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCFSCx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCFS_CFG*n* field descriptions

Field	Description
0–31 NCFSCx	Non-critical fault state configuration See Table 78-22 for register offset to channel number relationship. 00 No reset reaction 01 Short functional reset request pulse (FAULT state reaction) 10 Long functional reset request pulse (FAULT state reaction) 11 No reset reaction

78.7.6 Noncritical Fault Status (FCCU_NCF_S*n*)

The FCCU_NCF_Sx register contains the latched fault indication collected from the non-critical fault sources. Faults are latched also in the CONFIG state and independently from the enabling or reactions programmed for the NCF.

No reactions are executed until the FCCU moves in the NORMAL state.

FCCU reacts and moves from the NORMAL state into the ALARM state only if the respective enable bit for a fault is set in the FCCU_NCF_Ex register and the respective enable bit for the timeout is set in the FCCU_TOEx register.

FCCU reacts and moves from the NORMAL or ALARM state into the FAULT state if the respective enable bit for a fault is set in the FCCU_NCF_Ex register and the respective enable bit for the timeout is disabled in the FCCU_TOEx register.

FCCU reacts and moves from the ALARM state into the FAULT state if the timeout (FCCU_TO register) is elapsed before recovering from the fault.

The timeout is stopped only when the FCCU returns in the NORMAL state.

The FCCU_NCF_Sx register is encoded respectively into the N2FF_STATUS or A2FF_STATUS register to freeze the entry condition in the FAULT state. The status bits of the FCCU_NCF_Sx register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the proper key into the FCCU_NCFK register.
2. Clear the status (flag) bit NCFSx => the opcode OP12 is automatically set into the FCCU_CTRL.OPR field.
3. Wait for the completion of the operation (FCCU_CTRL.OPS field).
4. Read the FCCU_NCF_Sx register in order to verify the effective deletion and in case of failure to repeat the sequence.

As result of the above sequence, in addition the FAULT interface provides support to clear the external fake FAULT root .

NOTE

There should not be any other operation in between the above steps.

The FCCU moves from the FAULT or ALARM state into the NORMAL state if all the source faults that caused the transition into the FAULT state have been removed (HW recoverable fault) or cleared via SW (SW recoverable fault). In case of nested faults that are not all recovered, the FCCU will remain in the FAULT or ALARM state.

The SW application executes the FCCU_NCF_Sx read operation by the following sequence:

1. Set the OP10 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU_NCF_Sx register.

In case of re-configuration of the FCCU (CONFIG state), before to return in NORMAL state the pending status bits into the FCCU_NCF_Sx must be cleared in order to avoid a false transition in ALARM/FAULT state.

The following faults are ignored:

- to write a wrong key into the FCCU_NCFK register
- to attempt to clear a HW recoverable fault

Table 78-36 shows FCCU NCF status register channels.

Table 78-36. FCCU NCF status register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
80h	FCCU_NCF_S0	NCFS[31:0]
84h	FCCU_NCF_S1	NCFS[63:32]
88h	FCCU_NCF_S2	NCFS[95:64]

Address: 0h base + 80h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NCFSx																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NCF_S*n* field descriptions

Field	Description
0–31 NCFSx	Non-critical fault status. The status bits related to the non-critical fault configured as HW recoverable faults are read-only and the flag is self cleared when the fault source is removed. See Table 78-36 for register offset to channel number relationship. 0 No "non-critical" fault latched 1 "Non-critical" fault latched

78.7.7 Noncritical Fault Key (FCCU_NCFK)

The FCCU_NCFK register implements the key access to clear the status flags of the FCCU_NCF_Sx register.

The status bits of the FCCU_NCF_Sx register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the key into the FCCU_NCFK register.
2. Clear the status (flag) bit NCFSx .

The key must be written for each FCCU_NCF_Sx clear operation.

The FCCU_NCFK register is not readable; a 0000_0000h value is always returned in case of read operation.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																															
W	NCFK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NCFK field descriptions

Field	Description
0–31 NCFK	Non-critical fault key = AB34_98FEh

78.7.8 Noncritical Fault Enable (FCCU_NCF_E*n*)

The FCCU_NCF_En registers enable the fault sources to allow a transition from the NORMAL into the FAULT or ALARM state. In case of fault masking, the respective status bit into the FCCU_NCF_Sn register is set (for debugging purposes), only the reaction is masked.

NOTE

These registers are writable only when the FCCU is in the CONFIG state.

NOTE

Any enabled fault should be programmed to result in a defined action. For example, set up an ALARM IRQ action by programming the IRQ Alarm Enable Register (FCCU_IRQ_ALARM_ENn).

Table 78-45 shows FCCU NCFE enable register channels.

Table 78-45. FCCU NCF enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
94h	FCCU_NCF_E0	NCFE[31:0]
98h	FCCU_NCF_E1	NCFE[63:32]
9Ch	FCCU_NCF_E2	NCFE[95:64]

Address: 0h base + 94h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCFEx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_E*n* field descriptions

Field	Description
0–31 NCFEx	Non-critical fault enable See Table 78-45 for register offset to channel number relationship. 0 No actions following the respective non-critical fault assertion 1 FCCU moves to ALARM or FAULT state

78.7.9 Noncritical Fault Timeout Enable (FCCU_NCF_TOE*n*)

The FCCU_NCF_TOEx registers enable a transition from the NORMAL state into the ALARM state if the respective non-critical fault is enabled (NCFEx and NCFTOEx are set). In case the respective timeout is disabled (NCFTOEx is cleared) and the non-critical fault is enabled (NCFEx is set) the FCCU moves into the FAULT state if the related non-critical fault is asserted. The timer (preset with the timeout value defined by FCCU_TO register) is started when the FCCU moves into the ALARM state. If the fault is not recovered within the timeout the FCCU moves from the ALARM state to the FAULT state.

NOTE

These registers are writable only when the FCCU is in the CONFIG state.

Table 78-53 shows FCCU NCF timeout enable register channels.

Table 78-53. FCCU NCF timeout enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
A4h	FCCU_NCF_TOE0	NCFTOE[31:0]
A8h	FCCU_NCF_TOE1	NCFTOE[63:32]
ACh	FCCU_NCF_TOE2	NCFTOE[95:64]

Address: 0h base + A4h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCFTOEx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_TOE*n* field descriptions

Field	Description
0–31 NCFTOEx	Fault timeout enable See Table 78-53 for register offset to channel number relationship. 0 FCCU moves into the FAULT state if the respective fault is enabled 1 FCCU moves into the ALARM state if the respective fault is enabled

78.7.10 Noncritical Fault Timeout (FCCU_NCF_TO)

Defines the preset value of the timer for the recovery of enabled noncritical faults. Once FCCU enters Alarm state, following the assertion of a noncritical fault that is enabled (NCFEa and NCFTOEa are set), the timer starts the countdown.

If the fault is not recovered within the timeout period, the FCCU moves from the Alarm state to the Fault state. The alarm timeout value should be programmed to be less than FOSU_COUNT, or destructive resets may be generated by FOSU timeouts.

NOTE

This register is writable only when FCCU is in Configuration state.

Address: 0h base + B4h offset = B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TO																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_TO field descriptions

Field	Description
0–31 TO	Noncritical Fault Timeout Timeout = (TO) × $T_{RC16MHz}$

78.7.11 Configuration-State Timeout (FCCU_CFG_TO)

The FCCU_CFG_TO register defines the preset value of the watchdog timer for the recovery from the CONFIG state. Once FCCU enters in CONFIG state, following a SW request (OP1 opcode), the watchdog timer is initialized and starts the countdown if the reset is not asserted.

If the configuration is not completed within the timeout, the FCCU moves automatically from the CONFIG state to the NORMAL state and the default values for all the configuration register is restored. Configuration registers are those registers which can be written only in CONFIG state. The description for a configuration register will contain a NOTE that it is writable only in the CONFIG state. The watchdog timeout is clocked

with CLKSAFE. The default timeout value is 4.096 ms. Longer activation of CONFIG state can lead to resets if a failure is indicated during the time the FCCU is in CONFIG state due to FOSU.

NOTE

The FCCU_CFG_TO register is writable in any state excluding the CONFIG state as follows the execution of the OP1 opcode (NORMAL to CONFIG state) and until the completion of the OP2 opcode (CONFIG to NORMAL state).

In case of watchdog timeout the FCCU_CFG_TO register is not accessible until the OP14 operation (CONFIG to NORMAL) has been completed.

NOTE

When the configuration watchdog timer expires, FCCU changes the value of this register to its reset value.

Address: 0h base + B8h offset = B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																													TO		
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_CFG_TO field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 TO	Configuration timeout Timeout = $T_{RC16MHz} \times 2^{(TO + 10)}$ 000 Timeout = 64 µs ... 111 Timeout = 8.192 ms

78.7.12 IO Control (FCCU_EINOUT)

The FCCU_EINOUT register allows the following operations typically in NORMAL state:

- to control the EOUT[1] output level when the FCCU is configured in "Test1" or "Test0" fault output mode (FCCU_CFG.FOM)

- to control the EOUT[0] output level when the FCCU is configured in "Test1" or "Test2" fault output mode (FCCU_CFG.FOM)
- to observe the EOUT[1:0] signals in input mode

Table 78-63 shows Bi-stable encoding.

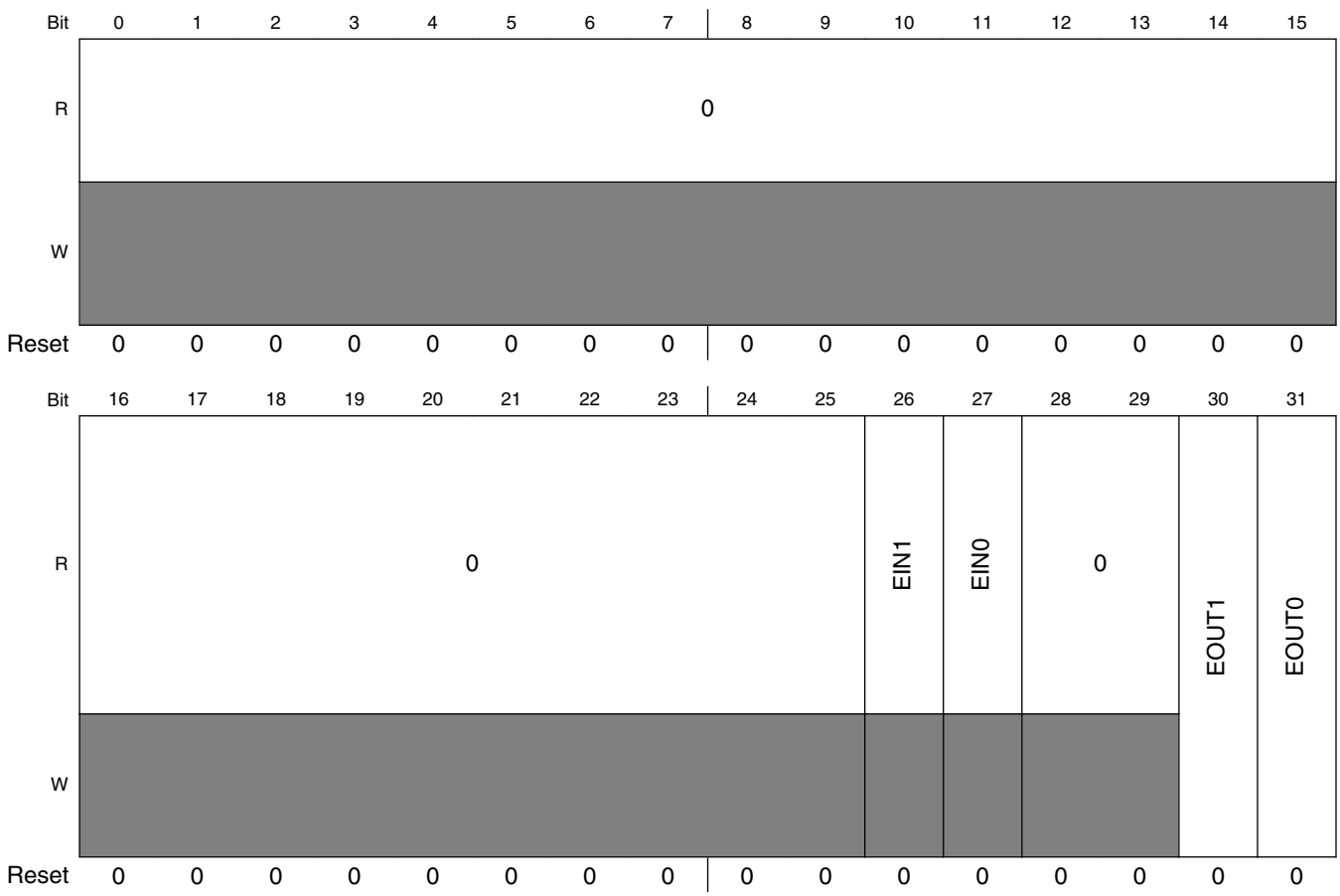
Table 78-63. Bi-stable encoding

Mode = FCCU_CFG.FOM	EOUT[0]	EOUT[1]
Test1	output	output
Test2	output	input
Test0	input	output

NOTE

Due to the resynchronization stage of the EOUT interface, there is a latency of a few CLKSAFE cycles following a write/read operation of the FCCU_EINOUT register.

Address: 0h base + BCh offset = BCh



FCCU_EINOUT field descriptions

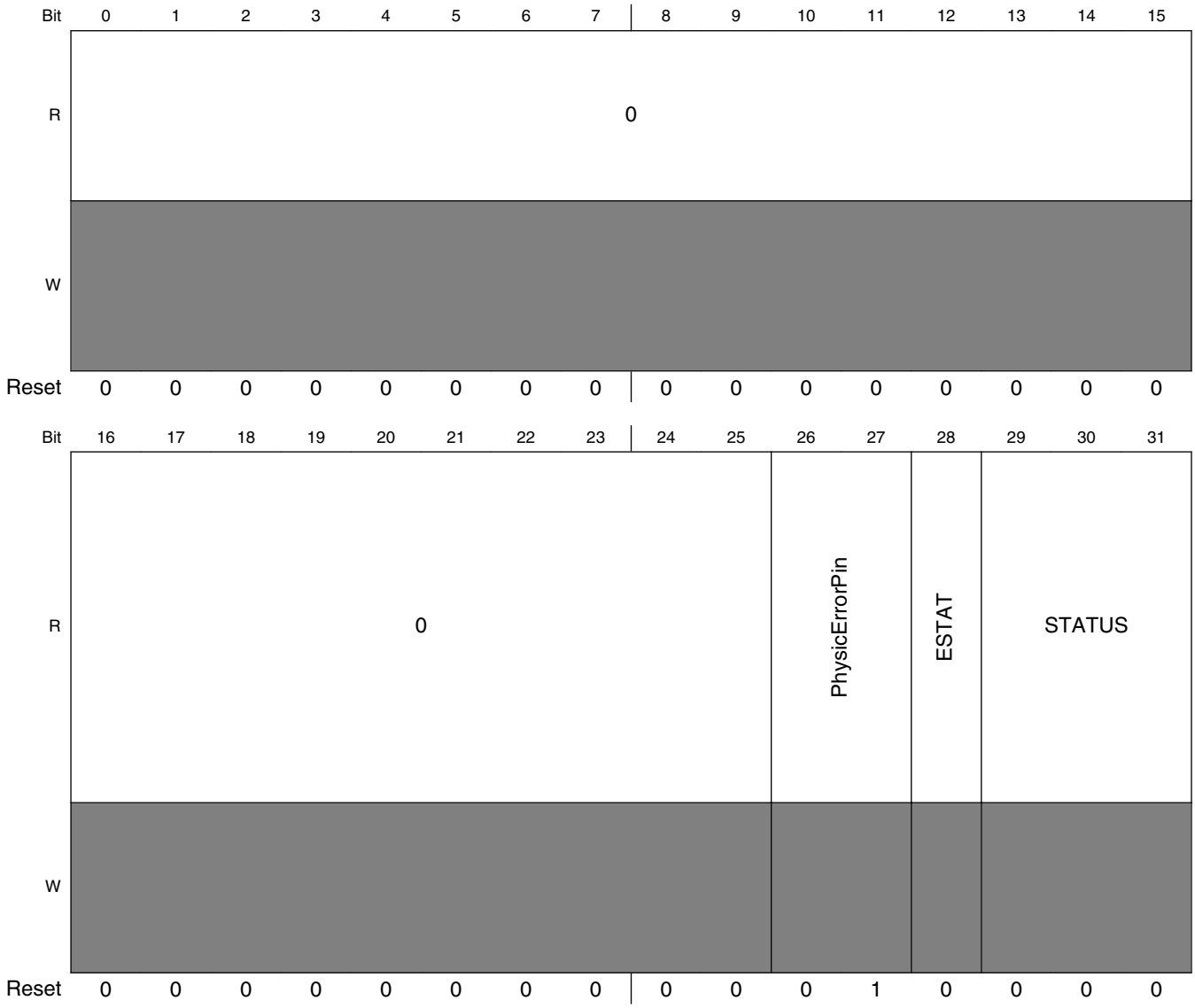
Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 EIN1	Error input 1. The EIN1 captures the respective fccu_ein[1] input signal. While this field is set to zero by reset, a read of this field always returns the logic value on the fccu_ein[1] pin. 0 when fccu_ein[1] = 0 1 when fccu_ein[1] = 1
27 EIN0	Error input 0.The EIN0 captures the fccu_ein[0] input signal. While this field is set to zero by reset, a read of this field always returns the logic value on the fccu_ein[0] pin. 0 when fccu_ein[0] = 0 1 when fccu_ein[0] = 1
28–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 EOUT1	Error out 1 (significant only if the FCCU_CFG.FOM = Test1 or Test0 => EOUT[1] configured in output mode). The EOUT1 set/clear the respective EOUT[1] output signal if FCCU_CFG.FOM = 110 or 101, otherwise it is a "don't-care" value. **NOTE:** When the configuration watchdog timer expires, FCCU changes the value of this field to its reset value. 0 force EOUT[1] = 0 1 force EOUT[1] = 1
31 EOUT0	Error out 0 (significative only if the FCCU_CFG.FOM = Test1 or Test2 => EOUT[0] configured in output mode). The EOUT0 set/clear the respective EOUT[0] output signal if FCCU_CFG.FOM = 110 or 111, otherwise it is a "don't care" value. **NOTE:** When the configuration watchdog timer expires, FCCU changes the value of this field to its reset value. 0 force EOUT[0] = 0 1 force EOUT[0] = 1

78.7.13 Status (FCCU_STAT)

The FCCU_STAT register includes the FCCU status for debugging/test purposes. The FCCU finite state machine operates by the RC oscillator clock asynchronous with the system clock. The FCCU status read operation requires a safe mechanism operated by a HW/SW synchronization sequence. The SW application executes a FCCU status read operation by the following sequence:

1. Set the OP3 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU status (FCCU_STAT register).

Address: 0h base + C0h offset = C0h



FCCU_STAT field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–27 PhysicErrorPin	PhysicErrorPin : Physical Error Pin's state. PhysicErrorPin[0] corresponds to error pin 0 and PhysicErrorPin[1] corresponds to error pin 1. For each of these, the bit values have the following meanings: 0: Error Pin is at logical 0 1: Error Pin is at logical 1 During Fault state, a static or toggling value is observed based on selected protocol. 0 Error Pin is at logical 0 1 Error Pin is at logical 1

Table continues on the next page...

FCCU_STAT field descriptions (continued)

Field	Description
28 ESTAT	Current Error Pin Status 0 The system is operational (OK). 1 The system is in Fault state. This state is taken from the FCCU EOUT logic (that is, as late as possible).
29–31 STATUS	FCCU Status 000 NORMAL state 001 CONFIG state 010 ALARM state 011 FAULT state 100 Reserved 101 Reserved 110 Reserved 111 Reserved

78.7.14 NA Freeze Status (FCCU_N2AF_STATUS)

The N2AF_STATUS register contains a unique code to identify the "non-critical fault" source (fccu_ncf[x]) that caused the state transition from the NORMAL state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

The SW application executes the N2AF_STATUS read operation by the following sequence:

1. Set the OP4 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2AF_STATUS register.

The SW application executes the N2AF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Address: 0h base + C4h offset = C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								NAFS							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_N2AF_STATUS field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 NAFS	Normal to Alarm Frozen Status **NOTE:** This field can be read and written but writing 1 has no effect. 00000000 No transition from NORMAL to ALARM state 00000001 NORMAL to ALARM state transition cause => fccu_ncf[0] fault 00000010 NORMAL to ALARM state transition cause => fccu_ncf[1] fault 00000011 NORMAL to ALARM state transition cause => fccu_ncf[2] fault ... 10000000 NORMAL to ALARM state transition cause => fccu_ncf[127]/ fault 11111111 NORMAL to ALARM state transition cause => multiple fccu_ncf[]/ faults

78.7.15 AF Freeze Status (FCCU_A2FF_STATUS)

The A2FF_STATUS register contains a unique code to identify the timeout trigger (non-critical fault) that caused the state transition from the ALARM state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

The SW application executes the A2FF_STATUS read operation by the following sequence:

1. Set the OP5 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the A2FF_STATUS register.

The SW application executes the FCCU_AFFS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).

All the freeze registers are cleared by this operation.

Address: 0h base + C8h offset = C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																						AF_ SRC		AFFS							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_A2FF_STATUS field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 AF_SRC	Fault source **NOTE:** This field can be read and written but writing 1 has no effect. 00 No fault 01 Reserved 10 "Non-critical" fault 11 Multiple and/or "non-critical" faults
24–31 AFFS	Alarm to Fault Frozen Status **NOTE:** This field can be read and written but writing 1 has no effect. 00h: No transition from ALARM to FAULT state 01h: ALARM to FAULT state transition cause => fccu_ncf[0]/ fault timeout fault 02h: ALARM to FAULT state transition cause => fccu_ncf[1]/ fault timeout fault 03h: ALARM to FAULT state transition cause => fccu_ncf[2]/ fault timeout fault ... 80h: ALARM to FAULT state transition cause => fccu_ncf[127]/ fault timeout fault FFh: ALARM to FAULT state transition cause => multiple fccu_ncf[]/ faults timeout faults

78.7.16 NF Freeze Status (FCCU_N2FF_STATUS)

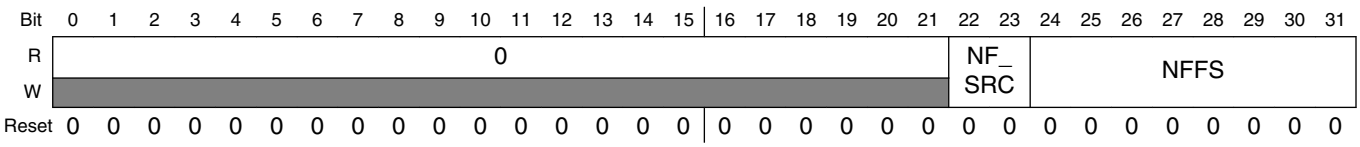
The N2FF_STATUS register contains a unique code to identify the source of the non-critical fault that caused the state transition from the NORMAL state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes. The SW application executes the N2FF_STATUS read operation by the following sequence:

1. Set the OP6 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2FF_STATUS register.

The SW application executes the N2FF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Address: 0h base + CCh offset = CCh



FCCU_N2FF_STATUS field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 NF_SRC	NF_SRC: Fault source **NOTE:** This field can be read and written but writing 1 has no effect. 00 No fault 01 Reserved 10 "Non-critical" fault 11 Multiple "non-critical" faults
24–31 NFFS	Normal to Fault Frozen Status **NOTE:** This field can be read and written but writing 1 has no effect. 00h: No transition from NORMAL to FAULT state 01h: NORMAL to FAULT state transition cause => fccu_ncf[0] fault 02h: NORMAL to FAULT state transition cause => fccu_ncf[1] fault 03h: NORMAL to FAULT state transition cause => fccu_ncf[2] fault ... 80h: NORMAL to FAULT state transition cause => fccu_ncf[127] fault FFh: NORMAL to FAULT state transition cause => multiple fccu_ncf[] faults

78.7.17 FA Freeze Status (FCCU_F2A_STATUS)

The F2AF_STATUS register contains a unique code to identify the source of the non-critical fault (fccu_ncf[x]) that caused the state transition from the FAULT state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

The SW application executes the F2AF_STATUS read operation by the following sequence:

1. Set the OP7 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the F2AF_STATUS register.

The SW application executes the F2AF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Address: 0h base + D0h offset = D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																							FAFS								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_F2A_STATUS field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 FAFS	Fault to Normal Frozen Status **NOTE:** This field can be read and written but writing 1 has no effect. 00h: No transition from FAULT to ALARM state 01h: FAULT to ALARM state transition cause => fccu_ncf[0]/ fault 02h: FAULT to ALARM state transition cause => fccu_ncf[1]/ fault 03h: FAULT to ALARM state transition cause => fccu_ncf[2]/ fault ... 80h: FAULT to ALARM state transition cause => fccu_ncf[127]/ fault FFh: FAULT to ALARM state transition cause => multiple fccu_ncf[] faults

78.7.18 SC Freeze Status (FCCU_SCFS)

The FCCU_SCFS register contains the status of the RCCx checkers. This register is for test/debug purposes.

Two interrupt lines are asserted by the correspondent status bits (FCCU_SCFS.RCCSx).

The SW application executes the FCCU_SCFS read operation by the following sequence:

1. Set the OP8 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU_SCFS register.

The SW application executes the FCCU_SCFS clear operation by the following sequence:

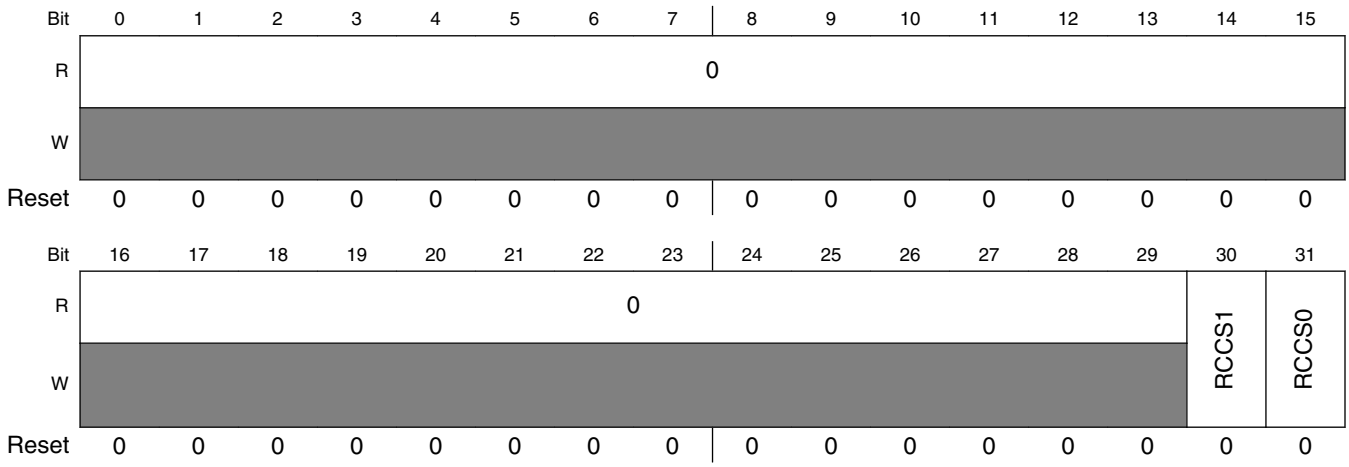
1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). All the freeze registers are cleared by this operation. The RCCx checkers must be disabled

(FCCU_CFG.RCCEx = 00b) during a clear operation otherwise the RCCSx flags are asserted again.

NOTE

The INTC will never attend the second interrupt line since RCCS0 and RCCS1 are always cleared together by the operation above. From SW perspective the INTC should be configured to attend RCCS0 or RCCS1 interrupt line only, not the pair.

Address: 0h base + D4h offset = D4h



FCCU_SCFS field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 RCCS1	RCC1 Status **NOTE:** This field can be read and written but writing 1 has no effect. 0 No miscompares 1 A miscompare has been detected. An interrupt line is asserted.
31 RCCS0	RCC0 Status **NOTE:** This field can be read and written but writing 1 has no effect. 0 No miscompares 1 A miscompare has been detected. The interrupt line is asserted.

78.7.19 Noncritical Fault Fake (FCCU_NCFF)

The FCCU_NCFF register contains a unique code to set a non-critical fault in mutually exclusive mode by the external FAULT interface (signal setting). It allows the SW emulation of the non-critical faults, by the injection of the fault directly in the FAULT root, in order to verify the entire path and reaction. The reaction following a fake non-critical fault cannot be masked. The FCCU_NCFF is a write-only register with a set of codes corresponding to each non-critical fault injection.

Address: 0h base + DCh offset = DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																															
W																										FNCFC						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NCFF field descriptions

Field	Description
0–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 FNCFC	FNCFC/ : Fake non-critical fault code **NOTE:** Only writing to this register, fake fault injection occurs, writing 00 and default value being zero give different results. 00h: Fake non-criticalfault injection at non-criticalfault source 0 01h: Fake non-criticalfault injection at non-criticalfault source 1 02h: Fake non-criticalfault injection at non-criticalfault source 2 ... 7Fh: Fake non-criticalfault injection at non-criticalfault source 127

78.7.20 IRQ Status (FCCU_IRQ_STAT)

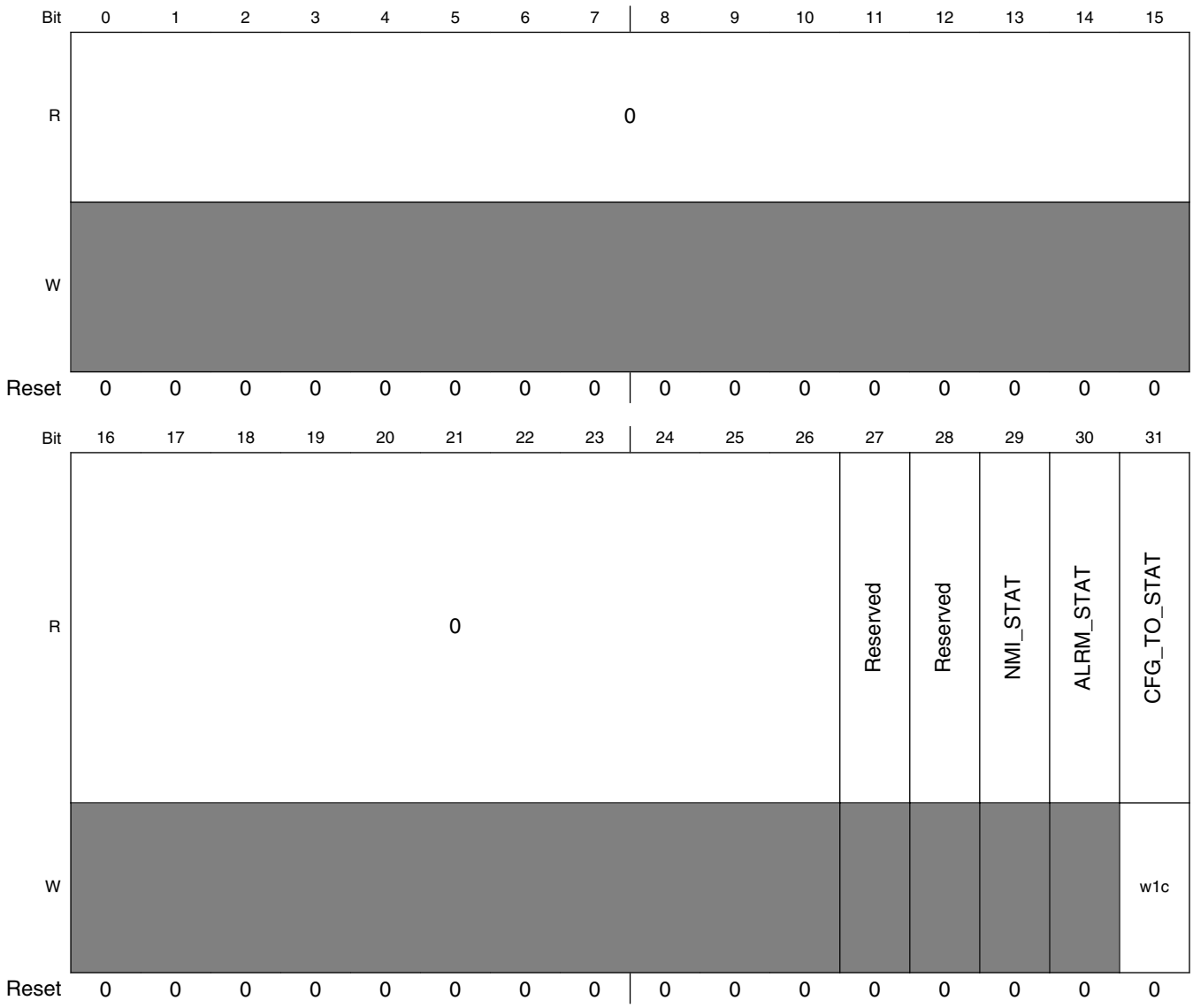
The FCCU_IRQ_STAT register provides the FCCU interrupt status related to the following events:

- Configuration timeout error
- Alarm interrupt
- NMI interrupt

The configuration timeout interrupt is asserted if the CFG_TO_STAT bit of the FCCU_IRQ_STAT register is set and the CFG_TO_IEN bit of the FCCU_IRQ_EN register is also set. It is cleared when a 1 is written to the CFG_TO_STAT bit.

The NMI and ALARM interrupts are asserted and cleared according to the FCCU state. The status bits of the FCCU_IRQ_STAT trace the status of the related interrupt lines.

Address: 0h base + E0h offset = E0h



FCCU_IRQ_STAT field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 Reserved	This field is reserved.

Table continues on the next page...

FCCU_IRQ_STAT field descriptions (continued)

Field	Description
28 Reserved	This field is reserved.
29 NMI_STAT	NMI Interrupt Status 0 NMI interrupt is OFF 1 NMI interrupt is ON
30 ALRM_STAT	Alarm Interrupt Status 0 Alarm interrupt is OFF 1 Alarm interrupt is ON
31 CFG_TO_STAT	Configuration Timeout Status 0 No configuration timeout error 1 Configuration timeout error

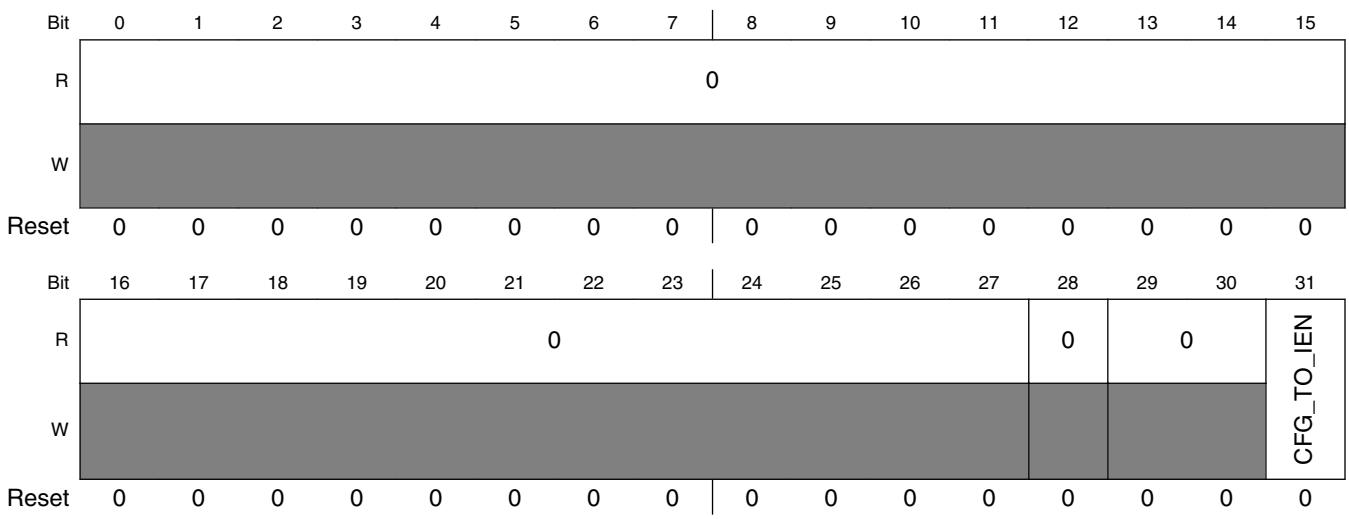
78.7.21 IRQ Enable (FCCU_IRQ_EN)

The FCCU_IRQ_EN register defines the FCCU interrupt enable register related to the following event:

- Configuration timeout error

The configuration timeout interrupt is asserted if the CFG_TO_STAT bit of the FCCU_IRQ_STAT register is set and the CFG_TO_IEN bit of the FCCU_IRQ_EN register is also set.

Address: 0h base + E4h offset = E4h



FCCU_IRQ_EN field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CFG_TO_IEN	Configuration Timeout Interrupt Enable 0 Configuration timeout interrupt disabled 1 Configuration timeout interrupt enabled

78.7.22 XTMR (FCCU_XTMR)

The FCCU_XTMR register contains the read values of the Alarm or Watchdog Timer. These timers are clocked by CLKSAFE.

The SW application executes the timer read operation by the following sequence:

1. Set any of the following operations into the FCCU_CTRL.OPR field:
   - OP17
   - OP19
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU_XTMR register.

Table 78-74 shows Timer state/value.

Table 78-74. Timer state/value

TIMER	CONFIG state	NORMAL state	ALARM state	FAULT state
ALARM	00000000h	Initial value	Running	Idle/End of count
CFG	Running	0001FFFFh	0001FFFFh	0001FFFFh

Address: 0h base + E8h offset = E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	XTMR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_XTMR field descriptions

Field	Description
0–31 XTMR	Alarm/Watchdog/Safe request timer

FCCU_XTMR field descriptions (continued)

Field	Description
	The current timer value is measured in CLKSAFE cycles. These bits can be read by the software.

78.7.23 Mode Controller Status (FCCU_MCS)

Indicates the last four chip modes—as defined by the MC_ME module—and whether FCCU was in Fault state when FCCU captured each mode.

Address: 0h base + ECh offset = ECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VL3	FS3	0		MCS3				VL2	FS2	0		MCS2			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VL1	FS1	0		MCS1				VL0	FS0	0		MCS0			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The MC_ME module changes the value of the register immediately after the chip leaves RESET mode.

FCCU_MCS field descriptions

Field	Description
0 VL3	Valid 3—Indicates whether MCS3 and FS3 are valid. 0 Invalid 1 Valid
1 FS3	Fault Status 3—Indicates whether FCCU was in Fault state when FCCU captured MCS3. **NOTE:** This field might occasionally be inaccurate because the MC_ME module provides the chip modes to FCCU in synchronization with CLKSYS, but FCCU captures the chip modes in synchronization with CLKSAFE. 0 A state other than Fault state 1 Fault state
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 MCS3	Mode Controller State 3—Indicates the fourth most recent chip mode.

Table continues on the next page...

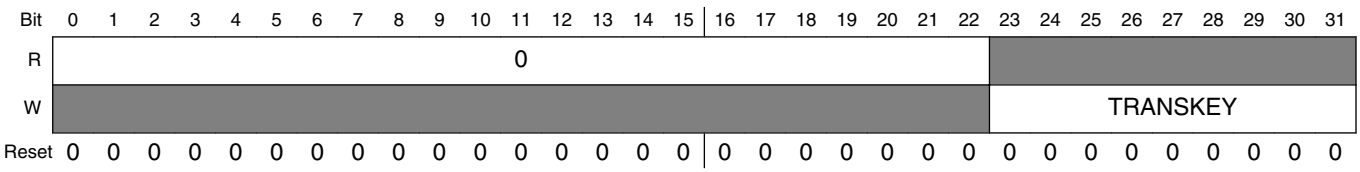
FCCU_MCS field descriptions (continued)

Field	Description
8 VL2	Valid 2—Indicates whether MCS2 and FS2 are valid. 0 Invalid 1 Valid
9 FS2	Fault Status 2—Indicates whether FCCU was in Fault state when FCCU captured MCS2. **NOTE:** This field might occasionally be inaccurate because the MC_ME module provides the chip modes to FCCU in synchronization with CLKSYS, but FCCU captures the chip modes in synchronization with CLKSAFE. 0 A state other than Fault state 1 Fault state
10–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 MCS2	Mode Controller State 2—Indicates the third most recent chip mode.
16 VL1	Valid 1—Indicates whether MCS1 and FS1 are valid. 0 Invalid 1 Valid
17 FS1	Fault Status 1—Indicates whether FCCU was in Fault state when FCCU captured MCS1. **NOTE:** This field might occasionally be inaccurate because the MC_ME module provides the chip modes to FCCU in synchronization with CLKSYS, but FCCU captures the chip modes in synchronization with CLKSAFE. 0 A state other than Fault state 1 Fault state
18–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 MCS1	Mode Controller State 1—Indicates the second most recent chip mode.
24 VL0	Valid 0—Indicates whether MCS0 and FS0 are valid. 0 Invalid 1 Valid
25 FS0	Fault Status 0—Indicates whether FCCU was in Fault state when FCCU captured MCS0. **NOTE:** This field might occasionally be inaccurate because the MC_ME module provides the chip modes to FCCU in synchronization with CLKSYS, but FCCU captures the chip modes in synchronization with CLKSAFE. 0 A state other than Fault state 1 Fault state
26–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 MCS0	Mode Controller State 0—Indicates the most recent (current) chip mode.

78.7.24 Transient Lock (FCCU_TRANS_LOCK)

The FCCU_TRANS register is used for unlocking configuration by writing BCh. This register is available only in supervisor mode. This register is write only.

Address: 0h base + F0h offset = F0h



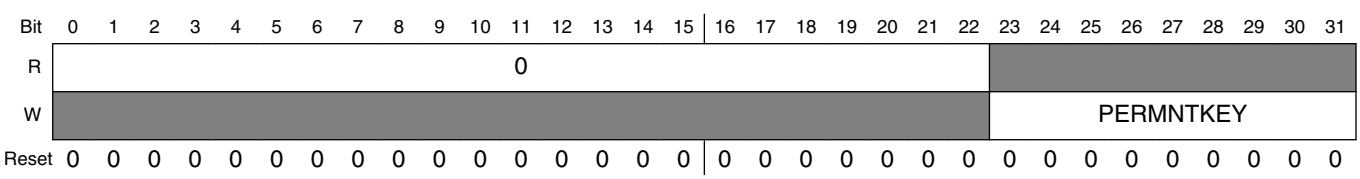
FCCU_TRANS_LOCK field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 TRANSKEY	Transition Unlocking Key value BCh: Configuration Unlocked Any other value:FCCU gets transiently locked again.

78.7.25 Permanent Lock (FCCU_PERMNT_LOCK)

The FCCU_PERMNT_LOCK register is used for locking configuration permanently by writing FFh. This register is available only in supervisor mode. The permanent lock can only be removed by applying a reset (not necessarily power on reset). This register is write only.

Address: 0h base + F4h offset = F4h



FCCU_PERMNT_LOCK field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FCCU_PERMNT_LOCK field descriptions (continued)

Field	Description
23–31 PERMNTKEY	Transition Locking Key value FFh: Configuration Locked Any other value: No Change

78.7.26 Delta T (FCCU_DELTA_T)

The FCCU_DELTA_T register is used for programming the value of delta_T constant, in microseconds.

NOTE

This register can be written only when the FCCU is in CONFIG state.

NOTE

Reserved bits should always be written as all 0's.

Address: 0h base + F8h offset = F8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved		Reserved													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved		DELTA_T													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_DELTA_T field descriptions

Field	Description
0–1 Reserved	This field is reserved.
2–15 Reserved	This field is reserved.
16–17 Reserved	This field is reserved.
18–31 DELTA_T	Controls the minimum amount of time (T_min) that the EOUT fault-indication timer runs according to this equation: T_min = 250 µs + DELTA_T FCCU starts the EOUT fault-indication timer when all of the following are true: • The EOUT fault-indication timer isn't already running. • FCCU enters Fault state as the result of a fault.

Table continues on the next page...

FCCU_DELTA_T field descriptions (continued)

Field	Description

	When the EOUT fault-indication timer is already running and a new fault occurs:
	• When FCCU is in Configuration state: FCCU doesn't restart the EOUT fault-indication timer.
	• When FCCU is in Normal state or Alarm state:
	  • And Alarm state is enabled for the fault (noncritical): FCCU enters (or remains in) Alarm state but doesn't restart the EOUT fault-indication timer.
	  • And Alarm state is disabled for the fault (noncritical) or not applicable: FCCU enters Fault state and restarts the EOUT fault-indication timer.
	• When FCCU is in Fault state: FCCU restarts the EOUT fault-indication timer.
	
	FCCU stops and reinitializes the EOUT fault-indication timer when all of the following are true:
	• T_min has expired.
	• All faults that caused FCCU to enter or remain in Fault state since FCCU started the EOUT fault-indication timer have been cleared, causing FCCU to return to Normal state.
	
	The following values and meanings are for DELTA_T:
	00_0000_0000_0000: 0 µs
	00_0000_0000_0001: 1 µs
	…
	11_1111_1111_1110: 16,382 µs
	11_1111_1111_1111: 16,383 µs
	
	NOTE: The specified values for DELTA_T may vary ±2% due to the post-trimming inaccuracy of the CLKSAFE clocks.

78.7.27 IRQ Alarm Enable (FCCU_IRQ_ALARM_EN*n*)

These registers enable the corresponding IRQ alarm.

Table 78-80 shows FCCU IRQ alarm enable register channels.

Table 78-80. FCCU IRQ alarm enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
0FCh	FCCU_IRQ_ALARM_EN0	IRQENE[31:0]
100h	FCCU_IRQ_ALARM_EN1	IRQENE[63:32]
104h	FCCU_IRQ_ALARM_EN2	IRQENE[95:64]

NOTE

These registers can be written only when the FCCU is in CONFIG state.

Address: 0h base + FCh offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IRQENx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_IRQ_ALARM_EN*n* field descriptions

Field	Description
0–31 IRQENx	IRQ alarm enable See Table 78-80 for register offset to channel number relationship. 0 Alarm is disabled for error source x. 1 Alarm is enabled for error source x.

78.7.28 NMI Enable (FCCU_NMI_EN*n*)

These registers enable the NMI.

NOTE

Do not configure the same channel for both a RESET reaction in the FCCU_NCFS_CFGn register and an NMI reaction in the FCCU_NMI_ENn register at the same time.

Table 78-88 shows FCCU NMI enable register channels.

Table 78-88. FCCU NMI enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
10Ch	FCCU_NMI_EN0	NMIENE[31:0]
110h	FCCU_NMI_EN1	NMIENE[63:32]
114h	FCCU_NMI_EN2	NMIENE[95:64]

NOTE

These registers can be written only when the FCCU is in CONFIG state.

Address: 0h base + 10Ch offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NMIENx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NMI_EN*n* field descriptions

Field	Description
0–31 NMIENx	NMI enable See Table 78-88 for register offset to channel number relationship. 0 NMI is disabled for error (NCF) source x. 1 NMI is enabled for error (NCF) source x.

78.7.29 Fault-Output (EOUT) Signaling Enable (FCCU_EOUT_SIG_EN*n*)

Writable only when FCCU is in Configuration state. Controls whether fault-output (EOUT) signaling is enabled for the associated noncritical fault channels when FCCU is configured for Bistable fault-output mode (FCCU_CFG[FOM]).

Register	Offset	EOUTENx fields	
		Most significant (leftmost)	Least significant (rightmost)
FCCU_EOUT_SIG_EN0	11Ch	EOUTEN31	EOUTEN0
FCCU_EOUT_SIG_EN1	120h	EOUTEN63	EOUTEN32
FCCU_EOUT_SIG_EN2	124h	EOUTEN95	EOUTEN64

Address: 0h base + 11Ch offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EOUTENx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_EOUT_SIG_EN*n* field descriptions

Field	Description
0–31 EOUTENx	Fault-Output (EOUT) Signaling Enable Writable only when FCCU is in Configuration state. Controls whether fault-output (EOUT) signaling is enabled for the associated noncritical fault channel (x) when FCCU is configured for Bistable fault-output mode (FCCU_CFG[FOM]). Fault-output signaling is always enabled for the associated channel, regardless of the value of this field, when FCCU is configured for any other fault-output mode. **NOTE:** To ensure that the FOSU module is aware that fault-output signaling is enabled as a reaction for the associated noncritical fault channel (so that FOSU doesn't mistakenly generate a destructive chip reset), you must set this field to enabled when FCCU is configured for any mode other than Bistable fault-output mode. For the mapping of the EOUTENx fields among registers, see the earlier table in this register description.

FCCU_EOUT_SIG_EN*n* field descriptions (continued)

Field	Description
	0 Disabled 1 Enabled

78.8 FCCU Output Supervision Unit

The FOSU provides a supervision of the primary fault notification path by analyzing FCCU behavior for correctness. It waits for any reaction of the FCCU in a fixed time window after a fault is signaled.

The intention of the FOSU is to provide a secondary fault reaction path in most cases when the FCCU fails but not to needlessly propagate a fault which is already handled by the FCCU in a full chip reset. Only a failed primary fault reaction (that is, FCCU's failure) is a reason for the secondary reaction to take over (and generate a destructive reset request).

There is a 'do nothing' input coming from the FCCU that indicates that the FCCU is programmed for no reaction for ALL FAULTS. It is a "static" input in the sense that it does not change after FCCU configuration. The FOSU masks the incoming faults with the 'do nothing' control from the FCCU, meaning that a fault is not captured by the FOSU if the 'do nothing' signal is asserted, (i.e., a disabled fault). There is no minimum pulse width requirement on the fault indication other than what is required by the technology, which is the same as that of the FCCU. FOSU does not monitor FCCU for the case of faults occurring during CONFIG state. Also, the case of a continuously incoming disabled fault being enabled later, is not monitored.

The FOSU contains a timer with a duration of FOSU_COUNT, driven by CLKSAFE. The timer is initialized and started on any captured, enabled fault. While the timer is running, any subsequent captured fault will neither restart nor reinitialize the timer. The timer is stopped when the FCCU shows any of the following reactions (the FOSU does not check whether the reaction is the configured one for the faults which occurred):

- Reset: Long or short functional reset
- IRQ: NMI or Alarm
- Error out triggered (by FCCU or by SW)

When the timer is stopped, the fault capture logic is cleared in order to ensure that the timer is not restarted due to faults still 'stuck' in the capture logic. The timer will then be restarted by the next new failure indication. When the timer expires, the FOSU's failure

indicator output is asserted after it ensures that the fault is enabled and the static "fccu program to do nothing" signal is deasserted. This is because FCCU uses settings after it exits CONFIG state, even if fault captured before the exit.

The FOSU's failure indicator output is connected to one of the MC_RGM's 'destructive' reset inputs, so its assertion will cause a reset sequence to be initiated starting at PHASE0. The FOSU module is reset with the same reset as is used by the FCCU, which is asserted on power-on, 'destructive', and external resets. When this reset is asserted, the FOSU's capture logic is cleared, its timer is kept stopped and in a non-expired state, and its failure indicator output is deasserted.

NOTE

FOSU is triggered on assertion of enabled fault. In case the triggering fault is disabled, FOSU times-out without reaction. All intermediate faults are masked (not monitored) during this timeout cycle.

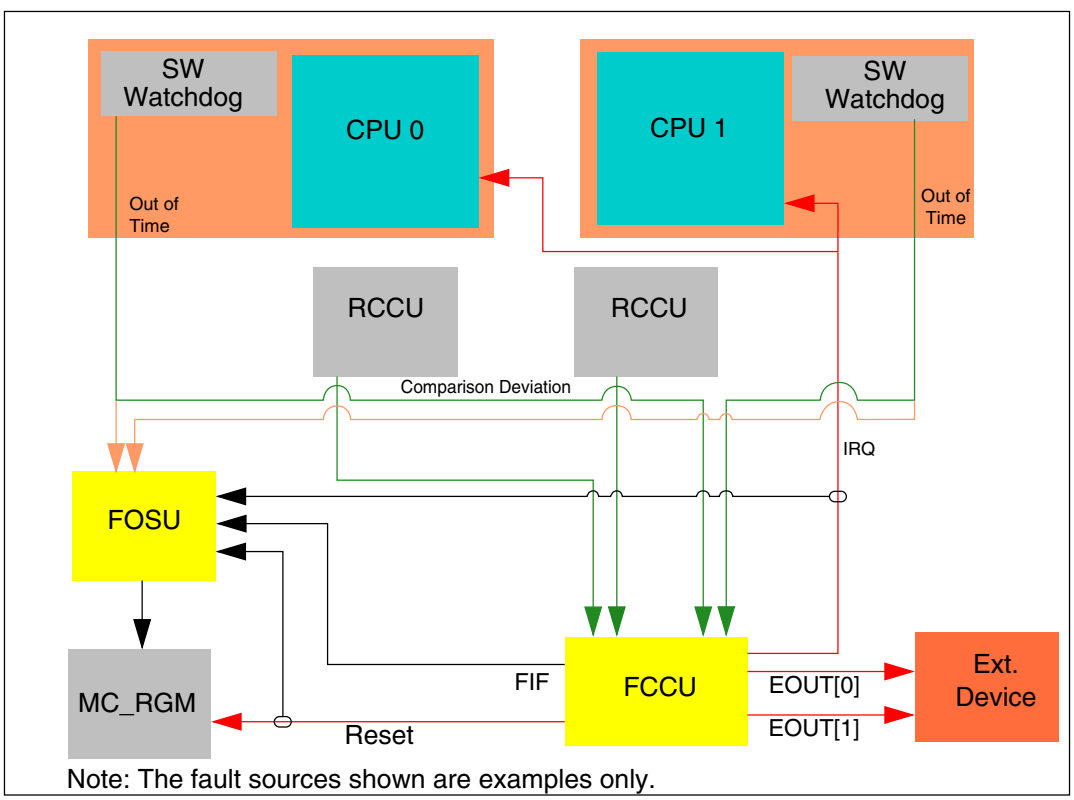


Figure 78-72. FOSU connections to the FCCU and MC_RGM

It is important to remember that there will be no FCCU reaction to a fault while the FCCU is in the CONFIG state. For this reason, the FCCU should not be kept in CONFIG for longer than the FOSU_COUNT duration. Otherwise, there is a risk that an incoming error report causing the FOSU to mistakenly see the FCCU as having failed, and then resets the MCU.

NOTE

FCCU and FOSU timeout counter settings: The FCCU counter value should always be programmed less than the FOSU counter value, FOSU_COUNT, or destructive resets may be generated by an FOSU time-out. This ensures that the FOSU reacts only when the FCCU fails to react to any particular fault.

78.9 Use cases and limitations

78.9.1 Misconfigurations

The following configurations are appropriate:

1. If at least one reaction to fault is enabled by default then the corresponding fault should also be enabled by default.
2. Enabling a fault but disabling all reactions is not a meaningful configuration from safety point of view.
3. Disable the fault and also its reactions. Applicable when a specific fault line is not of interest in a specific application scenario

   Example: FCCU goes into ALARM state due to a failure, but the IRQ is disabled for that failure.
4. If the user programs the FCCU for a higher timeout value than what is hardcoded for FOSU, then FCCU reacts later than what FOSU would be expecting and, as a result, FOSU will generate destructive reset request on its timeout.

78.9.2 Recommendations to configure FCCU

1. After a power on, 'destructive', or long external reset, where both system and FCCU are reset, the following steps could be followed to configure FCCU:
   a. Check and clear any pending fault status
   b. Verify FCCU is in NORMAL state, else repeat step(a) above
   c. Configure FCCU
2. After a short external or any 'functional' reset of the system, arising out of a reset request from FCCU or other sources, the following steps could be followed to reconfigure FCCU:
   a. If active, wait for the Error out T_min to expire

b. Check and clear fault status
c. Error pin moves to "non faulty" state, once fault status is cleared and T_min expires
d. Verify FCCU is in NORMAL state, else repeat step(a) above
e. Read and verify value in FCCU_NCF_Ex
f. Reconfigure FCCU, if necessary

Chapter 79
Self-Test Control Unit (STCU2)

79.1 STCU2 chip-specific module information

This section describes the chip-specific implementation of the STCU2 module.

79.1.1 STCU2 register reset values

The following table shows chip-specific reset values for several STCU2 registers. All other register reset values are shown in Register description.

NOTE

These are the reset values when startup self-test is not configured by DCFs. Otherwise, these will change per the DCF configurations of startup self-test.

Table 79-1. STCU2 register reset values

Register	Reset value
STCU2 Configuration Register (STCU2_CFG)	0000_0000h
STCU2 PLL Configuration Register (STCU2_PLL_CFG)	0000_0000h
STCU2 Watchdog Register Granularity (STCU2_WDG)	0000_FFFFh
STCU2 CRC Expected Status Register (STCU2_CRCE)	0000_0000h
STCU2 CRC Read Status Register (STCU2_CRCR)	FFFF_FFFFh
STCU2 Error FM Register (STCU2_ERR_FM)	0000_0000h
STCU2 LBIST Unrecoverable FM Register (STCU2_LBUFM)	0000_0000h
STCU2 MBIST Unrecoverable FM Low Register (STCU2_MBUFML)	0000_0000h
STCU2 MBIST Unrecoverable FM Medium Register (STCU2_MBUFMM)	0000_0000h
STCU2 MBIST Unrecoverable FM High Register (STCU2_MBUFMH)	0000_0000h
STCU2 LBIST Control Register (STCU2_LB_CTRLn)	0000_0000h
STCU2 LBIST PC Stop Register (STCU2_LB_PCSn)	0000_0000
STCU2 LBIST PRPG Low Register (STCU2_LB_PRPGLn)	FFFF_FFFFh

Table continues on the next page...

Table 79-1. STCU2 register reset values (continued)

Register	Reset value
STCU2 LBIST PRPG High Register (STCU2_LB_PRPGHn)	FFFF_FFFFh
STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELn)	FFFF_FFFFh
STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREHn)	FFFF_FFFFh
STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLn)	0000_0000h
STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRHn)	0000_0000h
STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSWn)	FFFF_FFFFh
STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSWn)	FFFF_FFFFh
STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSWn)	0000_0000h
STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSWn)	0000_0000h
STCU2 MBIST Control Register (STCU2_MB_CTRLn)	0000_0000h

79.1.2 Chip-specific STCU2 registers

Chip-specific implementation of registers are shown in the table below.

Table 79-2. Chip specific registers

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
STCU2 Off-Line LBIST Status Register (STCU2_LBS)	Reserved																												LBS3	LBS2	LBS1	LBS0
STCU2 Off-Line LBIST End Flag Register (STCU2_LBE)	Reserved																												LBE3	LBE2	LBE1	LBE0
STCU2 On-Line LBIST Status Register (STCU2_LBSSW)	Reserved																												LBSSW3	LBSSW2	LBSSW1	LBSSW0
STCU2 On-Line LBIST End Flag Register (STCU2_LBESW)	Reserved																												LBESW3	LBESW2	LBESW1	LBESW0
STCU2 LBIST Unrecoverable FM Register (STCU2_LBUFM)	Reserved																												LBUFM3	LBUFM2	LBUFM1	LBUFM0

Table continues on the next page...

Table 79-2. Chip specific registers (continued)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
STCU2 Off-Line MBIST Status Medium Register (STCU2_MBSM)	Reserved																	MBS46	MBS45	MBS44	MBS43	MBS42	MBS41	MBS40	MBS39	MBS38	MBS37	MBS36	MBS35	MBS34	MBS33	MBS32
STCU2 Off-Line MBIST Status High Register (STCU2_MBSH)	Reserved																															
STCU2 Off-Line MBIST End Flag Medium Register (STCU2_MBEM)	Reserved																	MBE46	MBE45	MBE44	MBE43	MBE42	MBE41	MBE40	MBE39	MBE38	MBE37	MBE36	MBE35	MBE34	MBE33	MBE32
STCU2 Off-Line MBIST End Flag High Register (STCU2_MBEH)	Reserved																															
STCU2 On-Line MBIST Status Medium Register (STCU2_MBSMSW)	Reserved																	MBSSW46	MBSSW45	MBSSW44	MBSSW43	MBSSW42	MBSSW41	MBSSW40	MBSSW39	MBSSW38	MBSSW37	MBSSW36	MBSSW35	MBSSW34	MBSSW33	MBSSW32
STCU2 On-Line MBIST Status High Register (STCU2_MBSHSW)	Reserved																															
STCU2 On-Line MBIST End Flag Medium Register (STCU2_MBEMSW)	Reserved																	MBESSW46	MBESSW45	MBESSW44	MBESSW43	MBESSW42	MBESSW41	MBESSW40	MBESSW39	MBESSW38	MBESSW37	MBESSW36	MBESSW35	MBESSW34	MBESSW33	MBESSW32
STCU2 On-Line MBIST End Flag High Register (STCU2_MBEHSW)	Reserved																															
STCU2 MBIST Unrecoverable FM Medium Register (STCU2_MBUFMM)	Reserved																	MBUFM46	MBUFM45	MBUFM44	MBUFM43	MBUFM42	MBUFM41	MBUFM40	MBUFM39	MBUFM38	MBUFM37	MBUFM36	MBUFM35	MBUFM34	MBUFM33	MBUFM32
STCU2 MBIST Unrecoverable FM High Register (STCU2_MBUFMH)	Reserved																															

79.1.3 MBIST Partitions

The following table describes the mapping between the MBIST number as the STCU2 captures it, the corresponding memory, and the corresponding MBIST control register.

Table 79-3. MBIST mapping

Partition Number	Memory	Corresponding MBIST Control Register
0	FlexCAN_3 MB/Mask (2 of 2)	STCU_MB_CTRL_0
1	FlexCAN_3 MB/Mask (1 of 2)	STCU_MB_CTRL_1
2	FlexCAN_1 MB/Mask (2 of 2)	STCU_MB_CTRL_2
3	FlexCAN_1 MB/Mask (1 of 2)	STCU_MB_CTRL_3
4	BAM ROM	STCU_MB_CTRL_4
5	Core0 I-MEM	STCU_MB_CTRL_5
6	Core0 D-MEM (2 of 2)	STCU_MB_CTRL_6
7	Core0 D-MEM (1 of 2)	STCU_MB_CTRL_7
8	Core0 I-Cache Data (4 of 4)	STCU_MB_CTRL_8
9	Core0 I-Cache Data (3 of 4)	STCU_MB_CTRL_9
10	Core0 I-Cache Data (2 of 4)	STCU_MB_CTRL_10
11	Core0 I-Cache Data (1 of 4)	STCU_MB_CTRL_11
12	Core0 I-Cache Tag	STCU_MB_CTRL_12
13	Core1 I-MEM	STCU_MB_CTRL_13
14	Core1 D-MEM (2 of 2)	STCU_MB_CTRL_14
15	Core1 D-MEM (1 of 2)	STCU_MB_CTRL_15
16	Core1 I-Cache Data (4 of 4)	STCU_MB_CTRL_16
17	Core1 I-Cache Data (3 of 4)	STCU_MB_CTRL_17
18	Core1 I-Cache Data (2 of 4)	STCU_MB_CTRL_18
19	Core1 I-Cache Data (1 of 4)	STCU_MB_CTRL_19
20	Core1 I-Cache Tag	STCU_MB_CTRL_20
21	PRAM0 32kB (Standby RAM)	STCU_MB_CTRL_21
22	PRAM1 32kB	STCU_MB_CTRL_22
23	PRAM3 64kB	STCU_MB_CTRL_23
24	PRAM2 64kB	STCU_MB_CTRL_24
25	PRAM4 64kB	STCU_MB_CTRL_25
26	eDMA TCD	STCU_MB_CTRL_26
27	FlexCAN_2 MB/Mask (2 of 2)	STCU_MB_CTRL_27
28	FlexCAN_2 MB/Mask (1 of 2)	STCU_MB_CTRL_28
29	FlexCAN_0 MB/Mask (2 of 2)	STCU_MB_CTRL_29
30	FlexCAN_0 MB/Mask (1 of 2)	STCU_MB_CTRL_30
31	FEC FIFO	STCU_MB_CTRL_31
32	FEC MIB Counter	STCU_MB_CTRL_32

Table continues on the next page...

Table 79-3. MBIST mapping (continued)

Partition Number	Memory	Corresponding MBIST Control Register
33	eTPU Shared Data Memory	STCU_MB_CTRL_33
34	eTPU Shared Code Memory	STCU_MB_CTRL_34
35	eTPU Nexus Trace (2 of 2)	STCU_MB_CTRL_35
36	eTPU Nexus Trace (1 of 2)	STCU_MB_CTRL_36
37	NAR1 (2 of 2)	STCU_MB_CTRL_37
38	NAR1 (1 of 2)	STCU_MB_CTRL_38
39	NAR0 (2 of 2)	STCU_MB_CTRL_39
40	NAR0 (1 of 2)	STCU_MB_CTRL_40
41	NAR3 (2 of 2)	STCU_MB_CTRL_41
42	NAR3 (1 of 2)	STCU_MB_CTRL_42
43	NAR2 (2 of 2)	STCU_MB_CTRL_43
44	NAR2 (1 of 2)	STCU_MB_CTRL_44
45	Internal Overlay RAM (2 of 2)	STCU_MB_CTRL_45
46	Internal Overlay RAM (1 of 2)	STCU_MB_CTRL_46

79.1.4 MBIST Test Cycles

The following table describes the test cycles for different MBIST partitions on this chip according to test type.

Table 79-4. Test cycles (TCK and MEM clocks) by MBIST partitions

Partition Number	Memory	Size	Full Test		Reduced Test		Auto Test	
			TCKs	MEM Clks	TCKs	MEM Clks	TCKs	MEM Clks
0	FlexCAN_3 MB/Mask (2 of 2)	288 x 52	2244	39695	2112	37293	528	6634
1	FlexCAN_3 MB/Mask (1 of 2)	288 x 52	2244	39695	2112	37293	528	6634
2	FlexCAN_1 MB/Mask (2 of 2)	288 x 52	2244	39695	2112	37293	528	6634
3	FlexCAN_1 MB/Mask (1 of 2)	288 x 52	2244	39695	2112	37293	528	6634
4	BAM ROM	2048 x 32	211	15422	179	10282	NA	NA
5	Core0 I-MEM	2048 x 72	2244	269167	2112	252429	528	36970
6	Core0 D-MEM (2 of 2)	4096 x 40	2244	525247	2112	504333	528	73834
7	Core0 D-MEM (1 of 2)	4096 x 40	2244	525247	2112	504333	528	73834
8	Core0 I-Cache Data (4 of 4)	256 x 72	2244	33903	2112	32013	528	4714
9	Core0 I-Cache Data (3 of 4)	256 x 72	2244	33903	2112	32013	528	4714
10	Core0 I-Cache Data (2 of 4)	256 x 72	2244	33903	2112	32013	528	4714
11	Core0 I-Cache Data (1 of 4)	256 x 72	2244	33903	2112	32013	528	4714
12	Core0 I-Cache Tag	128 x 65	2244	17135	2112	16269	528	2410

Table continues on the next page...

Table 79-4. Test cycles (TCK and MEM clocks) by MBIST partitions (continued)

Partition Number	Memory	Size	Full Test		Reduced Test		Auto Test	
			TCKs	MEM Clks	TCKs	MEM Clks	TCKs	MEM Clks
13	Core1 I-MEM	2048 x 72	2244	269167	2112	252429	528	36970
14	Core1 D-MEM (2 of 2)	4096 x 40	2244	525247	2112	504333	528	73834
15	Core1 D-MEM (1 of 2)	4096 x 40	2244	525247	2112	504333	528	73834
16	Core1 I-Cache Data (4 of 4)	256 x 72	2244	33903	2112	32013	528	4714
17	Core1 I-Cache Data (3 of 4)	256 x 72	2244	33903	2112	32013	528	4714
18	Core1 I-Cache Data (2 of 4)	256 x 72	2244	33903	2112	32013	528	4714
19	Core1 I-Cache Data (1 of 4)	256 x 72	2244	33903	2112	32013	528	4714
20	Core1 I-Cache Tag	128 x 65	2244	17135	2112	16269	528	2410
21	PRAM0 32kB (Standby RAM)	4096 x 72	2244	527087	2112	504333	528	73834
22	PRAM1 32kB	4096 x 72	2244	527087	2112	504333	528	73834
23	PRAM3 64kB	8192 x 72	2244	1035311	2112	1008141	528	147562
24	PRAM2 64kB	8192 x 72	2244	1035311	2112	1008141	528	147562
25	PRAM4 64kB	8192 x 72	2244	1035311	2112	1008141	528	147562
26	eDMA TCD	256 x 72	2244	33903	2112	32013	528	4714
27	FlexCAN_2 MB/Mask (2 of 2)	288 x 52	2244	39695	2112	37293	528	6634
28	FlexCAN_2 MB/Mask (1 of 2)	288 x 52	2244	39695	2112	37293	528	6634
29	FlexCAN_0 MB/Mask (2 of 2)	288 x 52	2244	39695	2112	37293	528	6634
30	FlexCAN_0 MB/Mask (1 of 2)	288 x 52	2244	39695	2112	37293	528	6634
31	FEC FIFO	128 x 44	2244	17135	2112	16269	528	2410
32	FEC MIB Counter	64 x 40	2244	8815	2112	8397	528	1258
33	eTPU Shared Data Memory	1536 x 52	2244	206527	2112	192525	528	30826
34	eTPU Shared Code Memory	6144 x 39	2244	800111	2112	768525	528	122986
35	eTPU Nexus Trace (2 of 2)	32 x 72	2244	4687	2112	4461	528	682
36	eTPU Nexus Trace (1 of 2)	32 x 72	2244	4687	2112	4461	528	682
37	NAR1 (2 of 2)	64 x 64	2244	8815	2112	8397	528	1258
38	NAR1 (1 of 2)	64 x 64	2244	8815	2112	8397	528	1258
39	NAR0 (2 of 2)	64 x 64	2244	8815	2112	8397	528	1258
40	NAR0 (1 of 2)	64 x 64	2244	8815	2112	8397	528	1258
41	NAR3 (2 of 2)	64 x 64	2244	8815	2112	8397	528	1258
42	NAR3 (1 of 2)	64 x 64	2244	8815	2112	8397	528	1258
43	NAR2 (2 of 2)	64 x 64	2244	8815	2112	8397	528	1258
44	NAR2 (1 of 2)	64 x 64	2244	8815	2112	8397	528	1258
45	Internal Overlay RAM (2 of 2)	1024 x 72	2244	135791	2112	126477	528	18538
46	Internal Overlay RAM (1 of 2)	1024 x 72	2244	135791	2112	126477	528	18538

79.1.5 Running Offline MBIST

To run offline MBIST on this chip you must follow the procedure below:

1. Run offline MBIST on the desired memories. If a memory fault occurs (single- or multi-bit), the corresponding bit in the Off-Line MBIST Status Register (STCU_MBSL, STCU_MBSM, or STCU_MBSH) will be set. When offline MBIST is done, a global functional reset is issued.
2. When code starts running on the CPU, check the Off-Line MBIST Status Registers (STCU_MBSL, STCU_MBSM, and STCU_MBSH) for failing BISTs. If a failed BIST execution is found, then you can choose between aborting at this point or getting more information (i.e. whether the fault occurred due to single- or multi-bit error(s)) about the failed BISTs.
3. If you choose to get more information about the failed BISTs, run online MBIST on the failed BISTs. When the online MBIST is done, you can decide on how to proceed based on this more specific information.

79.1.6 Off-line LBIST

The chip supports off-line LBIST operation for all LBIST partitions.

Summary of an off-line LBIST sequence:

1. The user stores off-line self-test parameters in a DCF record in flash memory. The self-test can include multiple LBIST partitions.
    - For off-line self-tests, LBIST execution on individual LBIST partitions occurs serially. The chip does not support LBIST execution on individual LBIST partitions in parallel.
    - An off-line LBIST can run with either the 16 MHz IRC or with PLL0 or PLL1.
    - For an off-line self-test, the STCU2 can enable and program PLL0 or PLL1 using STCU2 DCFs.
2. The user resets the chip by cycling power or asserting the RESET pin.
3. The STCU2 loads the stored off-line LBIST parameters from the DCF record via the SSCM interface.
4. The off-line LBIST executes and ends.
5. Hardware issues a functional reset. The STCU2 is not affected by this reset.

6. The user verifies the STCU2 result registers related to LBIST and FCCU channels corresponding to STCU2 faults.

79.1.7 On-line LBIST

The chip supports on-line LBIST operation on all LBIST partitions.

NOTE

When LBIST execution begins, the embedded flash memory must be in an idle state; program or erase operations must not be active.

Summary of an on-line LBIST sequence:

1. The user initiates an on-line LBIST by programming STCU2 memory-mapped registers.
   - An on-line LBIST is able to test only one partition at a time.
   - An on-line LBIST can run with the functional operating frequencies. The maximum-frequency clock for the LBIST partition is derived from PLL0 or PLL1.
   - The user can initiate a sequence of more than one on-line LBIST. Multiple on-line LBISTs can execute serially, with each self-test run on an individual partition.
2. The on-line LBIST executes and ends.
3. After the completion of the LBIST on-line self-test sequence, hardware issues a functional reset. The STCU2 is not affected by this reset.
4. The user verifies the STCU2 result registers related to LBIST and FCCU channels corresponding to STCU2 faults.

79.1.8 LBIST Partitions

The following table describes the LBIST partitions on this chip.

Table 79-5. LBIST partitions

Lake	Partition	IP Module
0	0	CAN 1

Table continues on the next page...

Table 79-5. LBIST partitions (continued)

Lake	Partition	IP Module
0	0	CAN 3
0	0	DMAMUX_3
0	0	DSPI_1
0	0	DSPI_3
0	0	DSPI_M1
0	0	SENT_1
0	0	CMU_0
0	0	CMU_1
0	0	CMU_2
0	0	CMU_3
0	0	CRC 1
0	0	LINFlexD_1
0	0	LINFlexD_m1
0	0	LINFlexD_3
0	0	eMIOS 1
0	0	AIPS1
0	0	INTC
0	0	Checker_DMA
0	0	Checker_Core_0
1	1	DSPI_0
1	1	DSPI_M0
1	1	DSPI_2
1	1	DSPI_4
1	1	PLLDIG
1	1	CMU_9
1	1	eMIOS 0
1	1	AXBS
1	1	PRAMC
1	1	Main Core_0
1	1	Main Core_1
1	1	DMA
1	2	CAN 0
1	2	CAN 2
1	2	eTPU
1	2	BAM
1	2	LFAST
1	2	DMAMUX_0
1	2	DMAMUX_1
1	2	DMAMUX_2

Table continues on the next page...

Table 79-5. LBIST partitions (continued)

Lake	Partition	IP Module
1	2	DTS
1	2	JDC
1	2	MC_ME
1	2	MC_PCU
1	2	SENT_0
1	2	TDM
1	2	WKPU
1	2	PIT0
1	2	PIT1
1	2	CMU_6
1	2	CMU_4
1	2	CMU_5
1	2	CMU_7
1	2	CMU_8
1	2	CRC 0
1	2	JTAGM
1	2	LINFlexD_0
1	2	LINFlexD_m0
1	2	LINFlexD_2
1	2	SDADC
1	2	SIPI
1	2	SIUL2
1	2	DECFILTER 0
1	2	DECFILTER 1
1	2	REACM2
1	2	IAHBG
1	2	IGF
1	2	BCTU
1	2	SEMA42
1	2	STM0
1	2	STM1
1	2	SWT0
1	2	SWT1
1	2	AIPS0
1	2	FEC
1	2	PCM
1	2	SWT3
1	3	SPU
1	3	MEMU

Table continues on the next page...

Table 79-5. LBIST partitions (continued)

Lake	Partition	IP Module
1	3	NAR
1	3	CMU_12
1	3	FCCU
1	3	FOSU
1	3	Flash
1	3	SMPU
1	3	Flash Memory Controller

79.1.9 LBIST Configuration

The following table shows the configurations the user should set to run LBIST on a specific partition as well as the expected MISR and coverage values at the end of the run.

NOTE

When running LBIST the Reserved bit DCF_SELFTEST_CONFIG_IPS_1[0] has to be set to "1".

The expected MISR values in the table assume that the UTEST DCF records are not modified from the default reset values except for the above.

Table 79-6. LBIST Configuration by Partition and Expected MISR

Partition	Clock Config	PRPG seed	MISR seed	Number of Patterns	Number of Shifts	Expected MISR	Coverage
L0	1	3fffffffffff	ffffffffffffffff	2,150	63	64'h18F2A1F5FDC31B19	90%
L1	1	ffffffffffff		2,000	48	64'hac001775d8e3627d	90%
W0	1	ffffffffffffffff		2,100	55	64'h765D7C3D4A32B225	90%
W1	1	3fffffffffff		2,750	62	64'hB489FD55EED3E3B7	90%

79.1.9.1 LBIST online self-test

For LBIST online self-test, the user must program the STCU_LBRMSW[LBRMSWn] field corresponding to the LBIST partition to "1" to generate a global functional reset at the end of the test:

- LBRMSW0: corresponds to LBIST L0

- LBRMSW1: corresponds to LBIST L1
- LBRMSW2: corresponds to LBIST W0
- LBRMSW3: corresponds to LBIST W1

79.1.10 AUTOLOCK_VALUE for register write access via STCU2_SKC

Register write-access watchdog timer explains that access to STCU2 registers via the STCU2_SKC security-key mechanism times out after a number of STCU2 clock cycles equal to AUTOLOCK_VALUE. On this chip, AUTOLOCK_VALUE is 1023.

79.1.11 Wait time for writing to the online registers

When writing to the online registers, the user must wait for the expiration of the offline key. The time the user should wait is (IPS_CLK)*4096 clock periods. If the IPS_CLK is 66.5 Mhz then the user must wait (1/66.5)*4096 = 61.6 µs.

79.1.12 On-Line reset generation (only MBIST)

The On-Line execution of the MBIST does not generate any reset from STCU2 at the end of the execution. This is because the software application has to manage all the possible consequences related to the write operations performed by the MBIST execution other than preventing any read/write operation during the Self-Test execution. If necessary, the software must also restore the memory content (only for RAMs) at the end of the Self-Test execution.

79.1.13 On-Line reset generation (LBIST enabled)

The LBRMSW register should be configured to generate a global functional reset after LBIST online self-test. It is not recommended to use a dedicated reset due to random data injected by the LBIST Controller.

NOTE

If an ABORT condition is detected by STCU2 during the On-Line Self-Test execution, independently of the value of the LBRMSW register, only the related reset line is pulsed to prevent any potential dangerous system level operations.

79.2 Introduction

Self Test Control Unit (STCU2) is a comprehensive programmable hardware module that controls the self test sequence applied both during the Off-Line and/or On-Line conditions. It is able to manage by hardware the device's Logic Built-In Self Test (LBIST) and the SRAM/ROM Built-In Self Test (MBIST) blocks. The STCU2 includes the SSCM DCF bus to load the Self-test parameters (L/MBIST scheduling activity, LBIST setup, Unrecoverable/Recoverable faults management, and MISR expected values, PLL management, and so on) from flash memory during the Off-Line Self-Test phase. This interface is able only to write the configuration parameters and start the Self-Test execution once after the STCU2 global reset has been applied. The register access by software is granted by an IPS interface with the purpose to check the results of the Off-Line Self-Test but also to load/check the execution of the On-Line Self-Test.

The STCU2 includes:

- A programmable Watchdog timer (WDG) to check the Self-Test operations (both LBIST and MBIST) have been completed within the assigned time slot or the STCU2 RUN or BYPASS bits have been programmed before Watchdog time-out
- The PLL direct control during the Off-Line Self-Test sequence
- Interrupt lines enabled only during the On-Line Self-Test to simplify the software handling of Self-Test execution end
- Two sets of Result registers to collect the L/MBIST result during Off/On-Line Self-Test separately

To increase the security capability of this module, a different Security Key sequence is also required during Off/On-Line Self-Test in order to unlock the write access to the STCU2 registers. The STCU2 also implements two power-saving mechanisms to switch off:

- The global STCU2 clock after Self-Test sequence is completed
- The register ITF clock when the WDG time-out is detected after a Double Security Key sequence is applied during the application run (available only when IPS write access is enabled otherwise the clock is off).

79.3 Main features

- Double registers interface (SSCM DCF with priority over IPS)

- SSCM DCF write one time register interface
- IPS read and write (depends on SSCM enable parameters and STCU2_CFG.WRP bit) register interface
- Programmable scheduler for LBIST/MBIST execution
- LBIST sequential execution
- MBIST concurrent/sequential execution
- On-Line hardware Self-Test sequence abort capability
- On-Line software Self-Test sequence abort capability
- Programmable Internal clock prescaler to reduce Internal and MBIST/LBIST TCK clocks
- Programmable PLL direct control during Off-Line Self-test sequence
- PLL Lock signal monitoring during Off/On-Line Self Test
- L/MBIST dedicated On-Line interrupt source generation
- Programmable WDG Timer for Self-Check M/LBIST Time Execution
- Hard Coded WDG time-out to flag off-line initialization faults
- Hard Coded WDG time-out to Auto-lock STCU2 write access after Double-Key sequence
- LBIST and SRAM/ROM MBIST Off-Line status flags
- STCU2 internal errors Off-Line status flag
- LBIST and SRAM/ROM MBIST On-Line status flags
- STCU2 internal errors On-Line Status flag
- On-Line hardware abort status flag
- On-Line software abort status flag
- LBIST and SRAM/ROM MBIST Off/On-Line programmable failure severity - unrecoverable fault/recoverable fault (UF/RF)
- STCU2 internal errors Off-On/Line programmable failure severity - unrecoverable fault/recoverable fault (UF/RF)
- STCU2 errors (UF/RF) lines for FCCU signaling
- Redundant UF/RF generation logic to improve reliability

- FCCU UF/RF faults injection mechanism
- LBIST MISRE registers access to perform redundant check by software
- Self-Test Bypass capability
- Global write register's protection mechanism based on Double Security Key
- STCU2 Global Auto Power-saving when Self-Test is completed, bypass mode is selected or WDT time-out is detected
- STCU2 ITF/WDG clock wake-up mechanism when software application writes Double Security Key sequence (available when STCU2_CFG.WRP bit is active)
- STCU2 ITF/WDG Auto Power-saving when Hard Coded WDG time-out is detected after software application writes Double Security Key sequence (available when STCU2_CFG.WRP bit is active)

79.4 Block diagram

The top level diagram of the STCU2 module is given in Figure 79-1.

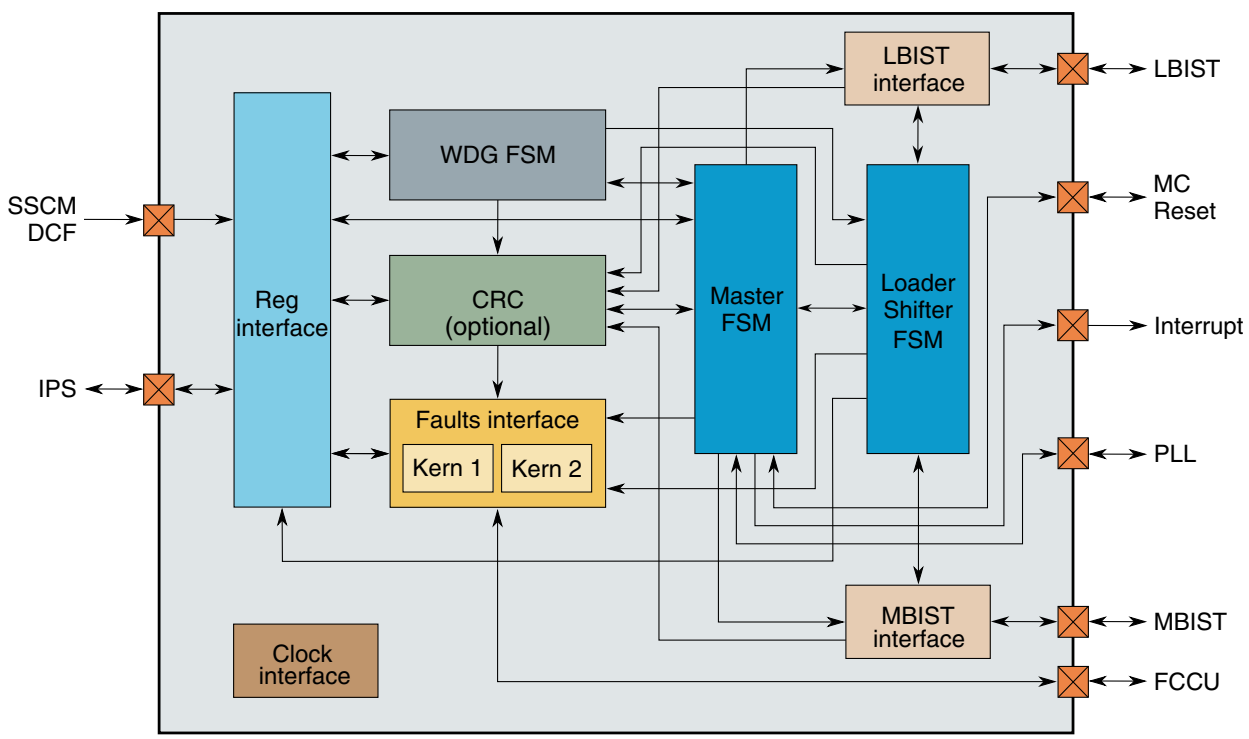


Figure 79-1. STCU2 block diagram

The STCU2 module includes the following sub-modules:

- ***Reg interface***: This module includes the registers, the security key logic, the IPS and SSCM DCF bus interface
- ***Fault interface***: This module collects the fault conditions related to L/MBIST execution and STCU2 internal failures and, depending on the UF/RF configuration of each one of them, sets the global STCU2 UF or RF flag. It also manages the UF/RF fault lines and the Set/Clear injection mechanism provided by the FCCU. To improve the intrinsic reliability of this critical logic, the generation logic is duplicated.
- ***Clock Interface***: This module manages the internal and the L/MBIST TCK clock prescaler, the internal Clock Gating Power Saving and the wake-up clock feature.
- ***WDG FSM***: This module includes two different Watchdogs. The first one is Hard Coded and is used to Auto Lock the STCU2 access forcing a reset condition on the Double Security Key registers while the second one has a double functionality. After a reset event initializes the STCU2, it is used as Hard Coded Watchdog time-out while during L/MBIST run it is used as programmable Watchdog timer to check the L/MBIST have been completed in the assigned L/MBIST time slot.
- ***STCU2 Master FSM***: This module includes the main FSM of the STCU2 used to coordinate and schedule all the operations performed during the Self-Test Sequence.
- ***STCU2 Load Shifter FSM***: This module includes the common shifter register and the related state machine used to program the L/MBIST registers and read back the data to be checked at the end of each Test operation.
- ***LBIST interface***: This module includes the interface between the device's LBIST engines and the STCU2 controllers.
- ***MBIST interface***: This module includes the interface between the MBIST controller and STCU2 controllers.

79.5 IPS bus interface

The IPS bus interface is a slave bus used for configuration purposes via CPU. The following bus read operations (contiguous byte enables) are supported:

- Word (32 bits) data read operations to any registers
- Low and high half-words (16 bits, data[31:16] or data[15:0]) data read operations to any registers

- Byte(8 bits, data[31:24] or data[23:16] or data[15:8] or data[7:0]) data read operations to any registers
- Any other operation (free byte enables or other operations) has to be avoided.

The same operations are also supported in write mode only in the following scenario:

- The Self-Test sequence parameters are loaded before Off-Line Self-test is executed
- The IPS access on a specific register is allowed by software only when the bit STCU2_CFG.WRP is cleared
- In all the other conditions the write operations are not supported and all the STCU2 registers can be only read. Only exceptions are the following read/write bits: STCU2_CFG.WRP, STCU2_ERR_STAT.UFSF and STCU2_ERR_STAT.RFSF. See related register description for additional details

The STCU2 module generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral
- Any write/read operation different from byte/halfword/word (free byte enables or other operations) on each register
- Any write operation on Double Security Key register applying a wrong sequence of keys (the two write operations cannot be interleaved with other access to STCU2 registers)
- Any write operation performed on a register when the Double Security keys have not been applied
- Any write operation performed on registers when the STCU2_CFG.WRP bit is set and the access is performed through software (IPS interface)
- Any write operation performed through software (IPS interface) on Read/On-Line_Write/Off-Line registers
- Any write operation performed off-line on registers in which writing operation is allowed only in the On-Line Self-Test phase when STCU2_CFG[WRP] = 1
- Any write operation performed on Read Only registers

The registers of the STCU2 module are accessible (read/write) in each access mode: user, supervisor or test.

In case there are write operations on bits marked as reserved, the transfer error is not generated.

NOTE

See the chip-specific STCU2 information for the wait time necessary to write to the online registers.

79.6 Functional description

79.6.1 FSM description

The module has three state machines that work together: Master State Machine, Loader/Shifter State Machine, and the Watchdog State Machine. Basically the Master State Machine is the core unit of the STCU2 module. It coordinates all the Self Test operations and the other State Machines. The Loader/Shifter State Machine is used to program the MBIST and the LBIST parameters and to retrieve the related results depending on the parameters stored into the STCU2 registers and under the control of the Master State Machine. The Watchdog State Machine evaluates all the schedule time for MBIST and LBIST and the time-out in case of wrong STCU2 programming.

79.6.2 Reset management

Assertion of the STCU2 reset input signal initializes the STCU2 module status, forcing the Off-Line Self-Test condition and cleaning up all the registers and state machine of the module.

The reset signals generated by the STCU2 module at the end of the Self-Test execution phase depends on the Self-Test conditions and on the content of the LBRMSW register.

79.6.2.1 Off-Line reset generation

At the end of the Off-Line Self-Test execution the STCU2 raises the global functional reset.

79.6.2.2 On-Line reset generation

See the chip-specific STCU2 information for details about On-Line reset generation.

79.6.3 Built-in self-test scheduling

The STCU2 module has been designed to be very flexible allowing to program the parallel/serial execution of the MBIST or LBIST depending on the power/timing/ coverage constraints. The limitation in the programming flexibility is described in Design implementation information.

The mechanism used to provide this flexibility is a linked list where the starting pointer is the bitfield STCU2_CFG.PTR. The first LBIST is mapped on 00h, the second on 01h, and so on. The first MBIST is mapped to the address 10h, the second to 11h and so on. The additional pointers are in the bitfield STCU2_LB_CTRL[X].PTR for LBIST[X] (where X is the selected LBIST) and STCU2_MB_CTRL[Y].PTR for MBIST (where Y is the selected MBIST) and have to be filled depending on the selected sequence of run. The additional bitfield STCU2_MB_CTRL.CSM provide the flexibility to run concurrently or sequentially the chosen set of the MBIST or to close the linked list setting the NIL pointer. For more details, see STCU2 LBIST Control Register (STCU2_LB_CTRL*n*) and STCU2 MBIST Control Register (STCU2_MB_CTRL*n*).

NOTE

Program operation should ensure that no flash write/erase operations are ongoing when online LBIST is triggered. Otherwise, a flash segment may be corrupted.

79.6.4 ABORT management

The STCU2 module provides two mechanisms to perform the On-Line Self-Test execution abort. When the abort is detected in both cases, the On-Line Self-Test operation stops running and the status of the currently running MBIST or LBIST is saved into the related registers to provide intermediate results that might be useful in case of debug or in case of concurrent run. In case of LBIST run, see Reset management, for more details about reset specific management when abort is detected.The following sections describe the differences between these two modalities.

79.6.4.1 Hardware ABORT management

The STCU2 enters the hardware abort condition as a consequence of a Functional reset which causes Self-Test execution to abort .

When hardware abort is detected the bit ABORTHW of the STCU2_ERR_STAT register is set. This flag allows the software to diagnose what has happened during the On-Line Self-Test execution run.

79.6.4.2 Software ABORT management

The STCU2 enters in the abort condition because the software set the bit RUNSW_ABORT in the STCU2_RUNSW register. In order to allow the software to understand when STCU2 has completed the abort sequence, this bit will be maintained active until the abort sequence is completed. At that time the STCU2 will clear this bit as described in STCU2 Run Software Register (STCU2_RUNSW).

When software abort is raised the bit ABORTSW of the STCU2_ERR_STAT register is set. Even if the abort condition is forced by software operation, this flag allows to diagnose what has happened during the On-Line Self-Test execution run.

79.6.5 PLL interface

The STCU2 module is able to directly control the PLL during the Off-Line Self-Test operations depending on the status of the STCU2_RUN[MBPLLEN] and STCU2_RUN[LBPLLEN].

When STCU2_RUN[RUN] is set to start the Off-Line Self-test operations, the STCU2 takes control of the PLL and changes the PLL clock configurations parameters according to the values programmed into the STCU2_PLL_CFG register.

When the STCU2_RUN[MBPLLEN] is active and the MBIST is currently selected, the STCU2 waits for the PLL lock signal. As soon as the lock happens, the STCU2 forces the clock to switch to the PLL output clock. This allows the MBIST to proceed at the PLL output frequency. At the end of the MBIST run the clock source is reverted to the original one.

Same behavior will be applied when STCU2_RUN[LBPLLEN] is active and LBIST is currently selected.

79.6.6 Interrupt interface

Interrupt sources:

- Triggered by MBIST (enabled by programming STCU2_RUNSW[MBIE] = 1)
- Triggered by LBIST (enabled by programming STCU2_RUNSW[LBIE] = 1)

As described in the STCU2 Run Software Register (STCU2_RUNSW) description, the interrupts are generated at the end of any concurrent run. This means that in case of multiple sequential run in the same session, the number of interrupts generated is equivalent to the number of the sequential session providing in such a way a fine granularity with respect to the completed run.

The LBIFLG bit and MBIFLG bit in the STCU2 Interrupt Flag Register (STCU2_INT_FLG) register flag interrupt conditions and allow the clearing of interrupt requests. Interrupt bits are cleared by writing the related flag bit to 1 to enable the next interrupt request.

79.6.7 FCCU interface

The FCCU interface is the hardware flag mechanism towards the system, to indicate the occurrence of an Unrecoverable fault and/or a Recoverable fault failure during the Self Test sequence.

To diagnose physical defects on the two fault signals, a fault injection mechanism is also provided. In this case, the FCCU interface allows the user application to check the integrity of the UF and RF connection lines between the STCU2 and the FCCU. Refer to the description of FCCU fault injection mechanism to understand how the UF/RF set/clear mechanism works.

79.6.8 Watchdogs

The STCU2 implements three different watchdogs to ensure that operations are finished in time.

79.6.8.1 Initialization watchdog timer

The STCU2 has two initialization operating modes: program the self-test sequence and run it (Safety mode) or bypass completely the self-test sequence setting the bit STCU2_CFG.BYP. In case there are faults during the initialization phase preventing the selection of one of these two operating modes, a hard-coded watchdog time-out flags the wrong behavior.

79.6.8.2 Built-in self-test watchdog timer

The LBIST and MBIST execution time has to be configured as described in STCU2 Watchdog Register Granularity (STCU2_WDG), to account for the overall execution time of the Self-Test sequence. In case the selected LBISTs or MBISTs are not yet completed during the Off-line Self-Test assigned time, the current LBISTs or MBISTs execution is interrupted and a failure is flagged into STCU2_ERR_STAT.WDTO and STCU2_MBEL/M/H or STCU2_LBE registers while in case of On-Line Self-Test into STCU2_ERR_STAT.WDTOSW and STCU2_MBELSW/MSW/HSW or STCU2_LBESW.

In case of multiple sequential run in the same On-Line/Off-Line session and the time-out happens in the middle of a sequential run, the next sequential run will be skipped and the execution ends with the current updated status of the registers reported above.

In all the cases, the STCU2 reset generation is managed in one of the modes described in Reset management.

79.6.8.3 Register write-access watchdog timer

The access on the STCU2 registers during the self-test configuration phase (both online/offline) is protected by a key mechanism to prevent any unwanted access, as described in the STCU2_SKC register description. In addition to this malicious access protection feature, there is a hardware watchdog timer that, after AUTOLOCK_VALUE STCU2 clock cycles, lock the access requiring to apply again the STCU2_SKC keys. AUTOLOCK_VALUE is given in the chip-specific STCU2 information.

NOTE

Each DCF record takes 8 STCU2 clock cycles for write completion. The maximum number of DCF records that can be executed before the watchdog timer must be refreshed is AUTOLOCK Value divided by 8. To refresh the hardware watchdog timer, the user must write key 2 into the STCU2_SKC register.

This feature is particularly useful in case STCU2_CFG[WRP] is 0 during the software self-test configuration since in this case the software application might enable the write access to the STCU2 registers.

79.6.9 SSCM interface

The SSCM interface is used to program the STCU2's configuration parameters without the CPU intervention after a reset trigger event initializes the STCU2. This bus interface has the priority over the IPS.

79.7 Register description

During Offline SELFTEST, once the STCU2 registers are configured they cannot be overridden.

During Online SELFTEST, once the STCU2 registers are configured they cannot be overridden via IPS till the selftest is complete.

The STCU2 registers are listed in this section.

STCU2 memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	STCU2 Run Register (STCU2_RUN)	32	R/W	0000_0000h	79.7.1/3378
4	STCU2 Run Software Register (STCU2_RUNSW)	32	R/W	0000_0000h	79.7.2/3379
8	STCU2 SK Code Register (STCU2_SKC)	32	W	0000_0000h	79.7.3/3381
C	STCU2 Configuration Register (STCU2_CFG)	32	R/W	See section	79.7.4/3382
10	STCU2 PLL Configuration Register (STCU2_PLL_CFG)	32	R/W	See section	79.7.5/3385
14	STCU2 Watchdog Register Granularity (STCU2_WDG)	32	R/W	See section	79.7.6/3386
18	STCU2 Interrupt Flag Register (STCU2_INT_FLG)	32	w1c	0000_0000h	79.7.7/3388
24	STCU2 Error Register (STCU2_ERR_STAT)	32	R/W	0000_0000h	79.7.8/3389
28	STCU2 Error FM Register (STCU2_ERR_FM)	32	R/W	See section	79.7.9/3392
2C	STCU2 Off-Line LBIST Status Register (STCU2_LBS)	32	R	0000_0000h	79.7.10/ 3393
30	STCU2 Off-Line LBIST End Flag Register (STCU2_LBE)	32	R	0000_0000h	79.7.11/ 3394
34	STCU2 On-Line LBIST Status Register (STCU2_LBSSW)	32	R	0000_0000h	79.7.12/ 3395
38	STCU2 On-Line LBIST End Flag Register (STCU2_LBESW)	32	R	0000_0000h	79.7.13/ 3397
3C	STCU2 On-Line LBIST Reset Management (STCU2_LBRMSW)	32	R/W	See section	79.7.14/ 3398
40	STCU2 LBIST Unrecoverable FM Register (STCU2_LBUFM)	32	R/W	See section	79.7.15/ 3399
44	STCU2 Off-Line MBIST Status Low Register (STCU2_MBSL)	32	R	0000_0000h	79.7.16/ 3400

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
48	STCU2 Off-Line MBIST Status Medium Register (STCU2_MBSM)	32	R	0000_0000h	79.7.17/ 3405
4C	STCU2 Off-Line MBIST Status High Register (STCU2_MBSH)	32	R	0000_0000h	79.7.18/ 3408
50	STCU2 Off-Line MBIST End Flag Low Register (STCU2_MBEL)	32	R	0000_0000h	79.7.19/ 3409
54	STCU2 Off-Line MBIST End Flag Medium Register (STCU2_MBEM)	32	R	0000_0000h	79.7.20/ 3412
58	STCU2 Off-Line MBIST End Flag High Register (STCU2_MBEH)	32	R	0000_0000h	79.7.21/ 3414
5C	STCU2 On-Line MBIST Status Low Register (STCU2_MBSLSW)	32	R	0000_0000h	79.7.22/ 3415
60	STCU2 On-Line MBIST Status Medium Register (STCU2_MBSMSW)	32	R	0000_0000h	79.7.23/ 3419
64	STCU2 On-Line MBIST Status High Register (STCU2_MBSHSW)	32	R	0000_0000h	79.7.24/ 3421
68	STCU2 On-Line MBIST End Flag Low Register (STCU2_MBELSW)	32	R	0000_0000h	79.7.25/ 3421
6C	STCU2 On-Line MBIST End Flag Medium Register (STCU2_MBEMSW)	32	R	0000_0000h	79.7.26/ 3425
70	STCU2 On-Line MBIST End Flag High Register (STCU2_MBEHSW)	32	R	0000_0000h	79.7.27/ 3427
74	STCU2 MBIST Unrecoverable FM Low Register (STCU2_MBUFML)	32	R/W	See section	79.7.28/ 3428
78	STCU2 MBIST Unrecoverable FM Medium Register (STCU2_MBUFMM)	32	R/W	See section	79.7.29/ 3431
7C	STCU2 MBIST Unrecoverable FM High Register (STCU2_MBUFMH)	32	R/W	See section	79.7.30/ 3434
100	STCU2 LBIST Control Register (STCU2_LB_CTRL0)	32	R/W	See section	79.7.31/ 3434
104	STCU2 LBIST PC Stop Register (STCU2_LB_PCS0)	32	R/W	See section	79.7.32/ 3437
108	STCU2 LBIST PRPG Low Register (STCU2_LB_PRPGL0)	32	R/W	See section	79.7.33/ 3438
10C	STCU2 LBIST PRPG High Register (STCU2_LB_PRPGH0)	32	R/W	See section	79.7.34/ 3439
110	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL0)	32	R/W	See section	79.7.35/ 3439
114	STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH0)	32	R/W	See section	79.7.36/ 3440
118	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL0)	32	R	See section	79.7.37/ 3441
11C	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH0)	32	R	See section	79.7.38/ 3441

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
120	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW0)	32	R/W	See section	79.7.39/ 3442
124	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW0)	32	R/W	See section	79.7.40/ 3443
128	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW0)	32	R	See section	79.7.41/ 3444
12C	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW0)	32	R	See section	79.7.42/ 3444
140	STCU2 LBIST Control Register (STCU2_LB_CTRL1)	32	R/W	See section	79.7.31/ 3434
144	STCU2 LBIST PC Stop Register (STCU2_LB_PCS1)	32	R/W	See section	79.7.32/ 3437
148	STCU2 LBIST PRPG Low Register (STCU2_LB_PRPGL1)	32	R/W	See section	79.7.33/ 3438
14C	STCU2 LBIST PRPG High Register (STCU2_LB_PRPGH1)	32	R/W	See section	79.7.34/ 3439
150	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL1)	32	R/W	See section	79.7.35/ 3439
158	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL1)	32	R	See section	79.7.37/ 3441
15C	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH1)	32	R	See section	79.7.38/ 3441
160	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW1)	32	R/W	See section	79.7.39/ 3442
164	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW1)	32	R/W	See section	79.7.40/ 3443
168	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW1)	32	R	See section	79.7.41/ 3444
16C	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW1)	32	R	See section	79.7.42/ 3444
180	STCU2 LBIST Control Register (STCU2_LB_CTRL2)	32	R/W	See section	79.7.31/ 3434
184	STCU2 LBIST PC Stop Register (STCU2_LB_PCS2)	32	R/W	See section	79.7.32/ 3437
188	STCU2 LBIST PRPG Low Register (STCU2_LB_PRPGL2)	32	R/W	See section	79.7.33/ 3438
18C	STCU2 LBIST PRPG High Register (STCU2_LB_PRPGH2)	32	R/W	See section	79.7.34/ 3439
190	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL2)	32	R/W	See section	79.7.35/ 3439
198	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL2)	32	R	See section	79.7.37/ 3441
19C	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH2)	32	R	See section	79.7.38/ 3441

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1A0	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW2)	32	R/W	See section	79.7.39/ 3442
1A4	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW2)	32	R/W	See section	79.7.40/ 3443
1A8	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW2)	32	R	See section	79.7.41/ 3444
1AC	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW2)	32	R	See section	79.7.42/ 3444
1C0	STCU2 LBIST Control Register (STCU2_LB_CTRL3)	32	R/W	See section	79.7.31/ 3434
1C4	STCU2 LBIST PC Stop Register (STCU2_LB_PCS3)	32	R/W	See section	79.7.32/ 3437
1C8	STCU2 LBIST PRPG Low Register (STCU2_LB_PRPGL3)	32	R/W	See section	79.7.33/ 3438
1CC	STCU2 LBIST PRPG High Register (STCU2_LB_PRPGH3)	32	R/W	See section	79.7.34/ 3439
1D0	STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL3)	32	R/W	See section	79.7.35/ 3439
1D8	STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL3)	32	R	See section	79.7.37/ 3441
1DC	STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH3)	32	R	See section	79.7.38/ 3441
1E0	STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW3)	32	R/W	See section	79.7.39/ 3442
1E4	STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW3)	32	R/W	See section	79.7.40/ 3443
1E8	STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW3)	32	R	See section	79.7.41/ 3444
1EC	STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW3)	32	R	See section	79.7.42/ 3444
600	STCU2 MBIST Control Register (STCU2_MB_CTRL0)	32	R/W	See section	79.7.43/ 3445
604	STCU2 MBIST Control Register (STCU2_MB_CTRL1)	32	R/W	See section	79.7.43/ 3445
608	STCU2 MBIST Control Register (STCU2_MB_CTRL2)	32	R/W	See section	79.7.43/ 3445
60C	STCU2 MBIST Control Register (STCU2_MB_CTRL3)	32	R/W	See section	79.7.43/ 3445
610	STCU2 MBIST Control Register (STCU2_MB_CTRL4)	32	R/W	See section	79.7.43/ 3445
614	STCU2 MBIST Control Register (STCU2_MB_CTRL5)	32	R/W	See section	79.7.43/ 3445
618	STCU2 MBIST Control Register (STCU2_MB_CTRL6)	32	R/W	See section	79.7.43/ 3445

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
61C	STCU2 MBIST Control Register (STCU2_MB_CTRL7)	32	R/W	See section	79.7.43/ 3445
620	STCU2 MBIST Control Register (STCU2_MB_CTRL8)	32	R/W	See section	79.7.43/ 3445
624	STCU2 MBIST Control Register (STCU2_MB_CTRL9)	32	R/W	See section	79.7.43/ 3445
628	STCU2 MBIST Control Register (STCU2_MB_CTRL10)	32	R/W	See section	79.7.43/ 3445
62C	STCU2 MBIST Control Register (STCU2_MB_CTRL11)	32	R/W	See section	79.7.43/ 3445
630	STCU2 MBIST Control Register (STCU2_MB_CTRL12)	32	R/W	See section	79.7.43/ 3445
634	STCU2 MBIST Control Register (STCU2_MB_CTRL13)	32	R/W	See section	79.7.43/ 3445
638	STCU2 MBIST Control Register (STCU2_MB_CTRL14)	32	R/W	See section	79.7.43/ 3445
63C	STCU2 MBIST Control Register (STCU2_MB_CTRL15)	32	R/W	See section	79.7.43/ 3445
640	STCU2 MBIST Control Register (STCU2_MB_CTRL16)	32	R/W	See section	79.7.43/ 3445
644	STCU2 MBIST Control Register (STCU2_MB_CTRL17)	32	R/W	See section	79.7.43/ 3445
648	STCU2 MBIST Control Register (STCU2_MB_CTRL18)	32	R/W	See section	79.7.43/ 3445
64C	STCU2 MBIST Control Register (STCU2_MB_CTRL19)	32	R/W	See section	79.7.43/ 3445
650	STCU2 MBIST Control Register (STCU2_MB_CTRL20)	32	R/W	See section	79.7.43/ 3445
654	STCU2 MBIST Control Register (STCU2_MB_CTRL21)	32	R/W	See section	79.7.43/ 3445
658	STCU2 MBIST Control Register (STCU2_MB_CTRL22)	32	R/W	See section	79.7.43/ 3445
65C	STCU2 MBIST Control Register (STCU2_MB_CTRL23)	32	R/W	See section	79.7.43/ 3445
660	STCU2 MBIST Control Register (STCU2_MB_CTRL24)	32	R/W	See section	79.7.43/ 3445
664	STCU2 MBIST Control Register (STCU2_MB_CTRL25)	32	R/W	See section	79.7.43/ 3445
668	STCU2 MBIST Control Register (STCU2_MB_CTRL26)	32	R/W	See section	79.7.43/ 3445
66C	STCU2 MBIST Control Register (STCU2_MB_CTRL27)	32	R/W	See section	79.7.43/ 3445
670	STCU2 MBIST Control Register (STCU2_MB_CTRL28)	32	R/W	See section	79.7.43/ 3445

Table continues on the next page...

STCU2 memory map (continued)

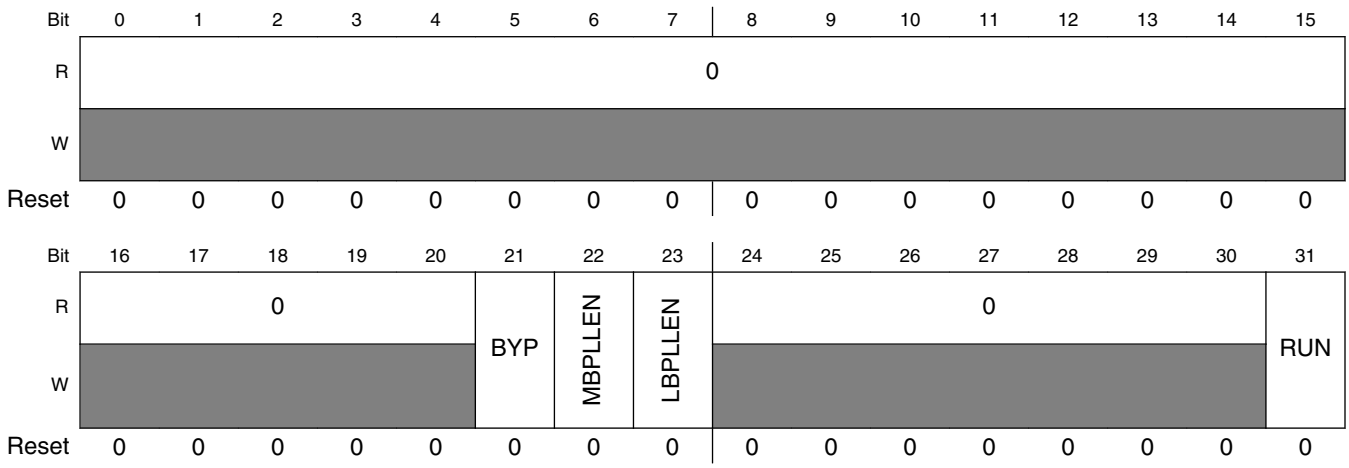
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
674	STCU2 MBIST Control Register (STCU2_MB_CTRL29)	32	R/W	See section	79.7.43/ 3445
678	STCU2 MBIST Control Register (STCU2_MB_CTRL30)	32	R/W	See section	79.7.43/ 3445
67C	STCU2 MBIST Control Register (STCU2_MB_CTRL31)	32	R/W	See section	79.7.43/ 3445
680	STCU2 MBIST Control Register (STCU2_MB_CTRL32)	32	R/W	See section	79.7.43/ 3445
684	STCU2 MBIST Control Register (STCU2_MB_CTRL33)	32	R/W	See section	79.7.43/ 3445
688	STCU2 MBIST Control Register (STCU2_MB_CTRL34)	32	R/W	See section	79.7.43/ 3445
68C	STCU2 MBIST Control Register (STCU2_MB_CTRL35)	32	R/W	See section	79.7.43/ 3445
690	STCU2 MBIST Control Register (STCU2_MB_CTRL36)	32	R/W	See section	79.7.43/ 3445
694	STCU2 MBIST Control Register (STCU2_MB_CTRL37)	32	R/W	See section	79.7.43/ 3445
698	STCU2 MBIST Control Register (STCU2_MB_CTRL38)	32	R/W	See section	79.7.43/ 3445
69C	STCU2 MBIST Control Register (STCU2_MB_CTRL39)	32	R/W	See section	79.7.43/ 3445
6A0	STCU2 MBIST Control Register (STCU2_MB_CTRL40)	32	R/W	See section	79.7.43/ 3445
6A4	STCU2 MBIST Control Register (STCU2_MB_CTRL41)	32	R/W	See section	79.7.43/ 3445
6A8	STCU2 MBIST Control Register (STCU2_MB_CTRL42)	32	R/W	See section	79.7.43/ 3445
6AC	STCU2 MBIST Control Register (STCU2_MB_CTRL43)	32	R/W	See section	79.7.43/ 3445
6B0	STCU2 MBIST Control Register (STCU2_MB_CTRL44)	32	R/W	See section	79.7.43/ 3445
6B4	STCU2 MBIST Control Register (STCU2_MB_CTRL45)	32	R/W	See section	79.7.43/ 3445
6B8	STCU2 MBIST Control Register (STCU2_MB_CTRL46)	32	R/W	See section	79.7.43/ 3445

79.7.1 STCU2 Run Register (STCU2_RUN)

The STCU2_RUN register defines the RUN bit to start the off-line self testing procedure.

The R/W fields in this register are readable at any time. However, you can write to these fields only when Off-line Self-Test phase is still active.

Address: 0h base + 0h offset = 0h



STCU2_RUN field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 BYP	Bypass mode 0 Off-Line Self-Test is active. In case the RUN bit into the STCU2_RUN register is not set before the Hard-Coded WDG Time-out, the WDTO error is generated into the STCU2_ERR_STAT register. 1 Off-Line Self-Test is completely bypassed and the access to STCU2 is locked until the STCU2_SKC register is written according to the request access mode.
22 MBPLLEN	Off-Line MBIST with PLL Enabled 0 Off-Line MBIST is executed without using the on-chip PLL. 1 Off-Line MBIST is executed enabling the on-chip PLL control interface selecting the parameters defined into STCU2_PLL_CFG register.
23 LBPLLEN	Off-Line LBIST with PLL Enabled 0 Off-Line LBIST is executed without using the on-chip PLL. 1 Off-Line LBIST is executed enabling the on-chip PLL control interface selecting the parameters defined into STCU2_PLL_CFG register.
24–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RUN	RUN: RUN The RUN bit is automatically cleared by the STCU2 when the off-line self testing procedure has been completed. 0 Idle 1 Off-Line Self testing procedure is running

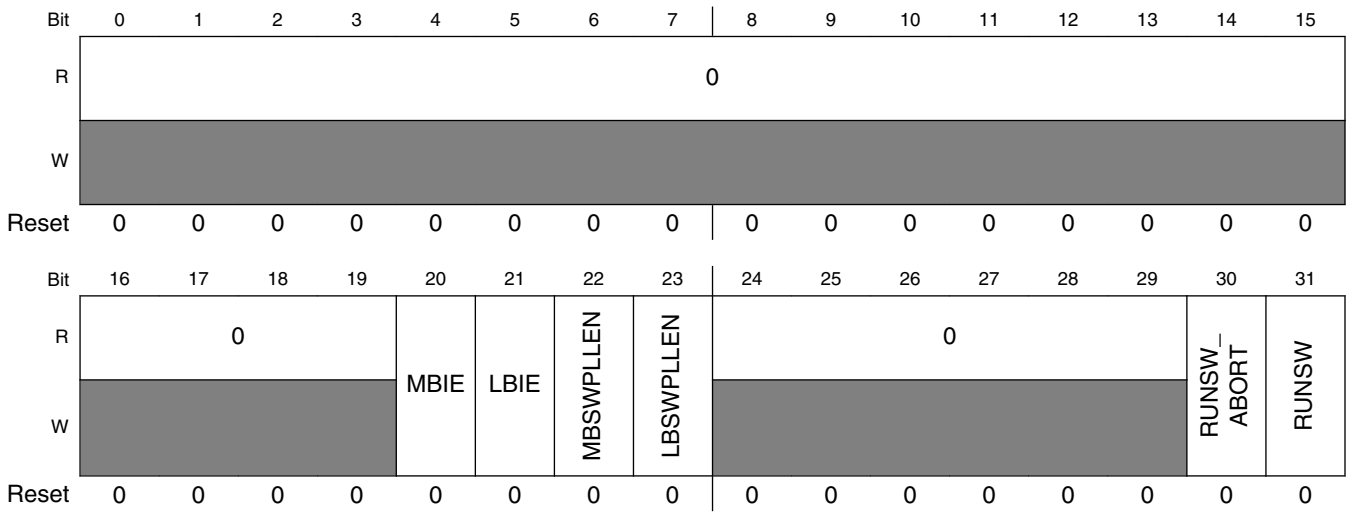
79.7.2 STCU2 Run Software Register (STCU2_RUNSW)

The STCU2_RUNSW register defines the RUN bit to start the on-line self testing procedure.

The R/W fields in this register are readable at any time. However, you can write to these fields only when both of the following conditions are true:

- STCU2_CFG[WRP] = 0
- On-line Self-Test phase is active. The On-Line condition starts when the Off-Line is over. In this condition, system is alive and the SW can program the STCU2

Address: 0h base + 4h offset = 4h



STCU2_RUNSW field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 MBIE	MBIST Interrupt Enable 0 Interrupt is not generated at the end of the software MBIST execution phase of the selected Memories. 1 At the end of the software MBIST execution phase of the concurrent selected memories the STCU2 forces an interrupt request to notify that MBISTs of concurrent selected memories are over. In case there is a sequence of concurrent runs in the same Self-Test run, the interrupt request is generated at the end of each concurrent runs provided that the software will clear the STCU2_INT_FLG[MBIFG] bit in between.
21 LBIE	LBIST Interrupt Enable 0 Interrupt is not generated at the end of the software LBIST execution phase of the selected LBIST. 1 At the end of the software LBIST execution phase of the concurrent selected LBIST the STCU2 forces an interrupt request to notify that LBISTs of the concurrent selected partitions are over. In case there is a sequence of concurrent runs in the same Self-Test run, the interrupt request is generated at the end of each concurrent run provided that the software will clear the STCU2_INT_FLG.LBIFG bit in between.
22 MBSWPLLEN	On-Line MBIST with PLL Enabled 0 On-Line MBIST is executed without using the on-chip PLL. 1 On-Line MBIST is executed using the PLL configuration provided by software. STCU2 does not take the PLL control but monitors the PLL lock signal to check if PLL is working correctly.

Table continues on the next page...

STCU2_RUNSW field descriptions (continued)

Field	Description
23 LBSWPLLEN	On-Line LBIST with PLL Enabled 0 On-Line LBIST is executed without using the on-chip PLL. 1 On-Line LBIST is executed using the PLL configuration provided by software. STCU2 does not take the PLL control but monitors the PLL lock signal to check if PLL is working correctly.
24–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 RUNSW_ABORT	Software abort of the self-test currently running This bit is automatically cleared by the STCU2 when the software self-test abort sequence has been completed. 0 On-line self-test abort is not requested. 1 On-line self-test abort is requested.
31 RUNSW	RUNSW The RUNSW bit is automatically cleared by the STCU2 when the on-line self testing procedure has been completed. 0 Idle 1 On-line self testing procedure is running

79.7.3 STCU2 SK Code Register (STCU2_SKC)

The STCU2_SKC register implements the security key code mechanism needed to access in write mode to the other STCU2 registers. In order to unlock the STCU2 access after:

- The STCU2 asynchronous reset
- The end of the STCU2 run

the software (IPS bus) or the SSCM interfaces have to apply the following sequence:

- write the key1 into the STCU2_SKC register
- write the key2 into the STCU2_SKC register

Depending on the off/on-line test step, the two keys will be different. Byte write operation is not allowed since the full key has to be recognized as one unit.

It should be noted that after the Self-Test sequence has been completed or the Bypass feature has been enabled (setting the bit BYP into the STCU2_CFG), the SSCM interface is no longer available.

In case of invalid access or sequence (Key1/2 have to be applied consecutively), a transfer error on the IPS or SSCM bus is asserted depending on the selected source. The STCU2 write access is locked and to unlock the access the sequence has to be applied again.

In case the STCU2 register access lasts more cycles than the one defined into the Hard-coded WDG time-out, the STCU2 write access is locked and the WDG and Register ITF clocks are switched off. Also in this case, in order to enable again the write access to the STCU2 and the WDG and Register ITF clocks, it is required to apply again the sequence.

In case it is required to extend the STCU2 register access cycles before the Hard-coded WDG time-out expires, only the Key2 has to be applied. The effect of this write operation is to re-initialize the WDG time-out counter. In such a condition, Key1 has not to be applied otherwise a transfer error on the IPS or SSCM bus is asserted depending on the selected source. The STCU2 write access is locked and to unlock the access the sequence has to be applied again.

The STCU2_SKC register is not readable. The value 00000000h is always returned in case of read operation.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	SKC																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_SKC field descriptions

Field	Description
0–31 SKC	STCU2 security key code for off-line Test = D3FEA98Bh: Key1 to unlock the write access the STCU2 = 2C015674h: Key2 to unlock the write access the STCU2 STCU2 security key code for on-line Test = 753F924Eh: Key1 to unlock the write access the STCU2 = 8AC06DB1h: Key2 to unlock the write access the STCU2

79.7.4 STCU2 Configuration Register (STCU2_CFG)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_CFG register includes the global configuration of the STCU2 and can be updated both in the Off/On-Line Test steps.

The access to this register is described in the following figure. It further depends on the state of the WRP bit as follows:

- When WRP = 0: The register can be written during the on-line Self-Test case without restrictions
- When WRP = 1:
  - The only bit that can be written without any restriction is WRP.
  - In case there are software operations that write all the register's bits, a transfer error is raised only if the value of the selected byte written differs from the current status of the register. This functionality has been implemented to prevent potential compilation behavior that might invalidate this single bit clean capability.

MBIST can be run in three different test types according to how the STCU2_CFG register bits 27 and 28 are set.

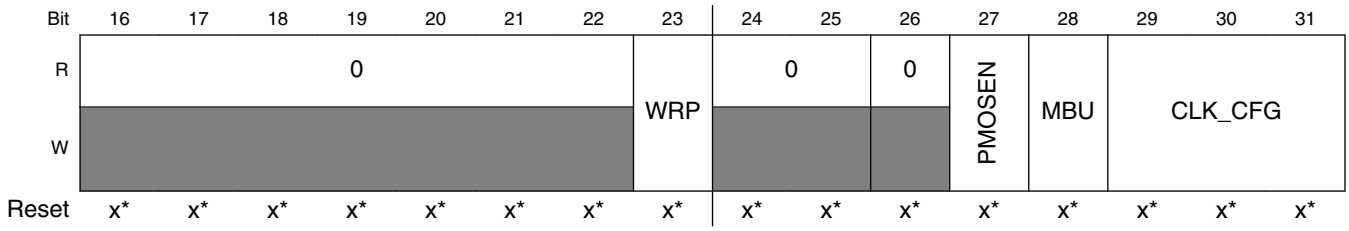
The following table describes the test algorithm and coverage of the different test types.

Table 79-12. MBIST algorithm and coverage according to test type

MBIST Type	STCU2_CFG[PMOSEN]	STCU2_CFG[MBU]	MBIST Algorithm	Internal Nodes/ Circuits Additionally Covered by Test	Usecase	Test Cycle
Full Test	1	0	Test memory using all built in algorithms. Open PMOS algorithm included	Address Decoder and Bitcell as well as resistive defects in the address decoder	Used by FSL production test. Recommended to use in the field for MBIST online self-test.	About 3 to 4x more than Auto Test
Reduced Test	0	0	Test memory using all built in algorithms except the open PMOS algorithm.	Address Decoder and Bitcell	Recommended to use in the field for MBIST online self-test if the Full Test takes too long.	About 10% less than Full Test
Auto Test	x	1	A smaller set of algorithms which target latent defect mechanisms.	Latent defects such as NBTI of the PMOS transistors in the bitcells	Recommended to use for MBIST offline self-test as an optimum balance of test time vs. fault coverage.	About 3 to 4x faster than Full Test

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PTR							0							
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*



* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_CFG field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 PTR	LBIST or MBIST pointer PTR defines the logical pointer to the first LBIST or MBIST to be scheduled when the self testing procedure is enabled. PTR is the entry pointer to a linked list of BIST descriptors. If PTR = 0 then the LBIST0 is initially scheduled. You can always read this field. In the off-line Self-Test phase, you can always write to this field. In the on-line Self-Test phase, you can write to this field only when WRP = 0. 0h to (LBIST -1): pointer to LBIST 10h to (10h + MBIST -1): pointer to MBIST 7Fh: pointer to NIL. No BIST execution.
8–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 WRP	Write Protection 0: Specific STCU2 registers can be written through IPS bus interface after Off-Line Self-Test sequence has been completed 1: Specific STCU2 registers cannot be written though IPS also after Off-Line Self-Test sequence has been completed preventing in such a way any user application write operation
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 PMOSEN	MBIST PMOS Test Enable When MBU is 1 then "Auto test" MBIST test type will be selected regardless of the setting of this field.. 0 When MBU is 0 then selects the "Reduced test" MBIST test type. 1 When MBU is 0 then selects the "Full test" MBIST test type.
28 MBU	MBIST MBU Test Enabled 0 Either "Full test" or "Reduced test" MBIST test type will be selected according to the PMOSEN value. 1 Selects "Auto test" MBIST test type regardless of the PMOSEN value.

Table continues on the next page...

STCU2_CFG field descriptions (continued)

Field	Description
29–31 CLK_CFG	Logic, Memory Bist and STCU2 core CLK Clock configuration CLK_CFG defines the ratio between the sys_clk and the TCK used to program both the LBIST and the MBIST and the STCU2 core clock. The punch-out mechanism is used to generate the derived clocks. The allowed configurations are the following: 000: sys_clk 001: sys_clk/2 010: sys_clk/3 ... 101: sys_clk/6 110: sys_clk/7 111: sys_clk/8 You can always read this field. In the off-line Self-Test phase, you can always write to this field. In the on-line Self-Test phase, you can write to this field only when WRP = 0.

79.7.5 STCU2 PLL Configuration Register (STCU2_PLL_CFG)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_PLL_CFG register defines the parameters used to program the PLL only when the Off-line L/MBIST are performed and STCU2_RUN[MBPLLEN] = 1 or STCU2_RUN[LBPLLEN] = 1.

In the On-line condition, these bits are not effective. It is also possible to set the PLL during active On-line mode to prevent any potential functionality issue.

The R/W fields in this register are readable at any time. However, you can write to these fields only when Off-line Self-Test phase is still active.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		PLLODF						0					PLLIDF			0									PLLLDF						
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_PLL_CFG field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 PLLODF	PLL Output Division Factor The value of this field drives the Output Division factor of the PLL embedded into the device. Refer to the PLL specifications to define the exact mapping between the PLLODF value and the related PLL's Output Division factor.
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 PLLIDF	PLL Input Division Factor The value of this field drives the Input Division factor of the PLL embedded into the device. Refer to the PLL specifications to define the exact mapping between the PLLIDF value and the related PLL's Input Division factor.
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 PLLLDF	PLL Loop Division Factor The value of this field drives the Loop Division factor of the PLL embedded into the device. Refer to the PLL specifications to define the exact mapping between the PLLLDF value and the related PLL's Loop Division factor.

79.7.6 STCU2 Watchdog Register Granularity (STCU2_WDG)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

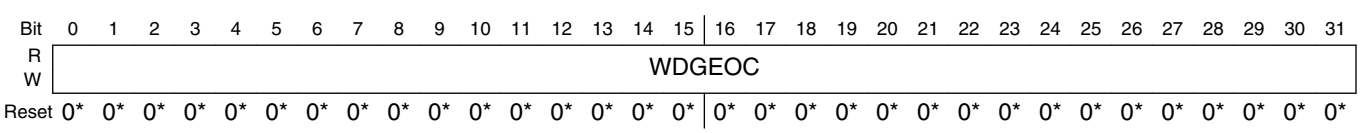
The STCU2_WDG register defines the time budget of LBIST and MBIST execution providing a protection mechanism in case of dead-lock or end-less conditions during the Self-Test procedure. When the Off-line Self-Test sequence is not run and the BYP bit of the STCU2_CFG register is not set, the bits 15...0 defines the time-out before the bit WDTO of the STCU2_ERR register is set and the STCU2 core clock is switched off.

In case Off/On-line Self Test sequence is run, it defines the time-out of the execution run.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 14h offset = 14h



* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

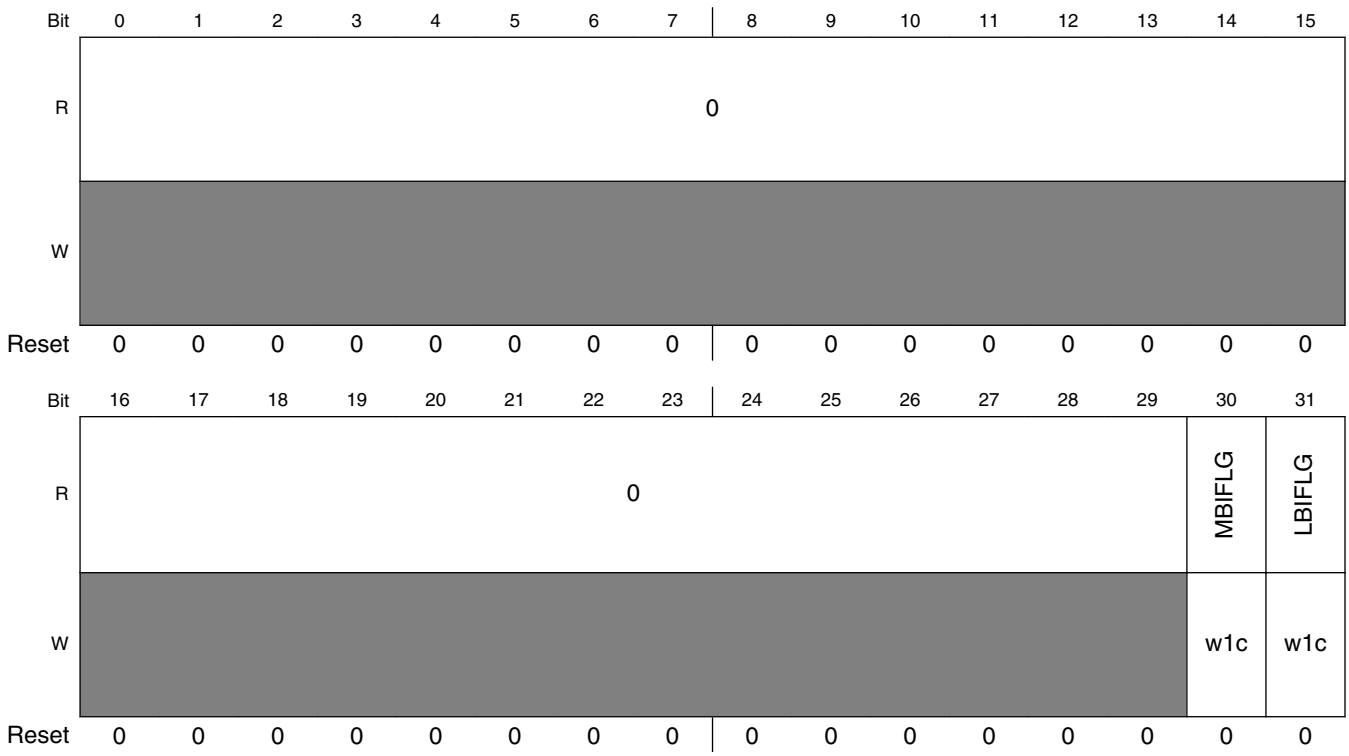
STCU2_WDG field descriptions

Field	Description
0–31 WDGEOC	Watchdog End of Count Timer This value has to be set in order to define the time budget related to the Off/On-Line Self Test execution and check that everything is correctly working within this slot of time. The allowed delay time are the following: 0000 0000h: 1 x 16 STCU2 Core clock cycles 0000 0001h: 2 x 16 STCU2 Core clock cycles 0000 0002h: 3 x 16 STCU2 Core clock cycles ... FFFF FFFDh: 4294967294 x 16 STCU2 Core clock cycles FFFF FFFEh: 4294967295 x 16 STCU2 Core clock cycles FFFF FFFFh: 4294967296 x 16 STCU2 Core clock cycles

79.7.7 STCU2 Interrupt Flag Register (STCU2_INT_FLG)

The STCU2_INT_FLG register includes an MBIST interrupt pending bit and an LBIST interrupt pending bit. Each bit is effective only during the On-Line Self-Test phase and is managed only when the related control bit in the STCU2_RUNSW register (MBIE or LBIE) is enabled.

Address: 0h base + 18h offset = 18h



STCU2_INT_FLG field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 MBIFLG	MBIST Interrupt Flag 0 No interrupt is pending 1 An interrupt, highlighting that the On-Line execution of the concurrent scheduled MBIST is completed, is pending. Write 1 to clear the bit.
31 LBIFLG	LBIST Interrupt Flag 0 No interrupt is pending 1 An interrupt, highlighting that the On-Line execution of the concurrent scheduled LBIST is completed, is pending. Write 1 to clear the bit.

79.7.8 STCU2 Error Register (STCU2_ERR_STAT)

The STCU2_ERR_STAT register includes the status flags related to the STCU2 internal fault conditions occurred during the configuration or the On/Off-Line self testing execution.

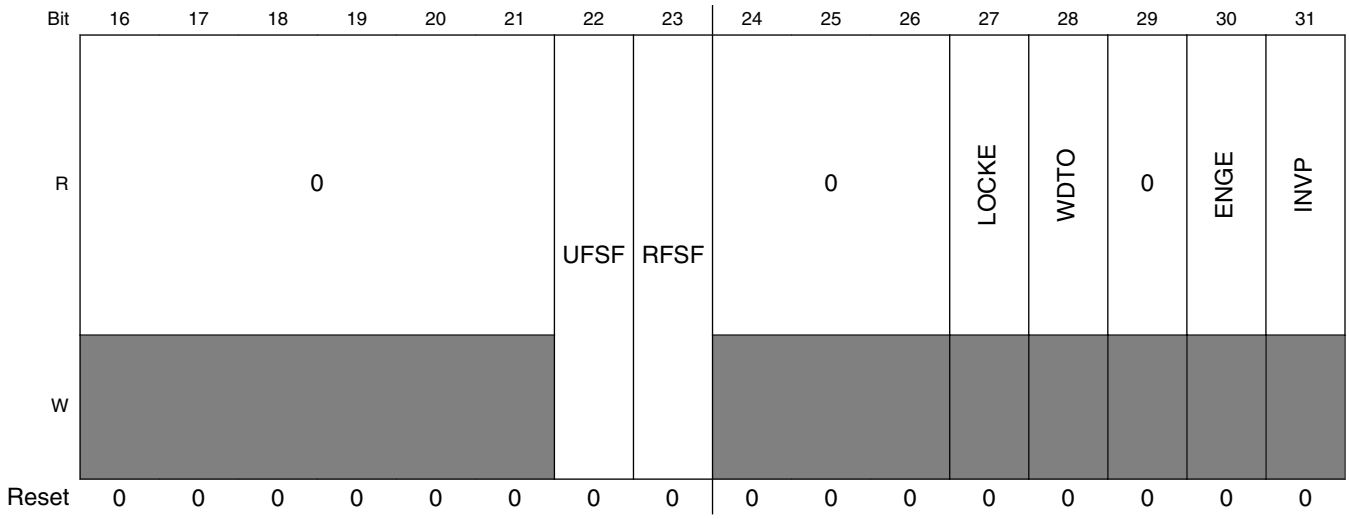
The UFSF and RFSF can be set/cleared using the FCCU dedicated channels.

The access to this register is described in the following figure and as follows:

- If you select the byte write capability to write only the UFSF and RFSF, then there is no restriction in writing these bits.
- If your software performs the write operations on other bits besides UFSF/RFSF, then a transfer error is generated only if the value you are writing to the other bits differs from their value currently stored in the register. This functionality has been implemented to prevent potential compilation behavior that might invalidate the UFSF/RFSF single bit set/clean capability.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0						ABORTHW	ABORTSW	0			LOCKESW	WDTOSW	0	ENGESW	INVPSW
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



STCU2_ERR_STAT field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 ABORTHW	On-line hardware abort flag You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 No hardware abort was requested during the On-Life Self-Test sequence 1 A hardware abort was detected during the On-Life Self-Test sequence
7 ABORTSW	On-line software abort flag You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 No software abort was requested during the On-Life Self-Test sequence 1 A software abort was detected during the On-Life Self-Test sequence
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 LOCKESW	On-Line LOCK Error You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 In case PLL is enabled, it is correctly locked during the Self-Test sequence 1 When the PLL is enabled, this flag highlight that there has been an unexpected PLL unlock event during the On-Line Self-Test sequence execution. The On-Line Self-test run is stopped and the status of the current running LBISTs or MBISTs is saved into the related registers. The LOCK signal is monitored during the LBIST run when STCU2_RUNSW.LBSWPLLEN is set and/or during the MBIST run when STCU2_RUNSW[MBSWPLLEN] = 1.
12 WDTOSW	On-Line Watchdog time-out You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Table continues on the next page...

STCU2_ERR_STAT field descriptions (continued)

Field	Description
	0 LBIST and MBIST time slot have been completed within the assigned watchdog time. 1 LBIST and MBIST time slot haven't been completed within the assigned watchdog time or there are internal mismatches among End of Execution signals. The conditions that flag the failures are the following: • MBIST BEND status flags that at least one of the selected MBIST run is not finished • lbist_done of the selected LBIST flags the LBIST run is/are not finished
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 ENGESW	On-Line Engine Error You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 Valid Engine execution 1 Invalid Engine execution. The following conditions set this bit: • STCU2 state machines (Watchdog, Master and Loader/Shifter) selects an unused code • Serial pass/fail status of the selected MBIST and the global BBAD flag are not aligned • Serial bend status of the selected MBIST and the global BEND flag are not aligned
15 INVPSW	On-Line Invalid Pointer You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 Valid linked pointer list 1 Invalid linked pointer list. The following conditions set this bit: • Initial LBIST or MBIST pointer is out of range • LBIST is selected when MBIST is concurrently running or vice versa • Error in the LBIST/MBIST linking (execution generates an infinite loop)
16–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 UFSF	Unrecoverable Faults Status Flag This flag reports the global status of the UF. This field can be set or cleared using the FCCU dedicated channel, and can also be set or cleared by software. 0 No errors that trigger the Unrecoverable Faults condition 1 There are errors that trigger the Unrecoverable Faults condition
23 RFSF	Recoverable Faults Status Flag This flag reports the global status of the RF. This field can be set or cleared using the FCCU dedicated channel, and can also be set or cleared by software. 0 No errors that trigger the Recoverable Faults condition 1 There are errors that trigger the Recoverable Faults condition
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 LOCKE	Off-Line LOCK Error 0 The PLL is correctly locked during the Self-Test sequence 1 When the PLL is enabled, this flag highlights that there has been an unexpected PLL unlock event during the Off-Line Self-Test sequence execution. The Off-Line Self-test run is stopped and the status of the current running LBISTs or MBISTs is saved into the related registers. The LOCK signal is

Table continues on the next page...

STCU2_ERR_STAT field descriptions (continued)

Field	Description
	monitored during the LBIST run when STCU2_RUN[LBPLLEN] = 1 and/or during the MBIST run when STCU2_RUN[MBPLLEN] = 1.
28 WDTO	Off-Line Watchdog time-out 0 LBIST and MBIST time slot have been completed within the assigned watchdog time. 1 LBIST and MBIST time slot haven't been completed within the assigned watchdog time or there are internal mismatches among End of Execution signals. The conditions that flag the failures are the following: • The STCU2_RUN[RUN] bit or the STCU2_RUN[BYP] bit is not set before the Watchdog reaches the End of Count • MBIST BEND status flags that at least one of the selected MBIST run is not finished • lbist_done of the selected LBIST flags the LBIST run is/are not finished
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 ENGE	Off-Line Engine Error 0 Valid Engine execution 1 Invalid Engine execution. The following conditions set this bit: • STCU2 state machines (Watchdog, Master and Loader/Shifter) selects an unused code • Serial pass/fail status of the selected MBIST and the global BBAD flag are not aligned • Serial bend status of the selected MBIST and the global BEND flag are not aligned
31 INVP	Off-Line Invalid pointer 0 Valid linked pointer list 1 Invalid linked pointer list. The following conditions set this bit: • Initial LBIST or MBIST pointer is out of range • LBIST is selected when MBIST is concurrently running or vice versa • Error in the LBIST/MBIST linking (execution generates an infinite loop)

79.7.9 STCU2 Error FM Register (STCU2_ERR_FM)

NOTE

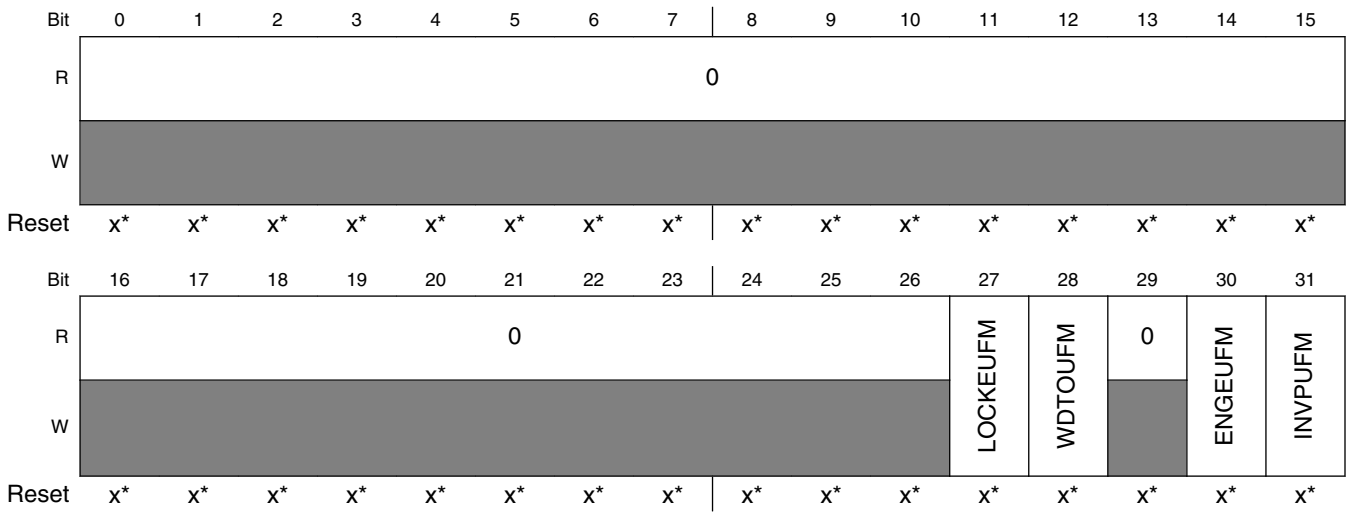
The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_ERR_FM register defines the fault mapping of the STCU2 faults described into the register STCU2_ERR_STAT in terms of Unrecoverable or Recoverable Fault. All sources of internal faults can be routed to UF and RF.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 28h offset = 28h



* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_ERR_FM field descriptions

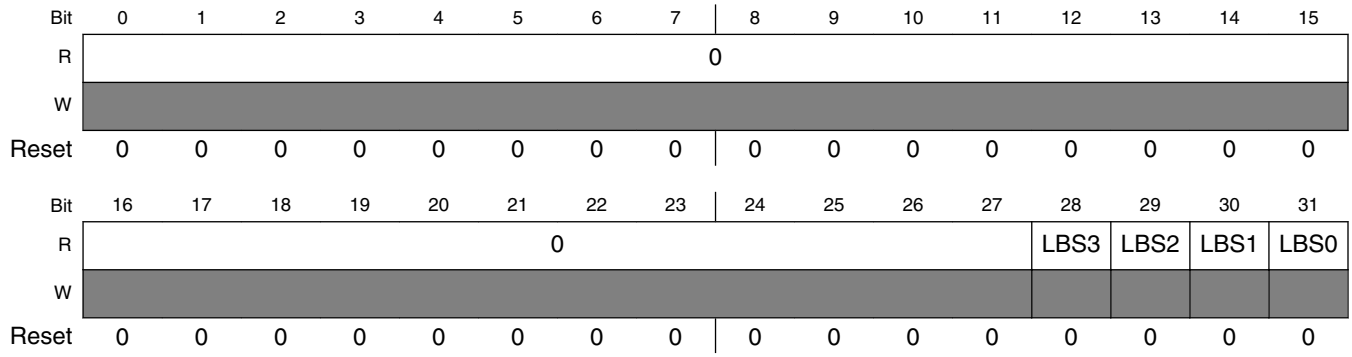
Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 LOCKEUFM	PLL LOCK Unrecoverable Fault Mapping 0 Recoverable Fault Mapping 1 Unrecoverable Fault Mapping
28 WDTOUFM	WatchDog time-out Unrecoverable Fault Mapping 0 Recoverable Fault Mapping 1 Unrecoverable Fault Mapping
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 ENGEUFM	Engine Error Unrecoverable fault Mapping 0 Recoverable Fault Mapping 1 Unrecoverable Fault Mapping
31 INVPUFM	Invalid Pointer Unrecoverable Fault Mapping 0 Recoverable Fault Mapping 1 Unrecoverable Mapping

79.7.10 STCU2 Off-Line LBIST Status Register (STCU2_LBS)

The STCU2_LBS register includes the results corresponding to the execution of the selected Off-Line LBIST.

The size of the register depends on the number of LBIST .

Address: 0h base + 2Ch offset = 2Ch



STCU2_LBS field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LBS3	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
29 LBS2	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
30 LBS1	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
31 LBS0	Off-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution

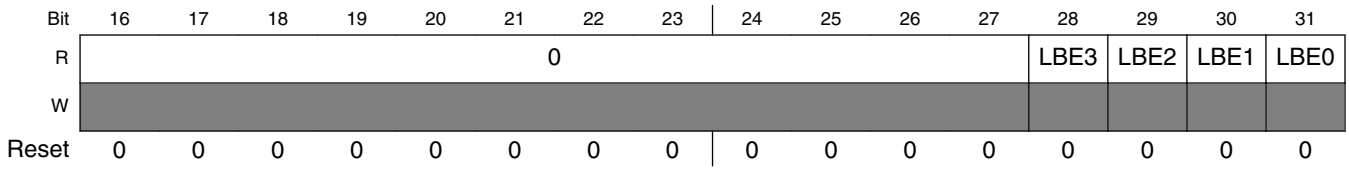
79.7.11 STCU2 Off-Line LBIST End Flag Register (STCU2_LBE)

The STCU2_LBE register includes the End Flag related to the execution of the selected Off-Line LBIST.

The size of the register depends on the number of LBIST .

Address: 0h base + 30h offset = 30h





STCU2_LBE field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LBE3	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished
29 LBE2	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished
30 LBE1	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished
31 LBE0	Off-Line End status of the selected LBIST 0 LBIST execution not yet completed 1 LBIST execution finished

79.7.12 STCU2 On-Line LBIST Status Register (STCU2_LBSSW)

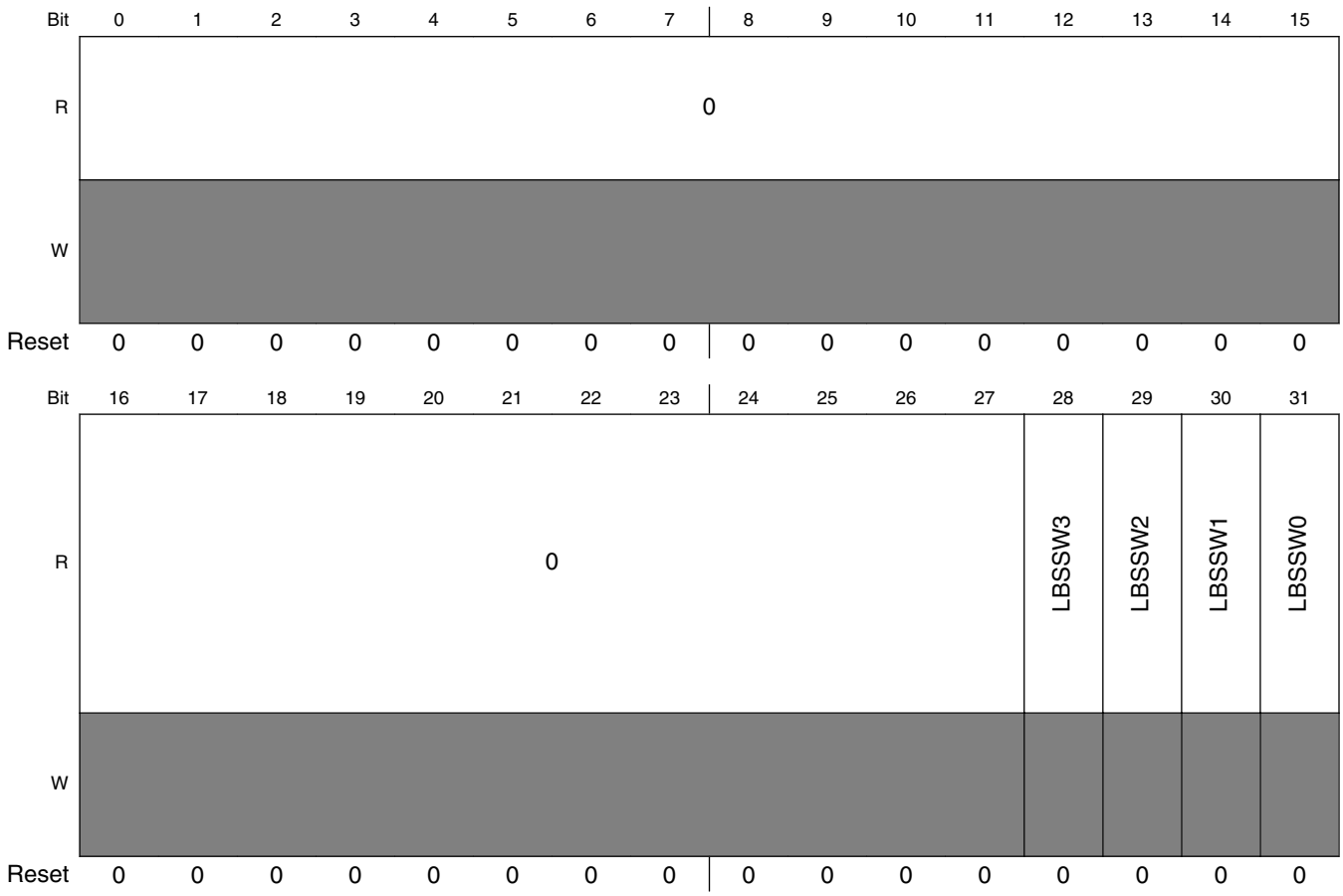
The STCU2_LBSSW register includes the results corresponding to the execution of the selected On-Line LBIST.

The size of the register depends on the number of LBIST .

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Register description

Address: 0h base + 34h offset = 34h



STCU2_LBSSW field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LBSSW3	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
29 LBSSW2	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
30 LBSSW1	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution
31 LBSSW0	LBSSWx: On-Line status of the selected LBIST 0 Failed LBIST execution 1 Successful LBIST execution

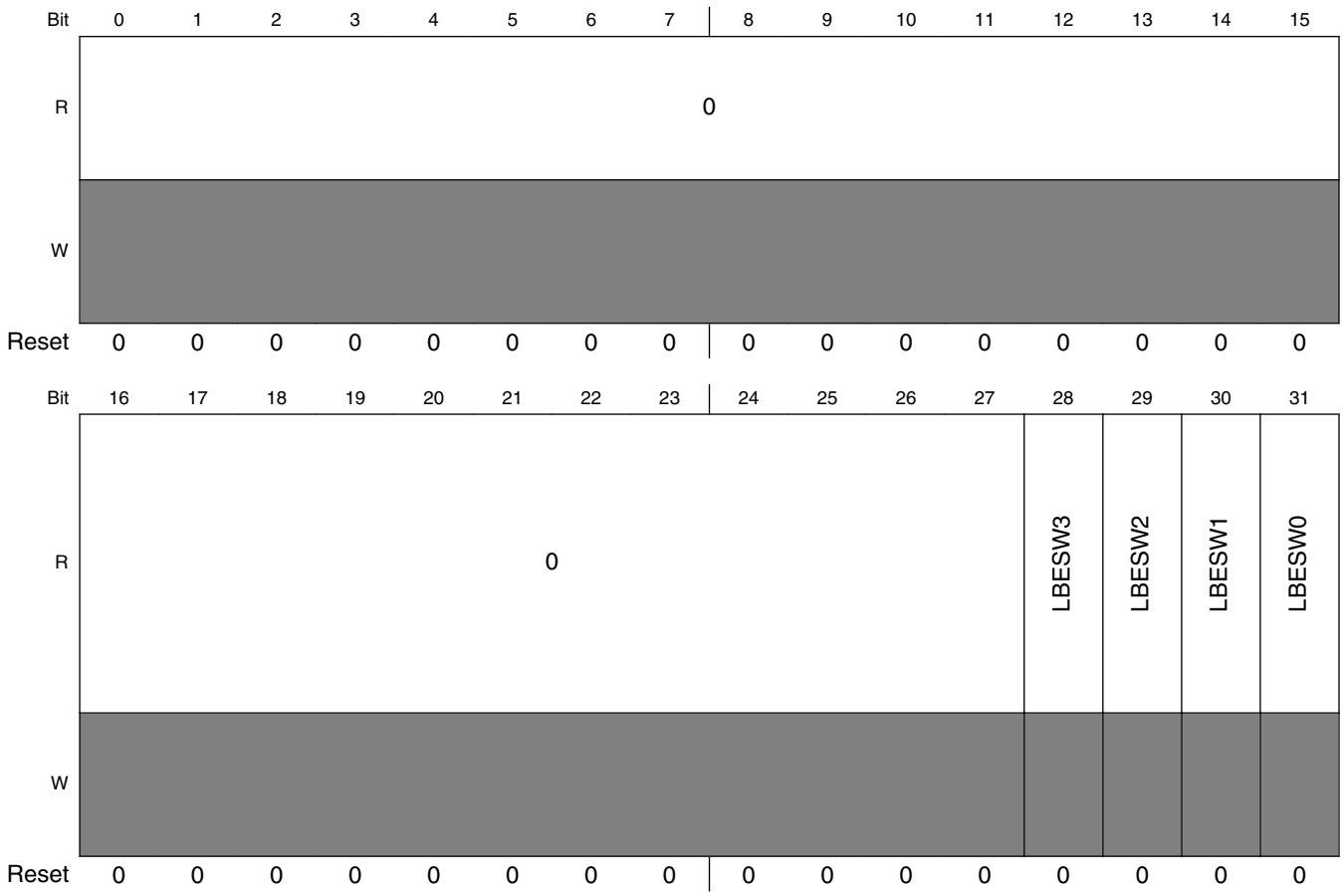
79.7.13 STCU2 On-Line LBIST End Flag Register (STCU2_LBESW)

The STCU2_LBESW register includes the End Flag related to the execution of the selected On-Line LBIST.

The size of the register depends on the number of LBIST .

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 38h offset = 38h



STCU2_LBESW field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LBESW3	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished

Table continues on the next page...

STCU2_LBESW field descriptions (continued)

Field	Description
29 LBESW2	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished
30 LBESW1	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished
31 LBESW0	LBESWx: On-Line LBIST End status 0 LBIST execution not yet completed 1 LBIST execution finished

79.7.14 STCU2 On-Line LBIST Reset Management (STCU2_LBRMSW)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LBRMSW register defines the On-Line reset mapping for each LBIST. Global or specific functional reset can be programmed depending on the On-Line Self-Test requirements and top level integration architecture.

The size of the register depends on the number of LBIST .

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												LBRMSW3	LBRMSW2	LBRMSW1	LBRMSW0
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LBRMSW field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LBRMSW3	On-Line LBIST Reset Management **NOTE:** In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Dedicated functional reset is pulsed at the end of LBIST run 1 Global functional reset is pulsed at the end of LBIST run
29 LBRMSW2	On-Line LBIST Reset Management **NOTE:** In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Dedicated functional reset is pulsed at the end of LBIST run 1 Global functional reset is pulsed at the end of LBIST run
30 LBRMSW1	On-Line LBIST Reset Management **NOTE:** In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Dedicated functional reset is pulsed at the end of LBIST run 1 Global functional reset is pulsed at the end of LBIST run
31 LBRMSW0	On-Line LBIST Reset Management **NOTE:** In case one of the selected LBIST has this bit set to '1', then only the Global functional reset will be pulsed. 0 Dedicated functional reset is pulsed at the end of LBIST run 1 Global functional reset is pulsed at the end of LBIST run

79.7.15 STCU2 LBIST Unrecoverable FM Register (STCU2_LBUFM)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

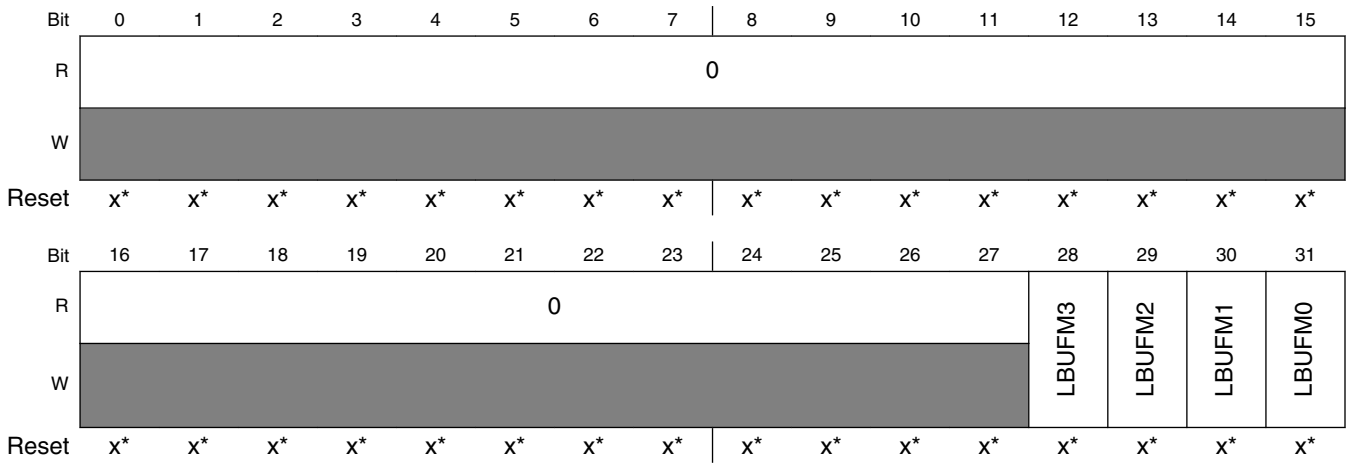
The STCU2_LBUFM register defines the fault mapping of each LBIST in terms of Unrecoverable or Recoverable Fault.

The size of the register depends on the number of LBIST .

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 40h offset = 40h



* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LBUFM field descriptions

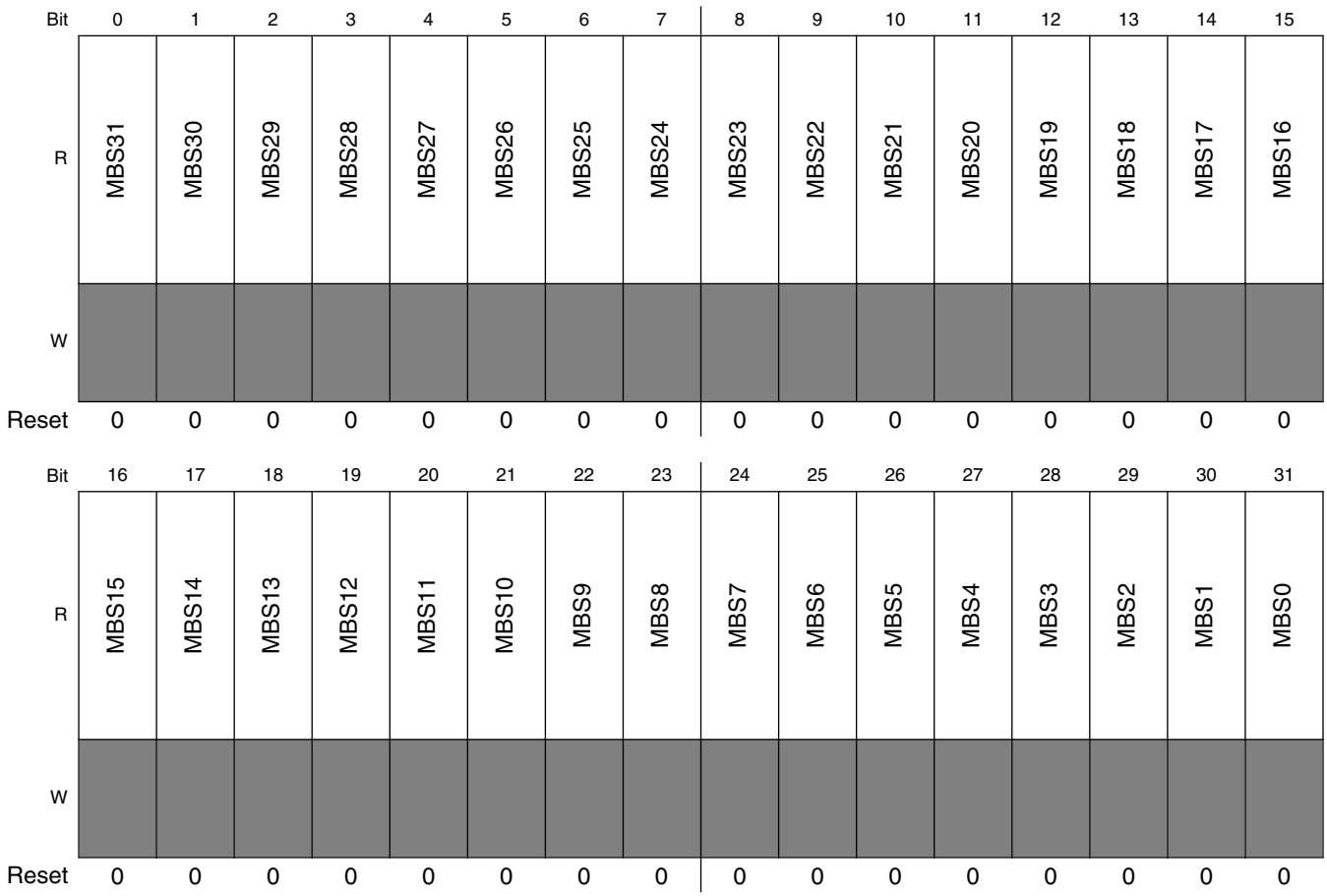
Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LBUFM3	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
29 LBUFM2	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
30 LBUFM1	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
31 LBUFM0	LBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

79.7.16 STCU2 Off-Line MBIST Status Low Register (STCU2_MBSL)

The STCU2_MBSL register includes the results corresponding to the execution of the selected Off-Line MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs .

Address: 0h base + 44h offset = 44h



STCU2_MBSL field descriptions

Field	Description
0 MBS31	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
1 MBS30	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
2 MBS29	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished.

Table continues on the next page...

STCU2_MBSL field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
3 MBS28	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
4 MBS27	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
5 MBS26	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
6 MBS25	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
7 MBS24	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
8 MBS23	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
9 MBS22	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
10 MBS21	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBSL field descriptions (continued)

Field	Description
	NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
11 MBS20	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
12 MBS19	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
13 MBS18	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
14 MBS17	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
15 MBS16	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
16 MBS15	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
17 MBS14	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSL field descriptions (continued)

Field	Description
18 MBS13	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
19 MBS12	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
20 MBS11	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBS10	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBS9	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBS8	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBS7	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBS6	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished.

Table continues on the next page...

STCU2_MBSL field descriptions (continued)

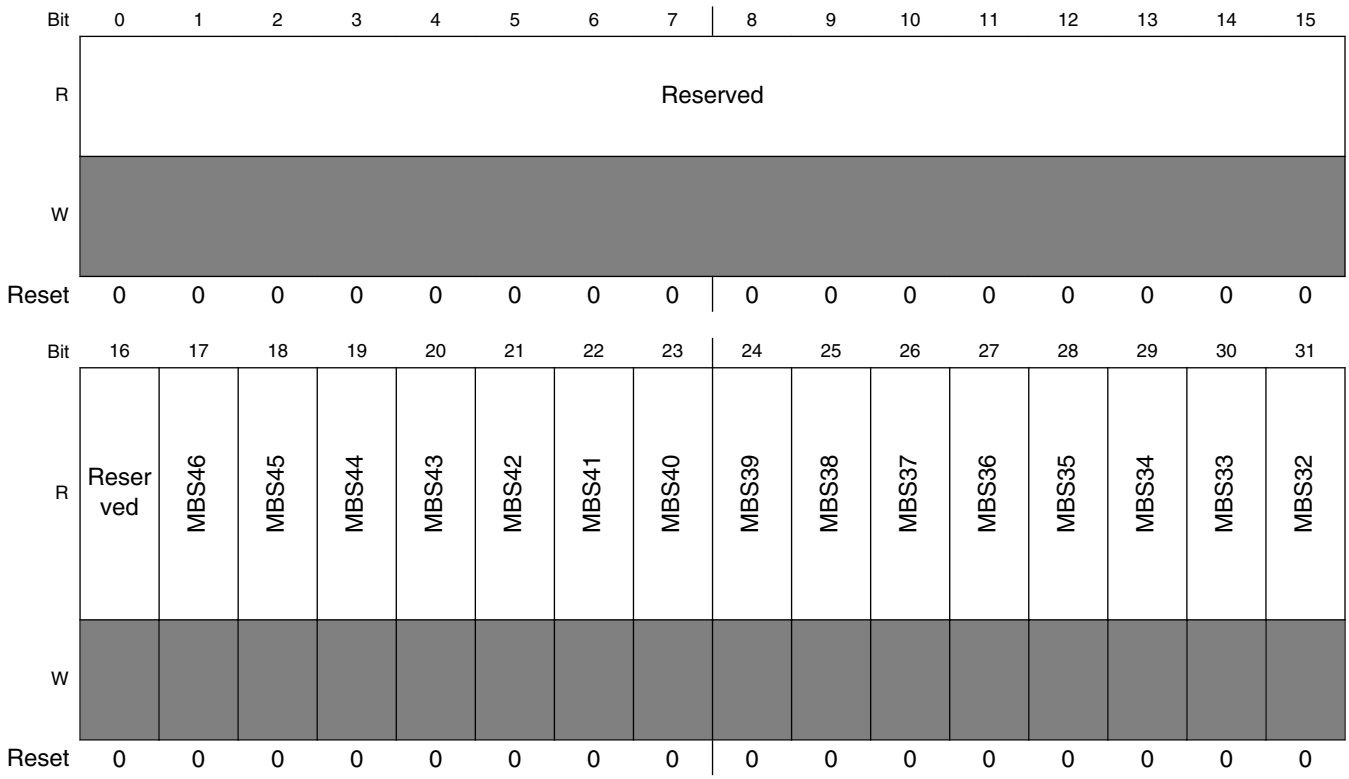
Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
26 MBS5	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBS4	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBS3	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBS2	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBS1	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBS0	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

79.7.17 STCU2 Off-Line MBIST Status Medium Register (STCU2_MBSM)

The STCU2_MBSM register includes the results corresponding to the execution of the selected Off-Line MBIST in the range NMCUT = 32-63.

The size of the register depends on the number of BISTed RAMs/ROMs .

Address: 0h base + 48h offset = 48h



STCU2_MBSM field descriptions

Field	Description
0–16 Reserved	This field is reserved.
17 MBS46	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
18 MBS45	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
19 MBS44	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSM field descriptions (continued)

Field	Description
20 MBS43	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBS42	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBS41	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBS40	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBS39	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBS38	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
26 MBS37	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBS36	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished).

Table continues on the next page...

STCU2_MBSM field descriptions (continued)

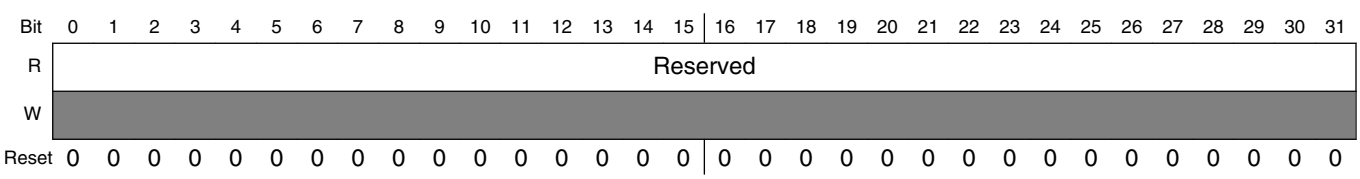
Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBS35	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBS34	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBS33	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBS32	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

79.7.18 STCU2 Off-Line MBIST Status High Register (STCU2_MBSH)

The STCU2_MBSH register includes the results corresponding to the execution of the selected Off-Line MBIST in the range NMCUT = 64-95.

The size of the register depends on the number of BISTed RAMs/ROMs.

Address: 0h base + 4Ch offset = 4Ch



STCU2_MBSH field descriptions

Field	Description
0–31 Reserved	This field is reserved.

79.7.19 STCU2 Off-Line MBIST End Flag Low Register (STCU2_MBEL)

The STCU2_MBEL register includes the End Flag related to the execution of the selected Off-Line MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs.

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBE31	MBE30	MBE29	MBE28	MBE27	MBE26	MBE25	MBE24	MBE23	MBE22	MBE21	MBE20	MBE19	MBE18	MBE17	MBE16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBE15	MBE14	MBE13	MBE12	MBE11	MBE10	MBE9	MBE8	MBE7	MBE6	MBE5	MBE4	MBE3	MBE2	MBE1	MBE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBEL field descriptions

Field	Description
0 MBE31	Off-Line End status (0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBEL field descriptions (continued)

Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
1 MBE30	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
2 MBE29	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
3 MBE28	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
4 MBE27	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
5 MBE26	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
6 MBE25	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
7 MBE24	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
8 MBE23	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
9 MBE22	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
10 MBE21	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
11 MBE20	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
12 MBE19	Off-Line End status (0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBEL field descriptions (continued)

Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
13 MBE18	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
14 MBE17	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
15 MBE16	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
16 MBE15	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
17 MBE14	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
18 MBE13	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBE12	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
20 MBE11	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBE10	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBE9	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBE8	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBE7	Off-Line End status (0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBEL field descriptions (continued)

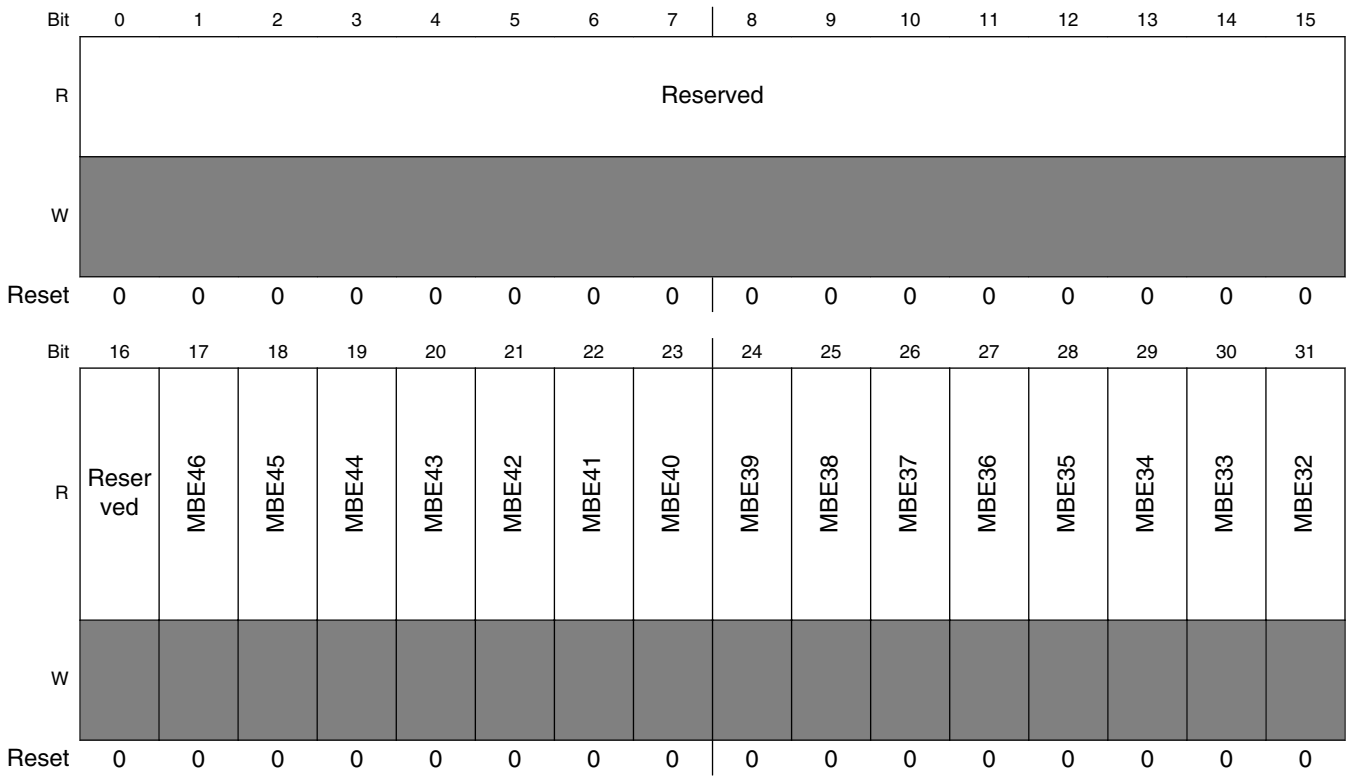
Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
25 MBE6	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
26 MBE5	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBE4	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
28 MBE3	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBE2	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBE1	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBE0	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

79.7.20 STCU2 Off-Line MBIST End Flag Medium Register (STCU2_MBEM)

The STCU2_MBEM register includes the End Flag related to the execution of the selected Off-Line MBIST in the range NMCUT = 32-63.

The size of the register depends on the number of BISTed RAMs/ROMs.

Address: 0h base + 54h offset = 54h



STCU2_MBEM field descriptions

Field	Description
0–16 Reserved	This field is reserved.
17 MBE46	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
18 MBE45	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBE44	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
20 MBE43	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBE42	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBEM field descriptions (continued)

Field	Description
22 MBE41	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBE40	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBE39	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBE38	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
26 MBE37	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBE36	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
28 MBE35	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBE34	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBE33	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBE32	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

79.7.21 STCU2 Off-Line MBIST End Flag High Register (STCU2_MBEH)

The STCU2_MBEH register includes the End Flag related to the execution of the selected Off-Line MBIST in the range NMCUT = 64..95.

The size of the register depends on the number of BISTed RAMs/ROMs.

Address: 0h base + 58h offset = 58h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBEH field descriptions

Field	Description
0–31 Reserved	This field is reserved.

79.7.22 STCU2 On-Line MBIST Status Low Register (STCU2_MBSLSW)

The STCU2_MBSLSW register includes the results corresponding to the execution of the selected On-Line MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 5Ch offset = 5Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBSSW31	MBSSW30	MBSSW29	MBSSW28	MBSSW27	MBSSW26	MBSSW25	MBSSW24	MBSSW23	MBSSW22	MBSSW21	MBSSW20	MBSSW19	MBSSW18	MBSSW17	MBSSW16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBSSW15	MBSSW14	MBSSW13	MBSSW12	MBSSW11	MBSSW10	MBSSW9	MBSSW8	MBSSW7	MBSSW6	MBSSW5	MBSSW4	MBSSW3	MBSSW2	MBSSW1	MBSSW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBSLSW field descriptions

Field	Description
0 MBSSW31	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
1 MBSSW30	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
2 MBSSW29	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
3 MBSSW28	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
4 MBSSW27	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
5 MBSSW26	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
6 MBSSW25	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
7 MBSSW24	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSLSW field descriptions (continued)

Field	Description
8 MBSSW23	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
9 MBSSW22	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
10 MBSSW21	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
11 MBSSW20	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
12 MBSSW19	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
13 MBSSW18	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
14 MBSSW17	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
15 MBSSW16	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
16 MBSSW15	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
17 MBSSW14	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
18 MBSSW13	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
19 MBSSW12	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSLSW field descriptions (continued)

Field	Description
20 MBSSW11	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBSSW10	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBSSW9	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBSSW8	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBSSW7	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBSSW6	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
26 MBSSW5	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBSSW4	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBSSW3	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBSSW2	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBSSW1	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBSSW0	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

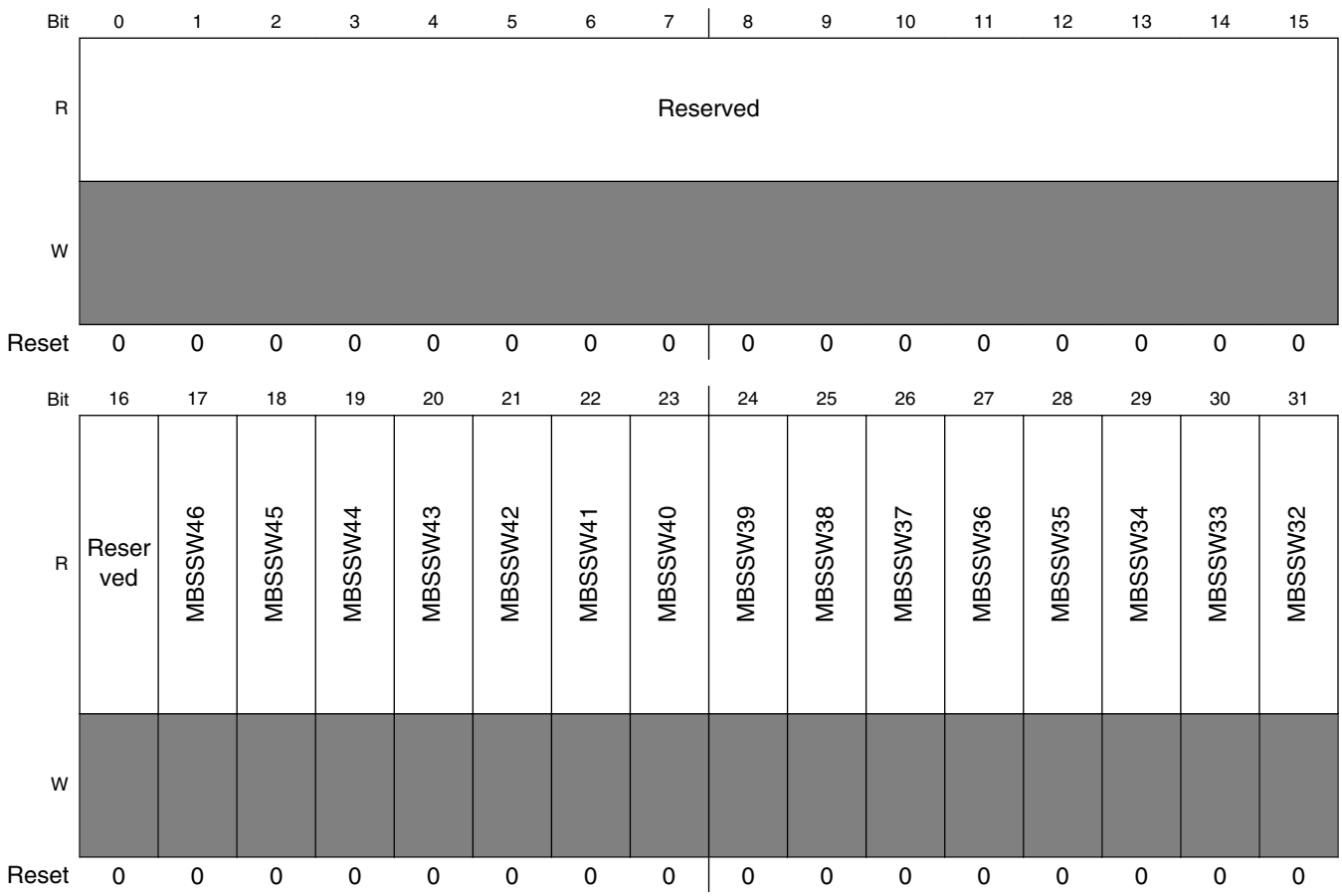
79.7.23 STCU2 On-Line MBIST Status Medium Register (STCU2_MBSMSW)

The STCU2_MBSMSW register includes the results corresponding to the execution of the selected On-Line MBIST in the range NMCUT = 32..63.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 60h offset = 60h



STCU2_MBSMSW field descriptions

Field	Description
0–16 Reserved	This field is reserved.
17 MBSSW46	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSMSW field descriptions (continued)

Field	Description
18 MBSSW45	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
19 MBSSW44	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
20 MBSSW43	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBSSW42	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBSSW41	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBSSW40	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBSSW39	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBSSW38	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
26 MBSSW37	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBSSW36	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBSSW35	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBSSW34	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSMSW field descriptions (continued)

Field	Description
30 MBSSW33	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBSSW32	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

79.7.24 STCU2 On-Line MBIST Status High Register (STCU2_MBSHSW)

The STCU2_MBSHSW register includes the results corresponding to the execution of the selected On-Line MBIST in the range NMCUT = 64..95.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 64h offset = 64h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBSHSW field descriptions

Field	Description
0–31 Reserved	This field is reserved.

79.7.25 STCU2 On-Line MBIST End Flag Low Register (STCU2_MBELSW)

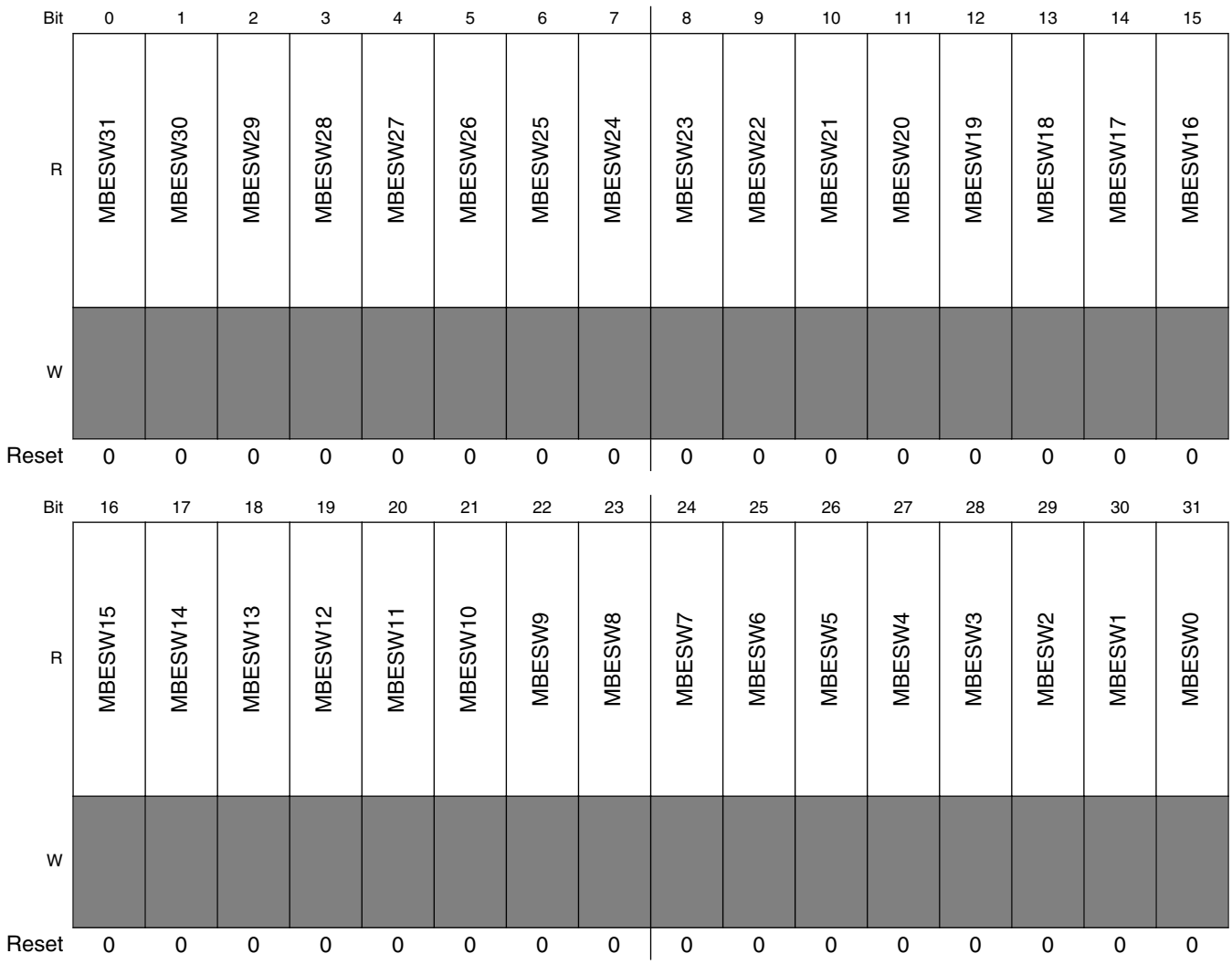
The STCU2_MBELSW register includes the End Flag related to the execution of the selected On-Line MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Register description

Address: 0h base + 68h offset = 68h



STCU2_MBELSW field descriptions

Field	Description
0 MBESW31	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
1 MBESW30	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
2 MBESW29	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
3 MBESW28	MBESWx: On-Line End status (0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBELSW field descriptions (continued)

Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
4 MBESW27	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
5 MBESW26	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
6 MBESW25	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
7 MBESW24	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
8 MBESW23	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
9 MBESW22	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
10 MBESW21	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
11 MBESW20	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
12 MBESW19	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
13 MBESW18	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
14 MBESW17	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
15 MBESW16	MBESWx: On-Line End status (0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBELSW field descriptions (continued)

Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
16 MBESW15	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
17 MBESW14	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
18 MBESW13	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBESW12	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
20 MBESW11	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBESW10	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBESW9	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBESW8	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBESW7	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBESW6	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
26 MBESW5	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBESW4	MBESWx: On-Line End status (0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBELSW field descriptions (continued)

Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
28 MBESW3	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBESW2	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBESW1	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBESW0	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

79.7.26 STCU2 On-Line MBIST End Flag Medium Register (STCU2_MBEMSW)

The STCU2_MBEMSW register includes the End Flag related to the execution of the selected On-Line MBIST in the range NMCUT = 32-63.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 6Ch offset = 6Ch



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved	MBESW46	MBESW45	MBESW44	MBESW43	MBESW42	MBESW41	MBESW40	MBESW39	MBESW38	MBESW37	MBESW36	MBESW35	MBESW34	MBESW33	MBESW32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBEMSW field descriptions

Field	Description
0–16 Reserved	This field is reserved.
17 MBESW46	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
18 MBESW45	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBESW44	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
20 MBESW43	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBESW42	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBESW41	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBESW40	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBESW39	On-Line End status (32 to 63) of the selected MBIST

Table continues on the next page...

STCU2_MBEMSW field descriptions (continued)

Field	Description
	0 MBIST execution still ongoing 1 MBIST execution finished
25 MBESW38	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
26 MBESW37	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBESW36	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
28 MBESW35	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBESW34	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBESW33	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBESW32	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

79.7.27 STCU2 On-Line MBIST End Flag High Register (STCU2_MBEHSW)

The STCU2_MBEHSW register includes the End Flag related to the execution of the selected On-Line MBIST in the range NMCUT = 64-95.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 70h offset = 70h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBEHSW field descriptions

Field	Description
0–31 Reserved	This field is reserved.

79.7.28 STCU2 MBIST Unrecoverable FM Low Register (STCU2_MBUFML)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_MBUFML register defines the fault mapping, in terms of Unrecoverable or Recoverable fault, of the MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 74h offset = 74h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MBUFM31	MBUFM30	MBUFM29	MBUFM28	MBUFM27	MBUFM26	MBUFM25	MBUFM24	MBUFM23	MBUFM22	MBUFM21	MBUFM20	MBUFM19	MBUFM18	MBUFM17	MBUFM16
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MBUFM15	MBUFM14	MBUFM13	MBUFM12	MBUFM11	MBUFM10	MBUFM9	MBUFM8	MBUFM7	MBUFM6	MBUFM5	MBUFM4	MBUFM3	MBUFM2	MBUFM1	MBUFM0
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_MBUFML field descriptions

Field	Description
0 MBUFM31	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
1 MBUFM30	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
2 MBUFM29	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
3 MBUFM28	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
4 MBUFM27	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
5 MBUFM26	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
6 MBUFM25	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
7 MBUFM24	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
8 MBUFM23	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
9 MBUFM22	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
10 MBUFM21	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
11 MBUFM20	MBIST Unrecoverable Fault Mapping

Table continues on the next page...

STCU2_MBUFML field descriptions (continued)

Field	Description
	0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
12 MBUFM19	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
13 MBUFM18	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
14 MBUFM17	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
15 MBUFM16	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
16 MBUFM15	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
17 MBUFM14	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
18 MBUFM13	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
19 MBUFM12	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
20 MBUFM11	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
21 MBUFM10	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
22 MBUFM9	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
23 MBUFM8	MBIST Unrecoverable Fault Mapping

Table continues on the next page...

STCU2_MBUFML field descriptions (continued)

Field	Description
	0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
24 MBUFM7	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
25 MBUFM6	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
26 MBUFM5	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
27 MBUFM4	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
28 MBUFM3	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
29 MBUFM2	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
30 MBUFM1	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
31 MBUFM0	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

79.7.29 STCU2 MBIST Unrecoverable FM Medium Register (STCU2_MBUFMM)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

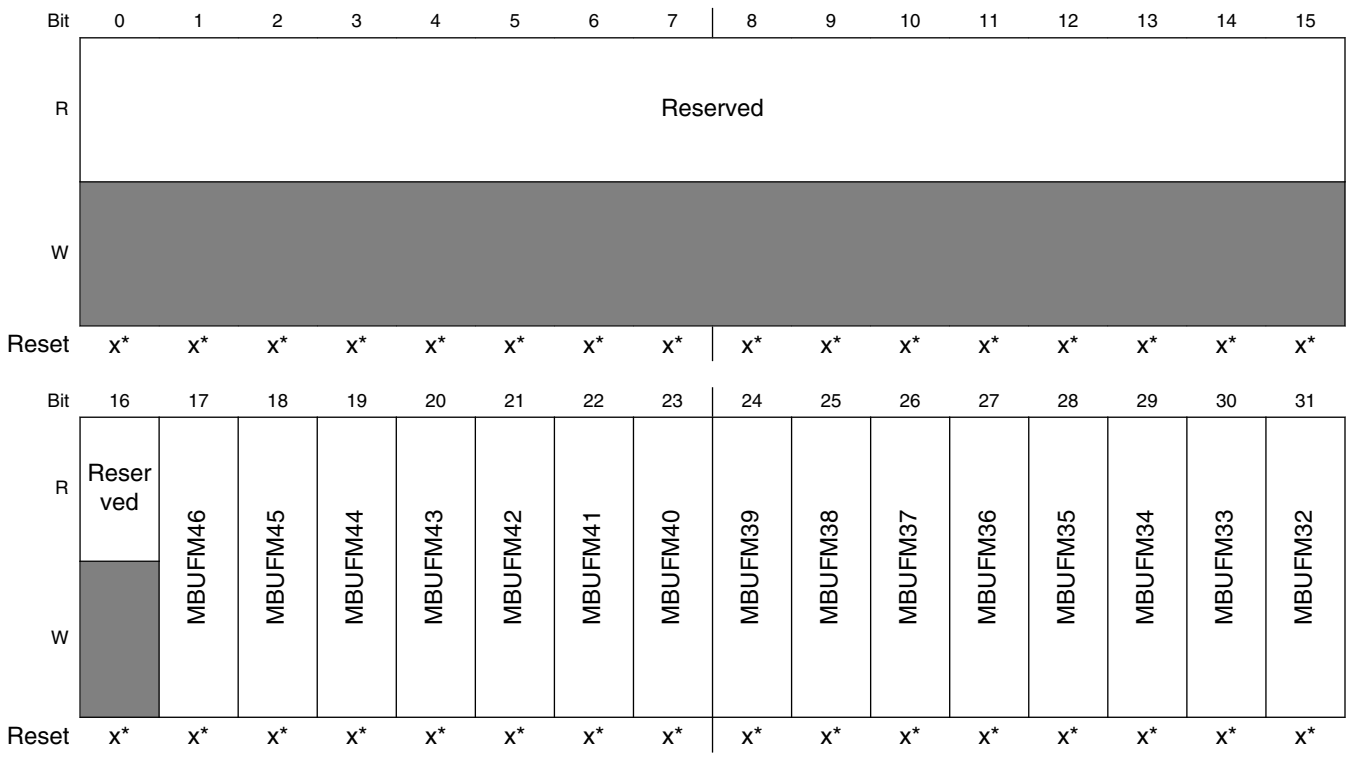
The STCU2_MBUFMM register defines the fault mapping, in terms of Unrecoverable or Recoverable fault, of the MBIST in the range NMCUT = 32-63.

The size of the register depends on the number of BISTed RAMs/ROMs.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 78h offset = 78h



* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_MBUFMM field descriptions

Field	Description
0–16 Reserved	This field is reserved.
17 MBUFM46	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
18 MBUFM45	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
19 MBUFM44	MBIST Unrecoverable Fault Mapping

Table continues on the next page...

STCU2_MBUFMM field descriptions (continued)

Field	Description
	0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
20 MBUFM43	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
21 MBUFM42	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
22 MBUFM41	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
23 MBUFM40	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
24 MBUFM39	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
25 MBUFM38	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
26 MBUFM37	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
27 MBUFM36	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
28 MBUFM35	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
29 MBUFM34	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
30 MBUFM33	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
31 MBUFM32	MBIST Unrecoverable Fault Mapping

Table continues on the next page...

STCU2_MBUFMM field descriptions (continued)

Field	Description
	0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

79.7.30 STCU2 MBIST Unrecoverable FM High Register (STCU2_MBUFMH)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_MBUFMH register defines the fault mapping, in terms of Unrecoverable or Recoverable fault, of the MBIST in the range NMCUT = 64-95.

The size of the register depends on the number of BISTed RAMs/ROMs.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 7Ch offset = 7Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_MBUFMH field descriptions

Field	Description
0–31 Reserved	This field is reserved.

79.7.31 STCU2 LBIST Control Register (STCU2_LB_CTRL*n*)

NOTE

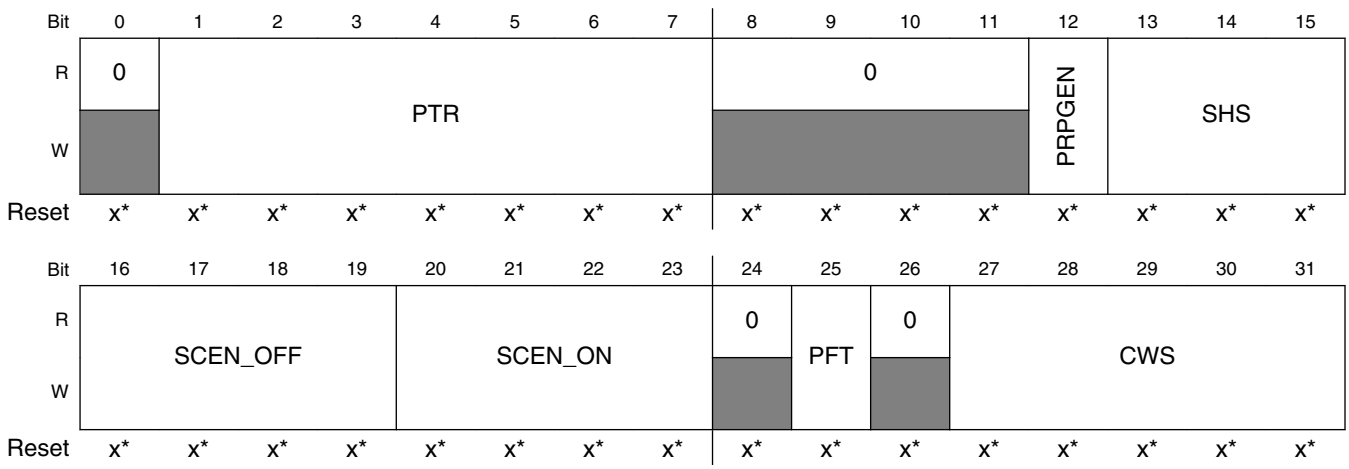
The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_CTRL register defines the control setting of each LBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 100h offset + (64d × i), where i=0d to 3d



* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_CTRL*n* field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 PTR	Next LBIST pointer PTR defines the logical pointer to the next LBIST. In case of NIL pointer the CSM bit has to be set Sequential (0) to define this is the end of the list. The self testing procedure is stopped when the current LBIST has been completed. LBIST POINTERS 0000b - pointer to LBIST partition 0 0001b - pointer to LBIST partition 1 0010b - pointer to LBIST partition 2 0011b - pointer to LBIST partition 3 ... 0x7F - pointer to NIL, end of LBIST execution Values not defined in this list will result in an error being set in the STCU2_ERR register.
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

STCU2_LB_CTRL*n* field descriptions (continued)

Field	Description
12 PRPGEN	Enable PRPG Loading 0 Default LBIST value of the PRPG is used during LBIST run 1 LBIST PRPG value is initialized with the STCU2_LB_PRPGL/H
13–15 SHS	Shift speed SHS defines the shift speed 000 Shift at full rate of STCU2 core clock 001 Shift at 1/2 rate of STCU2 core clock 010 Shift at 1/3 rate of STCU2 core clock 011 Shift at 1/4 rate of STCU2 core clock 100 Shift at 1/5 rate of STCU2 core clock 101 Shift at 1/6 rate of STCU2 core clock 110 Shift at 1/7 rate of STCU2 core clock 111 Shift at 1/8 rate of STCU2 core clock
16–19 SCEN_OFF	Scan enable OFF SCEN_OFF defines the number of clock cycles OFF following the falling transition on the SCEN **NOTE:** Scen_off must be programmed to a value >=1 0000 0 delay cycles 0001 1 delay cycle 0010 2 delay cycles 0011 3 delay cycles 0100 4 delay cycles 0101 5 delay cycles 0110 6 delay cycles 0111 7 delay cycles 1000 8 delay cycles 1001 9 delay cycles 1010 10 delay cycles 1011 11 delay cycles 1100 12 delay cycles 1101 13 delay cycles 1110 14 delay cycles 1111 15 delay cycles
20–23 SCEN_ON	Scan enable ON SCEN_ON defines the number of clock cycles OFF following the rising transition on the SCEN **NOTE:** Scen_on delay register value must be programmed to a value >=1> 0000 0 delay cycles 0001 1 delay cycle 0010 2 delay cycles 0011 3 delay cycles 0100 4 delay cycles 0101 5 delay cycles 0110 6 delay cycles

Table continues on the next page...

STCU2_LB_CTRL*n* field descriptions (continued)

Field	Description
	0111 7 delay cycles 1000 8 delay cycles 1001 9 delay cycles 1010 10 delay cycles 1011 11 delay cycles 1100 12 delay cycles 1101 13 delay cycles 1110 14 delay cycles 1111 15 delay cycles
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 PFT	Past Flush Test The LBIST controller, by default, applies 32 Flush Test patterns. This bit allows to skip/apply the Flush test pattern sequence.
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 CWS	Capture window size CWS defines the capture window size. 00000: Illegal 00001: Controller waits 1 shift cycle for capture to finish. 00010: Controller waits 2 shift cycles for capture to finish. 00011: Controller waits 3 shift cycles for capture to finish. ... 11110: Controller waits 30 shift cycles for capture to finish. 11111: Controller waits 31 shift cycles for capture to finish.

79.7.32 STCU2 LBIST PC Stop Register (STCU2_LB_PCS*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_PCS register defines the pattern counter stop of each LBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 104h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						PCS																									
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_PCS*n* field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–31 PCS	Pattern counter stop PCS defines the pattern counter stop value.

79.7.33 STCU2 LBIST PRPG Low Register (STCU2_LB_PRPGL*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_PRPGL register defines the LSB part (31-0) of the PRPG start value of the LBIST controller.

The size of the register depends on the number of PRPG bits of the related LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 108h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PRPGx																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_PRPGL*n* field descriptions

Field	Description
0–31 PRPGx	PRPG LBIST Low Bits This field defines the Low part (31-0) of the LBIST PRPG start value.

79.7.34 STCU2 LBIST PRPG High Register (STCU2_LB_PRPGH*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_PRPGH register defines the MSB part (63-32) of the PRPG start value of the LBIST controller.

The size of the register depends on the number of PRPG bits of the related LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 10Ch offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PRPGx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_PRPGH*n* field descriptions

Field	Description
0–31 PRPGx	PRPG LBIST High Bits This field is the High part (63-32) of the LBIST PRPG start value.

79.7.35 STCU2 Off-Line LBIST MISR Expected Low Register (STCU2_LB_MISREL*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISREL register defines the LSB part (31-0) of the Expected MISR of the Off-Line LBIST controller.

The size of the register depends on the number of MISR bits of the related LBIST.

The R/W fields in this register are readable at any time. However, you can write to these fields only when Off-line Self-Test phase is still active.

Address: 0h base + 110h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISREx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISREL*n* field descriptions

Field	Description
0–31 MISREx	Off-Line MISR Expected low Bits This field defines the low part (31..0) of the Expected MISR.

79.7.36 STCU2 Off-Line LBIST MISR Expected High Register (STCU2_LB_MISREH*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISREH register defines the MSB part (63-32) of the Expected MISR of the Off-Line LBIST controller.

The size of the register depends on the number of MISR bits of the related LBIST.

The R/W fields in this register are readable at any time. However, you can write to these fields only when Off-line Self-Test phase is still active.

Address: 0h base + 114h offset + (64d × i), where i=0d to -1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISREx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISREH*n* field descriptions

Field	Description
0–31 MISREx	Off-Line MISR Expected high Bits This field defines the high part (63-32) of the Expected MISR.

79.7.37 STCU2 Off-Line LBIST MISR Read Low Register (STCU2_LB_MISRRL*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISRRL register reports the LSB part (31-0) of the MISR obtained at the end of the Off-Line LBIST controller execution.

The size of the register depends on the number of MISR bits of the related LBIST.

Address: 0h base + 118h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRRx																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISRRL*n* field descriptions

Field	Description
0–31 MISRRx	Off-Line MISR Read Low Bits This field is equivalent to the Low Bits (31-0) of the MISR obtained at the end of the Off-Line LBIST execution

79.7.38 STCU2 Off-Line LBIST MISR Read High Register (STCU2_LB_MISRRH*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISRRH register report the MSB part (63-32) of the MISR obtained at the end of each Off-Line LBIST controller execution.

The size of the register depends on the number of MISR bits of the related LBIST.

Address: 0h base + 11Ch offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRRx																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISRRH*n* field descriptions

Field	Description
0–31 MISRRx	Off-Line MISR Read High Bits This field is equivalent to the High Bits (63-32) of the MISR obtained at the end of the Off-Line LBIST execution

79.7.39 STCU2 On-Line LBIST MISR Expected Low Register (STCU2_LB_MISRELSW*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISRELSW register defines the LSB part (31-0) of the Expected MISR of the On-Line LBIST controller.

The size of the register depends on the number of MISR bits of the related LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 120h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISRESWx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISRELSW*n* field descriptions

Field	Description
0–31 MISRESWx	On-Line MISR Expected low Bits This field defines the low part (31..0) of the Expected MISR.

79.7.40 STCU2 On-Line LBIST MISR Expected High Register (STCU2_LB_MISREHSW*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISREHSW register defines the MSB part (63-32) of the Expected MISR of the On-Line LBIST controller.

The size of the register depends on the number of MISR bits of the related LBIST.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 124h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISRESWx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISREHSW*n* field descriptions

Field	Description
0–31 MISRESWx	On-Line MISR Expected high Bits This field defines the high part (63-32) of the Expected MISR.

79.7.41 STCU2 On-Line LBIST MISR Read Low Register (STCU2_LB_MISRRLSW*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISRRLSW register report the LSB part (31-0) of the MISR obtained at the end of the On-Line LBIST controller execution.

The size of the register depends on the number of MISR bits of the related LBIST.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 128h offset + (64d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MISRRSWx																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISRRLSW*n* field descriptions

Field	Description
0–31 MISRRSWx	On-Line MISR Read Low Bin This field is equivalent to the Low Bits (31-0) of the MISR obtained at the end of the Off-Line LBIST execution.

79.7.42 STCU2 On-Line LBIST MISR Read High Register (STCU2_LB_MISRRHSW*n*)

NOTE

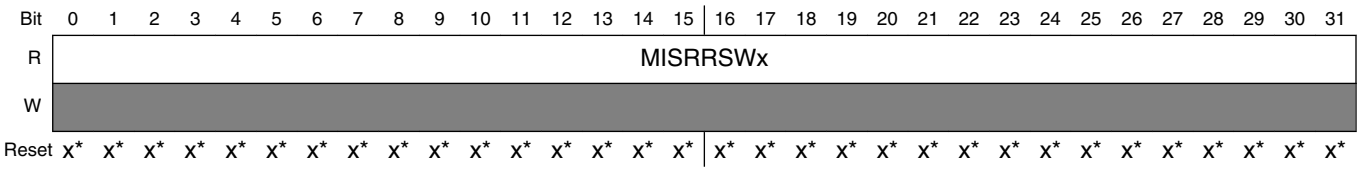
The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_LB_MISRRHSW register report the MSB part (63-32) of the MISR obtained at the end of each On-Line LBIST controller execution.

The size of the register depends on the number of MISR bits of the related LBIST.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 12Ch offset + (64d × i), where i=0d to 3d



* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_LB_MISRRHSW*n* field descriptions

Field	Description
0–31 MISRRSWx	On-Line MISR Read High Bits This field is equivalent to the Low Bits (63-32) of the MISR obtained at the end of the Off-Line LBIST execution.

79.7.43 STCU2 MBIST Control Register (STCU2_MB_CTRL*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

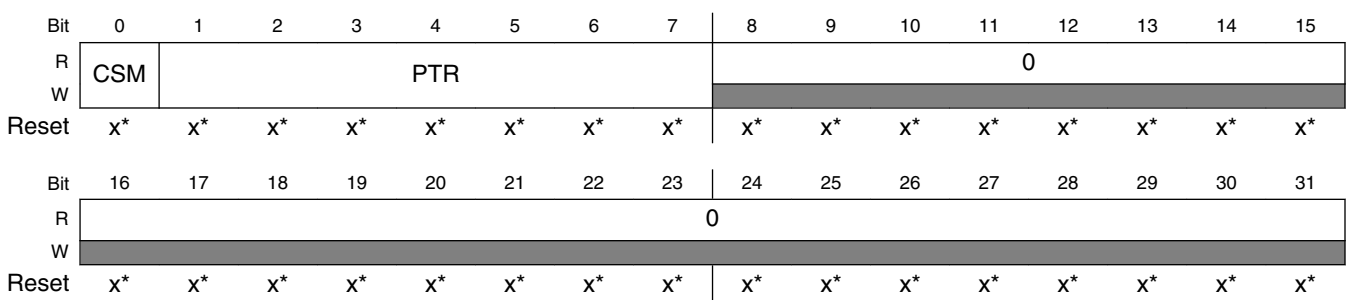
The STCU2_MB_CTRL register defines the control setting of MBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

The offset of this register is a function of NMCUT .

Address: 0h base + 600h offset + (4d × i), where i=0d to 46d



* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_MB_CTRL*n* field descriptions

Field	Description
0 CSM	Concurrent/sequential mode 0 Sequential mode 1 Concurrent mode
1–7 PTR	Next LBIST or MBIST pointer PTR defines the logical pointer to the next LBIST or MBIST to be scheduled. The next MBIST is scheduled concurrently to the current one if the CSM bit is set to 1; otherwise it is scheduled sequentially to the completion of the current MBIST execution. In case of NIL pointer the CSM bit must be set Sequential (0) to define this is the end of the list. The self-testing procedure is stopped when the current MBIST has been completed. 0h to (LBIST -1): pointer to LBIST 10h to (10h + MBIST -1b): pointer to MBIST 7Fh: pointer to NIL. No BIST execution. others: invalid pointer => an error is set into the STCU2_ERR register.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

79.8 Use cases and limitations

The STCU2 module is designed to give high flexibility during the on/off-line test. In the following are reported the programming operations that have to be performed to correctly use the functionality of this module considering the three typical usage cases:

- the Off-line Self Test Sequence after a reset trigger event
- the bypass mode of the Self Test Sequence after a reset trigger event
- the On-line Self-test Sequence

79.8.1 Offline self-test sequence

This is the typical mode of using the STCU2 module after reset trigger event is applied to STCU2. The SSCM DCF bus is used to retrieve the STCU2 schedule and L/MBIST execution parameters stored into the NVM memory.The target is to cover the amount of physical defects in the digital logic and in the System RAMs/ROMs according to the System specifications. The sequence is the following:

- Unlock the Off-Line STCU2 access writing the Key1/Key2 sequence into the STCU2_SKC register
- Program the LBIST fault reaction conditions (UF/RF) setting the STCU2_LBUFM register

- Program the MBIST fault reaction conditions (UF/RF) setting, depending on the number of memories, the STCU2_MBUFML/M/H registers
- Program the STCU2 internal fault reaction conditions (UF/RF) setting the STCU2_ERR_FM register
- Program the STCU2_LB_CTRL, STCU2_LB_PCS, STCU2_MISREL/H registers of each LBIST to be executed. Optionally, also the STCU2_LB_PRPGL/H can be programmed in order to load a specific seed.
- Program the STCU2_MB_CTRL registers of each NMCUT to be executed
- Program the STCU2_WDG register to define the time budget assigned for the L/MBIST execution
- Program the STCU2_CFG register to define: the core and LBIST/MBIST TCK clock prescaling factor setting the CLK_CFG bits, the PMOSEN or MBU in order to define the algorithm to be used during MBIST run, and finally set the pointer to the first LBIST/NMCUT to be executed
- In case the PLL usage has been enabled, program the STCU2_PLL_CFG value in order to set the required system frequency
- Program the RUN bit into the STCU2_RUN register to enable the Off-Line Self-Test execution and the L/MBPLLEN bits to enable the PLL during L/MBIST execution according to STCU2_PLL_CFG register content
- Start the Off-Line Self-Test and, depending on PLL management and L/MBIST execution order, when PLL LOCK is high.
- Wait the WDGEOC run time execution specified into the STCU2_WDG register.
- The STCU2 switchs off the core clock at the end of the Self-Test run and releases the PLL control in case this feature has been enabled.
- The STCU2 resets the STCU2_RUN[RUN] bit and the device exits from boot sequence. In case there are failures, the STCU2 flags these failures toward the FCCU.
- The software application reads the STCU2_LBS flag register to check the failed LBIST when UF or RF conditions is/are detected.
- The software application reads the STCU2_LBE flag register to check the LBIST still on-going when UF or RF is/are detected.
- The software application reads the STCU2_MBSL/M/H flag registers, depending on the number of memories, to check the failed RAMs/ROMs Memory BIST when UF or RF is/are detected.
- The software application reads the STCU2_MBEL/M/H flag registers, depending on the number of memories, to check if the RAMs/ROMs Memory BIST is still on-going when UF or RF is/are detected.

- The software application reads the bits INV, ENGE, WDTO, LOCKE of the STCU2_ERR_STAT register to check whether there has been an internal STCU2 engine/parameters failure when UF or RF is/are detected.
- The software application reads for each device's LBIST, the STCU2_LBMISREL/H and STCU2_LBIST_MISRRL/H registers to check the coherence with the STCU2_LBS bits.

NOTE

The software application reads operations on the STCU2 and LBIST MISR registers after off-line self test execution has been successfully performed are just an additional reliability layer added to improve the already implemented Auto Self-Test feature included into STCU2.

79.8.2 Online self-test sequence

This is the typical mode of using the STCU2 module during the application run. The IPS interface is used to program the STCU2 registers and to schedule the L/MBIST execution. Since the On-Line L/MBIST operations clean-up the content of the related Digital Logic (LBIST) or RAM (MBIST) the activation of this mode has to be done carefully. The suggested sequence is the following:

- Unlock the On-Line STCU2 access writing the Key1/Key2 sequence into the STCU2_SKC register
- Check and/or clean-up the bit WRP into the STCU2_CFG register to open the IPS access on the STCU2 On-Line Self-Test registers
- The LBIST, MBIST and STCU2 internal fault reaction should have been already programmed during the Off-Line Self-Test sequence and so they should not be programmed again. However, in case they have not been programmed or there is the need of changing them for any reason, they can be overwritten setting respectively the registers: STCU2_LBUFM, STCU2_MBUFML/M/H and STCU2_ERR_FM.
- Program the Reset management during on-Line LBIST execution setting the STCU2_LBRMSW register
- The STCU2_LB_CTRL, STCU2_LB_PCS should have been already programmed during Off-Line Self-test sequence and so they should not be programmed again. However, in case they have not been programmed or there is the need of changing

them for any reason, they can be overwritten. The registers STCU2_LB_MISRELSW/HSW have to be programmed according to the expected signature.

- Overwrite/Program the STCU2_MB_CTRL registers of each NMCUT to be executed
- Overwrite/Program the STCU2_WDG register to define the time budget assigned for the L/MBIST execution
- Overwrite/Program the STCU2_CFG register in order to define: the core and L/MBIST TCK clock prescaling factor setting the CLK_CFG bits, the PMOSEN or MBU in order to define the algorithm to be used during MBIST run, and finally set the pointer to the first LBIST/NMCUT to be executed.
- Program the RUNSW bit into the STCU2_RUNSW register to enable the On-Line Self-Test execution; the L/MBSWPLLEN bits to enable the PLL lock signal monitor during L/MBIST execution and the L/MBIE to enable the related Interrupt requests.
- Start the On-Line Self-Test and depending on PLL management and L/MBIST execution order, also when PLL LOCK is high.
- Wait the WDGEOC run time execution specified into the STCU2_WDG register
- In case the L/MBIE have been enabled, an interrupt is generated at the end of all the L/MBIST execution. The software application has to manage these interrupt signals and clean-up the related flag (L/MBIFLG) into the STCU2_INT_FLG register.
- The STCU2 switch-off the core clock at the end of the Self-Test run
- The functional reset activated is managed by the STCU2_LBRMSW.
- In case, at least one of the selected LBIST enables the Global functional reset, the STCU2 requests a functional reset which restarts the system while, when all the selected LBIST enables its own dedicated functional reset, only these local reset lines are applied and only the related LBIST partitions are cleaned-up.
- The STCU2 resets the STCU2_RUNSW[RUNSW] bit. In both the reset cases and in case of fault/s, the STCU2 flags the failure toward the FCCU.
- The software application reads the STCU2_LBSSW flag register to check the failed LBIST when UF or RF conditions is/are detected
- The software application reads the STCU2_LBESW flag register to check if the LBIST is still on-going when UF or RF is/are detected

- The software application reads the STCU2_MBSLSW/MSW/HSW flag registers, depending on the number of memories, to check the failed RAMs/ROMs Memory BIST when UF or RF is/are detected
- The software application reads the STCU2_MBELSW/MSW/HSW flag registers, depending on the number of memories, to check if the RAMs/ROMs Memory BIST is still on-going when UF or RF is/are detected
- The software application reads the bit INVPSW, ENGESW, WDTOSW, LOCKESW of the STCU2_ERR_STAT register to check whether there has been an internal STCU2 engine/parameters failure when UF or RF is/are detected
- The software application reads for each device's LBIST, the STCU2_LBMISRELSW/HSW and STCU2_LBIST_MISRRLSW/HSW registers to check the coherence with the STCU2_LBS bits

NOTE

The software application read operations on the STCU2 and LBIST MISR registers after on-line self-test execution has been successfully performed are just an additional reliability layer to be added to improve the already implemented Auto Self-Test feature included in the STCU2.

NOTE

Flash memory must be in the idle state when BIST execution is started. Program and erase functionality must not be enabled when entering BIST execution.

79.8.3 Bypass USER Mode

In this case, the USER application parameters stored in FLASH and read by SSCM are written in order to skip the Self-Test procedure after a reset trigger event is applied. It might be useful in case the device is not configured for applications requiring improved reliability. The sequence is the following:

1. Unlock the STCU2 access writing the Off-Line Key1/Key2 sequence into the STCU2_SKC register
2. Set the BYP bit in the STCU2_RUN register

After BYP bit is set, the core clock is switched off, the Self-Test sequence is not applied and the system can wake up and start the user application.

79.8.4 Design implementation information

The following list reports the limitations related to the current implementation:

- Every NMCUT BIST can be run once during the online or offline self-test procedure.
- It is not possible to run LBIST and MBIST concurrently.
- Because software has full control of the system while applications are running during the online self-test, STCU2 is not allowed to take control of the PLL but can just monitor the lock signal to flag wrong unlock events.

CAUTION

Do not execute an LBIST on more than one partition at a time.

Chapter 80
Register Protection (REG_PROT)

80.1 Register protection (REG_PROT) chip-specific module information

Some modules on this device include a built-in register protection mechanism that can be used to override the normal write permissions associated with individual registers in the module. This mechanism is software-controlled and can be dynamically reconfigured at runtime or made effective until device reset.

Table 80-1. Modules with protected registers

Topic	Related module
SIUL2 registers	SIUL2
CMU registers	CMU_0 (pll), CMU_1 (fxbar), CMU_2 (sxbar), CMU_3 (aips) and CMU_4 (per)
PMC registers	PMC
PLLDIG registers	PLLDIG
XOSC registers	XOSC
MEMU registers	MEMU
MC_ME registers	MC_ME
MC_CGM registers	MC_CGM
MC_RGM registers	MC_RGM
SSCM registers	SSCM
JTAGM registers	JTAGM

The protected registers are listed in subsequent sections.

80.1.1 SIUL2 protected registers

Table 80-2. Protected SIUL2 registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
DIRER0	32	18h	32
DIRSR0	32	20h	32
IREER0	32	28h	32
IFEER0	32	30h	32
MSCR0 – MSCR13	32	240h	32 × 14
MSCR16 – MSCR17	32	284h	32 × 2
MSCR32 – MSCR45	32	—[1, 1]	32 × 14
MSCR48 – MSCR64	32	—[1]	32 x 17
MSCR80 – MSCR93	32	—[1]	32 × 14
MSCR97 – MSCR103	32	—[1]	32 x 7
MSCR105 – MSCR133	32	—[1]	32 x 29
MSCR144 – MSCR162	32	—[1]	32 × 19
MSCR164 – MSCR165	32	—[1]	32 × 2
MSCR167 – MSCR174	32	—[1]	32 x 8
MSCR176 – MSCR177	32	—[1]	32 × 2
MSCR192 – MSCR195	32	—[1]	32 × 4
MSCR208 – MSCR255	32	—[1]	32 × 48
MSCR512 – MSCR585	32	—[1]	32 × 74
MSCR590 – MSCR609	32	—[1]	32 × 20
MSCR616 – MSCR635	32	—[1]	32 × 20
MSCR640 – MSCR694	32	—[1]	32 × 55
MSCR700 – MSCR703	32	—[1]	32 × 4
MSCR712 – MSCR717	32	—[1]	32 × 6
MSCR752 – MSCR757	32	—[1]	32 × 6
MSCR770 – MSCR773	32	—[1]	32 × 4
MSCR775 – MSCR778	32	—[1]	32 x 4
MSCR803	32	—[1]	32
MSCR809	32	—[1]	32
MSCR812 – MSCR813	32	—[1]	32 × 2
MSCR815 – MSCR816	32	—[1]	32 × 2
MSCR840 – MSCR935	32	—[1]	32 × 96
MSCR960 – MSCR962	32	—[1]	32 × 3
MSCR968 – MSCR970	32	—[1]	32 × 3
MSCR992 – MSCR996	32	—[1]	32 × 5
MSCR998	32	—[1]	32
GPDO0 – GPDO255[2]	8	1300h	8 × 256
PGPDO0 – PGPDO15	16	1700h	16 × 16
MPGPDO0 – MPGPDO15	32	1780h	32 × 16

1. See the module memory map in the SIUL2 chapter for address offset.
2. Register protection is implemented on GPDO registers that correspond to implemented pads.

80.1.2 CMU protected registers

Table 80-3. Protected CMU registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CSR	32	0h	8 (byte at address 0x03)
HFREFR_A	32	8h	32
LFREFR_A	32	Ch	32
MDR	32	18h	32

80.1.3 PMC_DIG protected registers

Table 80-4. Protected PMC_DIG registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
TS_IER	32	A4h	32
REE_TD	32	ACh	32
RES_TD	32	B0h	32
CTL_TD	32	B4h	32
TS_FIR	32	C0h	32
FIR	32	D4h	32
PMC_VD_INTR_ENB	32	D8h	32
PMC_REE	32	E0h	32
PMC_RES	32	E4h	32
ADC_CS	32	E8h	32

80.1.4 PLL_DIG protected registers

Table 80-5. Protected PLL_DIG registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
PLL0_CR	32	0h	32
PLL0_DIVR	32	8h	32
PLL1_CR	32	20h	32

Table continues on the next page...

Table 80-5. Protected PLL_DIG registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
PLL1_DIVR	32	28h	32
PLL1_FMR	32	2Ch	32
PLL1_FDR	32	30h	32

80.1.5 OSC_DIG protected registers

Table 80-6. Protected OSC_DIG registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
OSC_CTL	32	0h	32

80.1.6 MEMU protected registers

Table 80-7. Protected MEMU registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CTRL	32	0h	8 (byte at address 0x00)
DEBUG	32	Ch	32

80.1.7 MC_ME protected registers

Table 80-8. Protected MC_ME registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
ME_MCTL	32	4h	32
ME_ME	32	8h	32
ME_IM	32	10h	32
ME_SAFE_MC	32	28h	32
ME_DRUN_MC	32	2Ch	32
ME_RUN0_MC	32	30h	32
ME_RUN1_MC	32	34h	32
ME_RUN2_MC	32	38h	32

Table continues on the next page...

Table 80-8. Protected MC_ME registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
ME_RUN3_MC	32	3Ch	32
ME_RUN_PC0 – ME_RUN_PC7	32	80 – 90h	32 × 8
ME_LP_PC0 – ME_LP_PC7	32	A0 – BFh	32 × 8
ME_PCTL9	8	CEh	8 × 1
ME_PCTL11	8	D0h	8 × 1
ME_PCTL28 – ME_PCTL31	8	E1 – E4h	8 × 4
ME_PCTL36	8	E8h	8 × 1
ME_PCTL38	8	EAh	8 × 1
ME_PCTL59 – ME_PCTL60	8	FF – 100h	8 × 2
ME_PCTL69	8	109h	8 × 1
ME_PCTL71	8	10Bh	8 × 1
ME_PCTL81	8	115h	8 × 1
ME_PCTL83	8	117h	8 × 1
ME_PCTL85	8	119h	8 × 1
ME_PCTL91 – ME_PCTL92	8	11Fh – 120h	8 × 2
ME_PCTL96 – ME_PCTL99	8	124 – 127h	8 × 4
ME_PCTL102	8	12Ah	8 × 1
ME_PCTL104	8	12Ch	8 × 1
ME_PCTL143	8	153h	8 × 1
ME_PCTL166	8	16Ah	8 × 1
ME_PCTL188	8	180h	8 × 1
ME_PCTL196 – ME_PCTL198	8	188h – 18Ah	8 × 3
ME_PCTL202 – ME_PCTL208	8	18E – 194h	8 × 7
ME_PCTL210	8	196h	8 × 1
ME_PCTL213	8	199h	8 × 1
ME_PCTL219 – ME_PCTL220	8	19F – 1A0h	8 × 2
ME_PCTL225 – ME_PCTL227	8	1A5h – 1A7h	8 × 3
ME_PCTL230	8	1AAh	8 × 1
ME_PCTL232	8	1ACh	8 × 1
ME_PCTL251	8	1BFh	8 × 1
ME_CCTL0 – ME_CCTL2	16	1C4h – 1D2h	16 × 3

80.1.8 MC_CGM protected registers

Table 80-9. Protected MC_CGM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CGM_SC_DC[0..2]	32	7E8 – 7F0h	32
CGM_AC[0..6, 8..10]_SC	32	800h – 940h	8 (byte at lowest address) × 11
CGM_AC0_DC[0..4]	32	808 – 818h	32
CGM_AC1_DC0	32	828h	32
CGM_AC2_DC0	32	838h	32
CGM_AC5_DC[0,1]	32	8A8 – 8ACh	32
CGM_AC[6, 8..10]_DC0	32	8C8 – 948h	32

80.1.9 MC_RGM protected registers

Table 80-10. Protected MC_RGM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
DERD	32	10h	32
DEAR	32	20h	32
DBRE	32	30h	32
FERD	32	310h	32
FEAR	32	320h	32
FBRE	32	330h	32
FESS	32	340h	32

80.1.10 SSCM protected registers

Table 80-11. Protected SSCM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
ERROR	16	6h	16

80.1.11 JTAGM protected registers

Table 80-12. Protected JTAGM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
MCR	32	0h	32

80.2 Overview

The Register Protection (REG_PROT) module offers a mechanism to protect defined memory-mapped address locations in a module under protection from being written. The address locations that can be protected are module-specific.

The protection module is located between the module under protection and the peripheral bridge (PBRIDGE/AIPS-Lite). This is shown in the following figure.

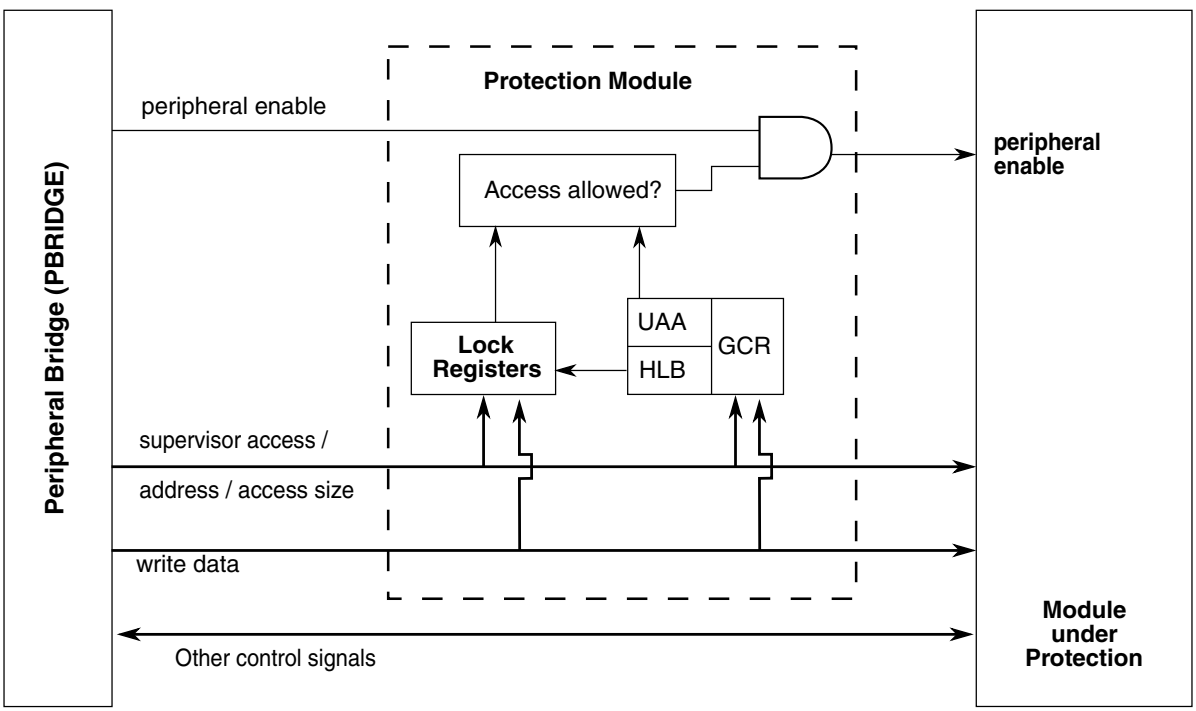


Figure 80-1. REG_PROT Block Diagram

80.3 Features

Register Protection includes these distinctive features:

- Restrict write accesses for the module under protection to supervisor mode only
- Lock registers for first 6 KB of memory-mapped address space
- Write to address mirror automatically sets corresponding lock bit
- Once configured lock bits can be protected from changes

80.4 Modes of operation

The Register Protection module is operable when the module under protection is operable.

80.5 External signal description

There are no external signals.

80.6 Memory map and register definition

This section provides a detailed description of the memory map of a module with register protection. The original 16 KB module memory space is divided into five areas as shown in the following figure.

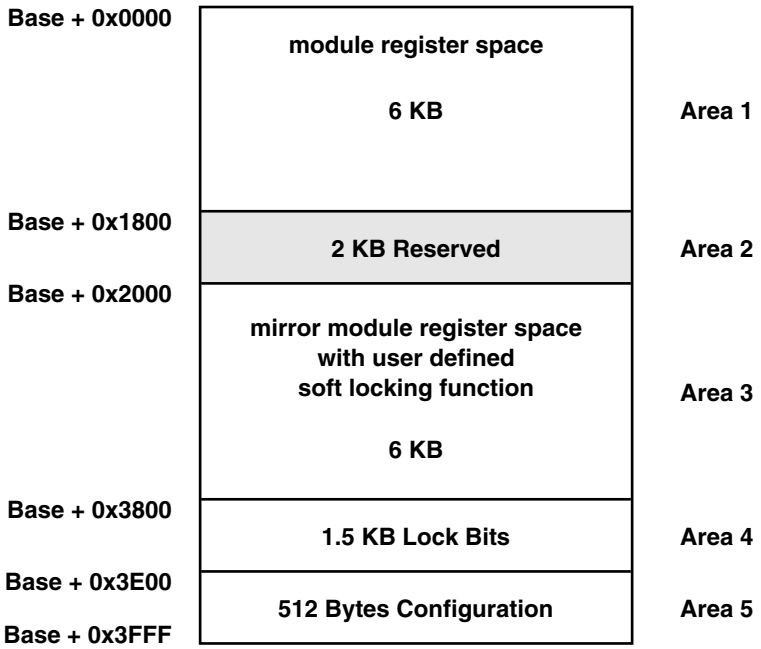


Figure 80-2. REG_PROT Memory Diagram

- Area 1 is 6 KB and holds the normal functional module registers and is transparent for all read/write operations.
- Area 2 is 2 KB starting at offset 0x1800 is a reserved area, which shall not be accessed.
- Area 3 is 6 KB, starting at offset 0x2000 and is a mirror of area 1. A read/write access to an offset 0x2000+X will read/write the register at offset X. However a write access to offset 0x2000+X will additionally set the optional Soft Lock Bits for this offset X in the same cycle as the register at offset X is written. This provides for an automatic write and lock operation. Not all registers in area 1 need to have protection defined by associated Soft Lock Bits. For unprotected registers at offset Y, accesses to offset 0x2000+Y will be identical to accesses at offset Y.
- Area 4 is 1.5 KB and holds the Soft Lock Bits, one bit per byte in area 1. The four Soft Lock Bits associated with one module register word are arranged at byte boundaries in the memory map. The Soft Lock Bit registers can be directly written using a bit mask.
- Area 5 is 512 bytes large and holds the configuration bits of the protection mode. There is one configuration hard lock bit per module that prevents all further modifications to the Soft Lock Bits and can only be cleared by a system reset once set. The other bits, if set, will allow user access to the protected module.

If any locked byte is accessed with a write transaction, a transfer error will be issued to the system and the write transaction will not be executed. This is true even if not all accessed bytes are locked.

Accessing unimplemented 32-bit registers in Areas 4 and 5 will result in a transfer error.

80.6.1 Memory map

Following is the memory map for a module which is protected via a REG_PROT module.

Table 80-13. Generic Protected Module Memory Map

Offset	Use
0x0000	Module Register 0 (MR0)
0x0001	Module Register 1 (MR1)
0x0002	Module Register 2 (MR2)
0x0003 - 0x17FF	Module Register 3 (MR3) - Module Register 6143(MR6143)
0x1800 - 0x1FFF	Reserved
0x2000	Module Register 0 (MR0) + Set Soft Lock Bit 0 (LMR0)
0x2001	Module Register 1 (MR1) + Set Soft Lock Bit 1 (LMR1)
0x2002 - 0x37FF	Module Register 2 (MR2) + Set Soft Lock Bit 2 (LMR2) - Module Register 6143 (MR6143) + Set Soft Lock Bit 6143 (LMR6143)
0x3800	Soft Lock Bit Register 0 (SLBR0): Soft Lock Bits 0-3
0x3801	Soft Lock Bit Register 1 (SLBR1): Soft Lock Bits 4-7
0x3802 - 0x3DFF	Soft Lock Bit Register 2 (SLBR2): Soft Lock Bits 8-11 -Soft Lock Bit Register 1535 (SLBR1535): Soft Lock Bits 6140-6143
0x3E00 - 0x3FFB	Reserved
0x3FFC	Global Configuration Register (GCR)

Note

Reserved registers in area #1 will be handled as specified in the protected module's documentation.

80.6.2 Register descriptions

This section describes in address order all the REG_PROT registers. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

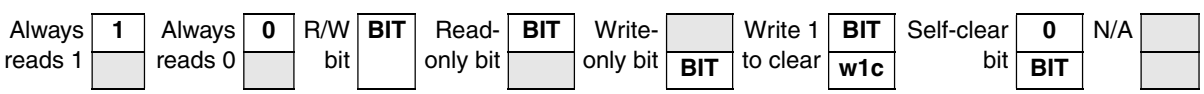


Figure 80-3. Register description

80.6.2.1 Module Registers (MR0-6143)

This is the lower 6 KB module memory space which holds all the functional registers of the module that is protected by the REG_PROT module.

80.6.2.2 Module register and set soft lock bit (LMR0-6143)

This is memory area #3 that provides mirrored access to the MR0-6143 registers with the side effect of setting Soft Lock Bits in case of a write access to a MR that is defined as protectable by the locking mechanism. Each MR is protectable by one associated bit in a SLBRn.SLBm, according to the mapping described in the Soft Lock Bit Register section.

80.7 Memory map and registers

REG_PROT memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3800	Soft Lock Bit Register n (REG_PROT_SLBRn)	8	R/W	00h	80.7.1/3463
3FFC	Global Configuration Register (REG_PROT_GCR)	32	R/W	0000_0000h	80.7.2/3465

80.7.1 Soft Lock Bit Register *n* (REG_PROT_SLBRn)

The Soft Lock Bit Registers hold the Soft Lock Bits for the protected registers in memory area #1, which is the normal register address space of the protected module. Each SLB register has a four Soft Lock Bits (SLB0-SLB3), each of which controls write access to a byte in memory area #1. Each Soft Lock Bit also has a corresponding Write Enable bit in the same register that controls whether the Soft Lock Bit can be written. The following table shows the mapping between the Soft Lock Bits to the bytes in memory area #1.

Table 80-16. Soft Lock Bits vs. Protected Address

Soft Lock Bit	Protected address
SLBR0.SLB0	MR0
SLBR0.SLB1	MR1
SLBR0.SLB2	MR2
SLBR0.SLB3	MR3
SLBR1.SLB0	MR4
SLBR1.SLB1	MR5
SLBR1.SLB2	MR6

Table continues on the next page...

Table 80-16. Soft Lock Bits vs. Protected Address (continued)

Soft Lock Bit	Protected address
SLBR1.SLB3	MR7
SLBR2.SLB0	MR8
...	...

NOTE

- As many as 1536 Soft Lock Bit Registers (SLBRn) may be implemented, depending on the number of protected module register bytes.
- Access in User Mode is read-only; in Supervisor Mode access is read/write.

Address: 0h base + 3800h offset = 3800h

Bit	0	1	2	3	4	5	6	7
Read	0	0	0	0	SLB0	SLB1	SLB2	SLB3
Write	WE0	WE1	WE2	WE3				
Reset	0	0	0	0	0	0	0	0

REG_PROT_SLBRn field descriptions

Field	Description
0 WE0	Write Enable Bits for Soft Lock Bits (SLB) WE0 enables writing to SLB0 1 Value is written to SLB 0 SLB is not modified
1 WE1	Write Enable Bits for Soft Lock Bits (SLB) WE1 enables writing to SLB1 1 Value is written to SLB 0 SLB is not modified
2 WE2	Write Enable Bits for Soft Lock Bits (SLB) WE2 enables writing to SLB2 1 Value is written to SLB 0 SLB is not modified
3 WE3	Write Enable Bits for Soft Lock Bits (SLB) WE3 enables writing to SLB3 1 Value is written to SLB 0 SLB is not modified
4 SLB0	Soft Lock Bits for one MRn register

Table continues on the next page...

REG_PROT_SLBRn field descriptions (continued)

Field	Description
	SLB0 can block accesses to MR[n *4 + 0] 1 Associated MRn byte is locked against write accesses 0 Associated MRn byte is unprotected and writable
5 SLB1	Soft Lock Bits for one MRn register SLB1 can block accesses to MR[n *4 + 1] 1 Associated MRn byte is locked against write accesses 0 Associated MRn byte is unprotected and writable
6 SLB2	Soft Lock Bits for one MRn register SLB2 can block accesses to MR[n *4 + 2] 1 Associated MRn byte is locked against write accesses 0 Associated MRn byte is unprotected and writable
7 SLB3	Soft Lock Bits for one MRn register SLB3 can block accesses to MR[n *4 + 3] 1 Associated MRn byte is locked against write accesses 0 Associated MRn byte is unprotected and writable

80.7.2 Global Configuration Register (REG_PROT_GCR)

NOTE

Access in User Mode is read-only; in Supervisor Mode access is read/write.

Address: 0h base + 3FFCh offset = 3FFCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HLB	0							UAA	0						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

REG_PROT_GCR field descriptions

Field	Description
0 HLB	Hard Lock Bit This register cannot be cleared once it is set by software. It can only be cleared by a system reset.

Table continues on the next page...

REG_PROT_GCR field descriptions (continued)

Field	Description
	NOTE: Access in User Mode is read-only; in Supervisor Mode access is read/write. 1 All SLB bits are write protected and can not be modified. 0 All SLB bits are accessible and can be modified.
1–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 UAA	User Access Allowed. 1 The registers in the module under protection can be accessed in the mode defined for the module registers without any additional restrictions. 0 The registers in the module under protection can only be written in supervisor mode. All write accesses in non-supervisor mode are not executed and a transfer error is issued. This access restriction is in addition to any access restrictions imposed by the protected IP module.
9–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

80.8 Functional description

The following sections describe the functional characteristics of the Register Protection module.

80.8.1 General

This module provides a generic register (address) write-protection mechanism. The protection size can be:

- 32-bit (address == multiples of 4)
- 16-bit (address == multiples of 2)
- 8-bit (address == multiples of 1)

Which addresses are protected and the protection size depends on the SoC and/or module.

For all addresses (peripheral registers) that are protected there are SLBRn.SLBm bits that specify whether the address is locked. When an address is locked it can only be read but not written in any mode (supervisor/normal). If an address is unprotected the corresponding SLBRn.SLBm bit is always 0b0 no matter what software is writing to.

80.8.2 Change lock settings

To change the setting whether an address is locked or unlocked the corresponding SLBRn.SLBm bit needs to be changed. This can be done using the following methods:

- Modify the SLBRn.SLBm directly by writing to area #4
- Set the SLBRn.SLBm bit(s) by writing to the mirror module space (area #3)

Both methods are explained in the following sections.

80.8.2.1 Change lock settings directly via Area #4

In memory area #4 the lock bits are located. They can be modified by writing to them. Each SLBRn.SLBm bit has a mask bit SLBRn.WEm which protects it from being modified. This masking makes clear-modify-write operations unnecessary.

The following figure shows two modification examples. In the left example there is a write access to the SLBRn register specifying a mask value which allows modification of all SLBRn.SLBm bits. The example on the right specifies a mask which only allows modification of the bits SLBRn.SLB[3:1].

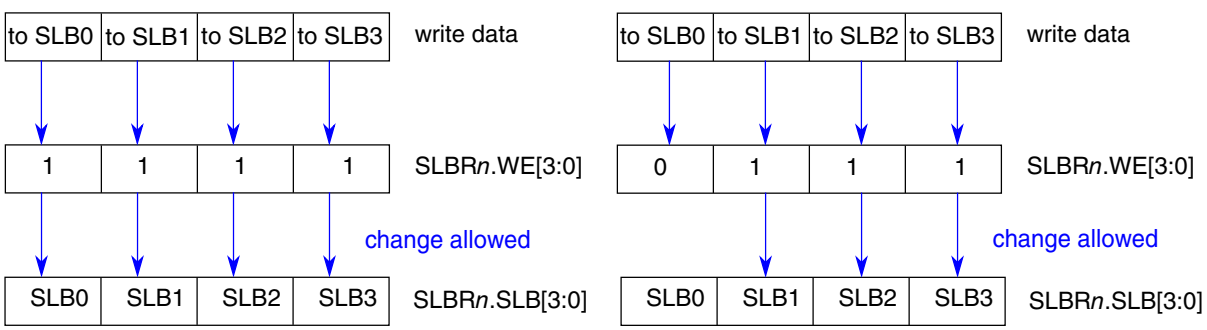


Figure 80-6. Change lock settings directly via Area #4

The previous figure shows four registers that can be protected 8-bit wise. The following figures show registers with 16- and 32-bit protection respectively:

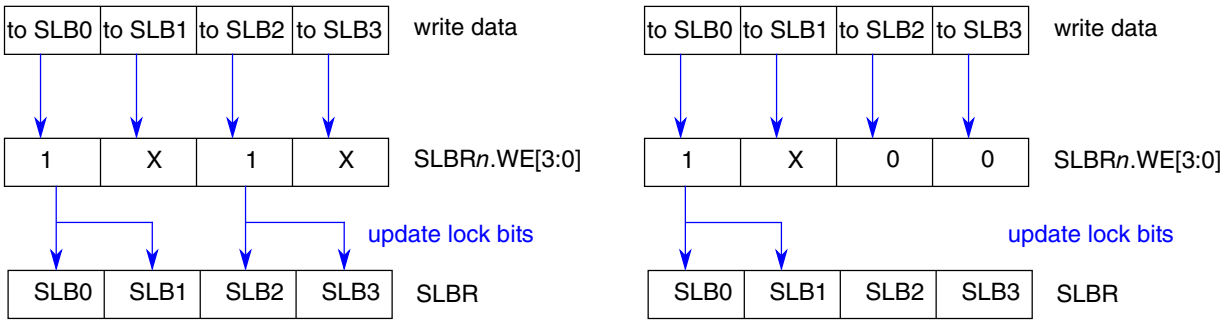


Figure 80-7. Change lock settings for 16-bit protected addresses

On the right side of the previous figure it is shown that the data written to SLBRn.SLB[0] is automatically written to SLBRn.SLB[1] also. This is done as the address reflected by SLBRn.SLB[0] is protected 16-bit wise. Note that in this case the write enable SLBRn.WE[0] must be set while SLBRn.WE[1] does not matter. As the enable bits SLBRn.WE[3:2] are cleared the lock bits SLBRn.SLB[3:2] remain unchanged.

In the example on the left side of the previous figure the data written to SLBRn.SLB[0] is mirrored to SLBRn.SLB[1] and the data written to SLBRn.SLB[2] is mirrored to SLBRn.SLB[3] as for both registers the write enables are set.

The next figure shows a 32-bit wise protected register. When SLBRn.WE[0] is set the data written to SLBRn.SLB[0] is automatically written to SLBRn.SLB[3:1] also. Otherwise SLBRn.SLB[3:0] remains unchanged.

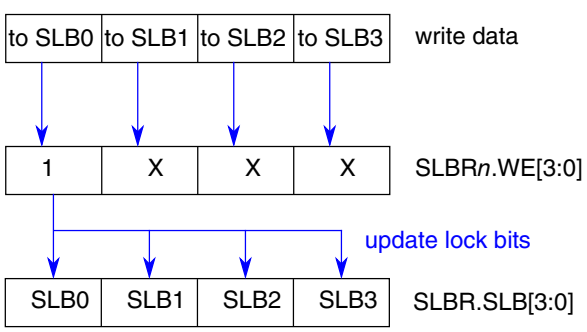


Figure 80-8. Change lock settings for 32-bit protected addresses

The following figure shows a mixed protection size configuration:

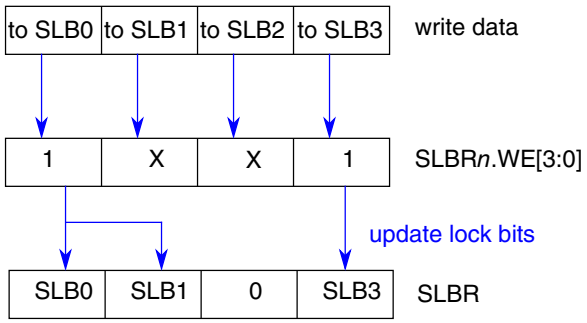


Figure 80-9. Change lock settings for mixed protection

The data written to SLBRn.SLB[0] is mirrored to SLBRn.SLB[1] as the corresponding register is 16-bit protected. The data written to SLBRn.SLB[2] is blocked as the corresponding register is unprotected. The data written to SLBRn.SLB[3] is written to SLBRn.SLB[3].

80.8.2.2 Enable locking via mirror module space (Area #3)

It is possible to enable locking for a register after writing to it. To do so the mirrored module address space must be used. For example:

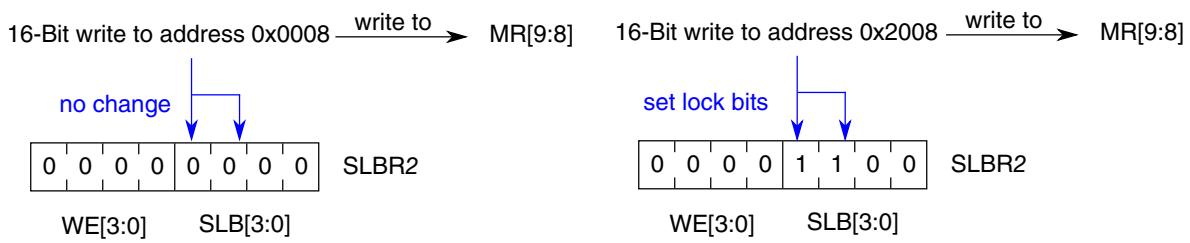


Figure 80-10. Enable Locking Via Mirror Module Space (Area #3)

When writing to address 0x0008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits remain unchanged (see the left part of Figure 80-7).

When writing to address 0x2008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits SLBR2.SLB[1:0] are set while the lock bits SLBR2.SLB[3:2] remain unchanged (right part of Figure 80-7).

The following figure shows an example where some addresses are protected and some are not:

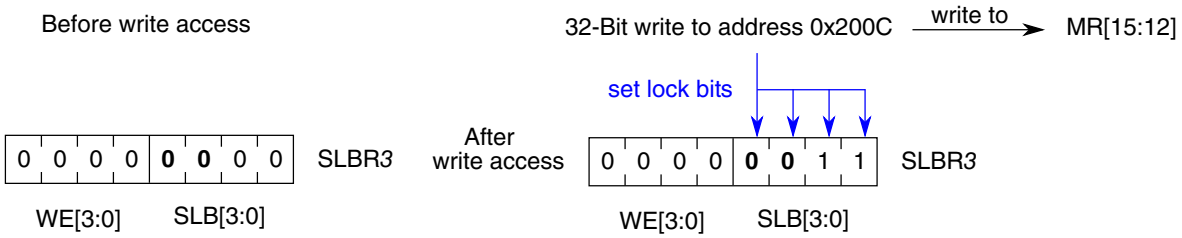


Figure 80-11. Enable locking for protected and unprotected addresses

In the previous figure addresses 0x0C and 0x0D are unprotected. Therefore their corresponding lock bits SLBR3.SLB[1:0] are always 0b0 (shown in bold). When doing a 32-bit write access to address 0x200C only lock bits SLBR3.SLB[3:2] are set while bits SLBR3.SLB[1:0] stay 0b0.

Note

Lock bits can only be set via writes to the mirror module space. Reads from the mirror module space will not change the lock bits.

80.8.2.3 Write protection for locking bits

Changing the locking bits through any of the procedures mentioned in Change lock settings directly via Area #4 and Enable locking via mirror module space (Area #3) is only possible as long as the bit GCR.HLB is cleared. Once this bit is set the locking bits can no longer be modified until there is a system reset.

80.8.3 Access errors

The protection module generates transfer errors under several circumstances. For the area definition refer to Figure 80-2

1. If accessing area #1 or area #3, the protection module will pass on any access error from the underlying Module under Protection.
2. If user mode is not allowed, user writes to all areas will assert a transfer error and the writes will be blocked.
3. If accessing the reserved area #2, a transfer error will be asserted.
4. If accessing unimplemented 32-bit registers in area #4 and area #5 a transfer error will be asserted.
5. If writing to a register in area #1 and area #3 with Soft Lock Bit set for any of the affected bytes a transfer error is asserted and the write will be blocked. Also the complete write operation to non-protected bytes in this word is ignored.

6. If writing to a Soft Lock Register in area #4 with the Hard Lock Bit being set a transfer error is asserted.
7. Any write operation in any access mode to area #3 while Hard Lock Bit GCR.HLB is set

Chapter 81
Security

81.1 Overview

The MCP574xR has a comprehensive set of customer-configurable security features designed to protect code and data from unauthorized access.

The security level for any MCP574xR microcontroller is dependent on a variety of factors, including configuration, level of maturity within the microcontroller's life cycle, and customer software design. This chapter gives an overview of available security features and how they interact.

81.1.1 Security Concepts

Table 81-1 shows the security concepts included in MCP574xR and their corresponding HW implementation. Most of these concepts are valid throughout the whole family. See Table 81-2 for a detailed description about what is available for each device of the MCP574xR.

Table 81-1. Security Concepts

Concepts	Description	HW Implementation
Secure system configuration	All the critical security-related data, which are transferred during the reset phase to their respective clients, are protected vs. manipulation	UTEST DCF Records parsed by SSCM See chapter Secure System Configuration
External access protection	Protection vs. access to the internal resources from an external tool	Debug access protection, Censorship, BAF Serial Boot See chapter External access protection
Flash memory protection	Protection vs. Flash memory access and modification	Censorship, PASS module, TDM module, Flash Sealing, See chapter Flash Memory Protection
End of life protection	Protection vs. the testing of the unit once it has been working in field	BAF execution See chapter End of Life Protection

81.1.2 Security Feature

Table 81-2. Security Features

Feature	MCP574xR
Device Life cycle	YES
BAF	YES
Censorship	YES
PASS Module	YES
TDM Module	YES
Flash Sealing	YES

81.2 Secure System Configuration

This section describes the secure system configurations.

81.2.1 Overview

The whole Security Architecture is based on a set of configuration data, which are permanently stored during both FSL Production and Customer Production/Development stages. The MCP574xR devices implements a set of security concepts aimed to guarantee:

- that the programmed data can not be modified (re-programmed)
- that the value of the data being read can not be altered
- that some secret data can not be read by the device CPUs

81.2.2 Security-related configuration

The security architecture and functionality is based on a set of information stored in the protected area of the Flash memory. Below is a list of security related content of the UTEST memory map:

- Device Life Cycle (UTest fixed locations)
- Passwords (UTest fixed locations)

- Censorship (DCF records)
- Block assignment versus TDM erase counter (DCF records)
- Sealing (UTest fixed locations)
- OTP assignment (UTest fixed locations)
- Block assignment versus PASS module password groups (DCF records)
- Security watchdog configuration (SW-DCF records)

Following sections explain how these data are protected versus reading and re-programming.

81.2.3 Device Configuration Format (DCF) records

Most device configuration data for MCP574xR microcontrollers is stored in a format referred to a DCF (Device Configuration Format) record. DCF is part of a mechanism developed to handle settings of device parameters via data stored in OTP Flash memory. Starting at a location in UTest flash memory block, the user stores a series of DCF records during configuration/test activities. The SSCM processes these records during the system reset sequence before the CPU leaves reset. See the DCF Records chapter for a detailed discussion of DCF records.

81.2.4 Security-related data protection

This section describes security-related data protection.

81.2.4.1 Permanent alteration (re-programming)

In order to avoid a malicious permanent alteration of the programmed security configuration data, the UTest and BAF Flash blocks are protected by a set of layers, as described into chapter Flash Memory Protection.

In addition, some of the DCF related to the security are qualified as "write-once", which means that even in case the same be programmed a second time, only the very first programmed value is considered to be valid. This prevents the possibility to alter some permanent DCF configuration.

81.2.4.2 Data alteration upon reading

In order to avoid a malicious alteration of value read from the programmed security configuration, the following protections are used:

- Data is read during the reset phase, when no CPU is enabled, and transferred directly to the intended system clients. This prevents any possibility to alter the data value through SW intervention
- The reading is done under the supervision of voltage, clock, and temperature monitors, which will trigger reset in case the operating conditions are manipulated. This prevents the possibility to corrupt the data being read, by altering the external operating conditions
- ECC engine is applied over the data being read and a reset will be triggered in case of 2 bit failure. This acts as an additional guardian, to prevent the system clients to use altered data
- In case the same DCF is found twice the DCF client may run some check on the read data, in order to allow or not a change of the data to be stored into the client. This acts as a further check to verify that the data being read is consistent.

81.2.4.3 Secret data read protection

In addition to the read and write protection described in the previous two sections, some of the security configuration data shall also be protected versus any type of reading done after the reset phase. Following data shall be protected:

- PASS Module passwords
- JTAG password
- Test Mode Disable Override Passcode

The MCP574xR implements two mechanism to guarantee the protection of those data vs. reading:

- the data is read by a dedicated peripheral under protected conditions, particularly

  - the passwords are read run-time by the peripheral in charge of running the password compare check, with a direct access to the Flash memory, i.e., without using the system data bus
- out the reset phase and depending on the Life Cycle value - see Table 1, the location where the data is stored is blocked versus any reading request coming from the system masters,

Furthermore, two additional mechanisms are available, aimed to protect the access to parts of the Flash memory in case any block of the Flash memory contains data intended to be secret. The mechanisms are:

- the Sealing - see End of Life Protection
- the PASS region read protection - see Block Read Protection

81.2.4.4 PASS Module passwords

The PASS module passwords are programmed by the customer in UTest Flash block locations, which are protected by the mechanism shown in Table 81-2.

81.2.4.5 JTAG password

The JTAG passwords is programmed by the customer in UTest Flash block locations, which are protected by the mechanism shown in Table 81-2. See Debug access protection.

81.3 External access protection

This section describes external access protection.

81.3.1 Overview

The MCP574xR includes several ports to access the device from the external, in order to guarantee the test and development. This chapter describes the concepts implemented in order to protect the external port access.

81.3.2 Debug access protection

This section describes the debug access protection.

81.3.2.1 JTAG TAP controller architecture

The device CPUs embeds a Nexus protocol engine, which makes it possible to control the CPU itself, to trace the read and write bus transactions, and to directly read and write the system memories. The Nexus protocol features are activated by the JTAG TAP controller hardware state machine, which evolves based on the signals forced at the JTAG interface. The JTAG interface is unique. On the other side, the system embeds multiple TAP controller state machines, each one dedicated to a specific resource (e.g. CPU). It is possible, via specific JTAG commands, to route the JTAG commands to each different TAP controllers in order to control the various system resources. The next sections show how the route to the various TAP controller is protected.

81.3.2.2 JTAG TAP controller protection scheme

The mechanism implemented to block the access to a TAP controller is keeping it under reset. As shown in Figure 81-1, when debug interfaces are enabled the JTAGC module is active, allowing access to the IDCODE register, censorship password register, and JDC registers. All other Auxiliary TAP/Nexus clients are held in reset.From security point of view, the system TAP controllers are grouped into:

- JTAG access to all the system TAP controllers (hereafter called Main TAP controller for sake of simplicity)

The external tool connection to the Main TAP controller is protected by a set of conditions:

- Censorship
- JTAG Password
- PASS Debug Lock

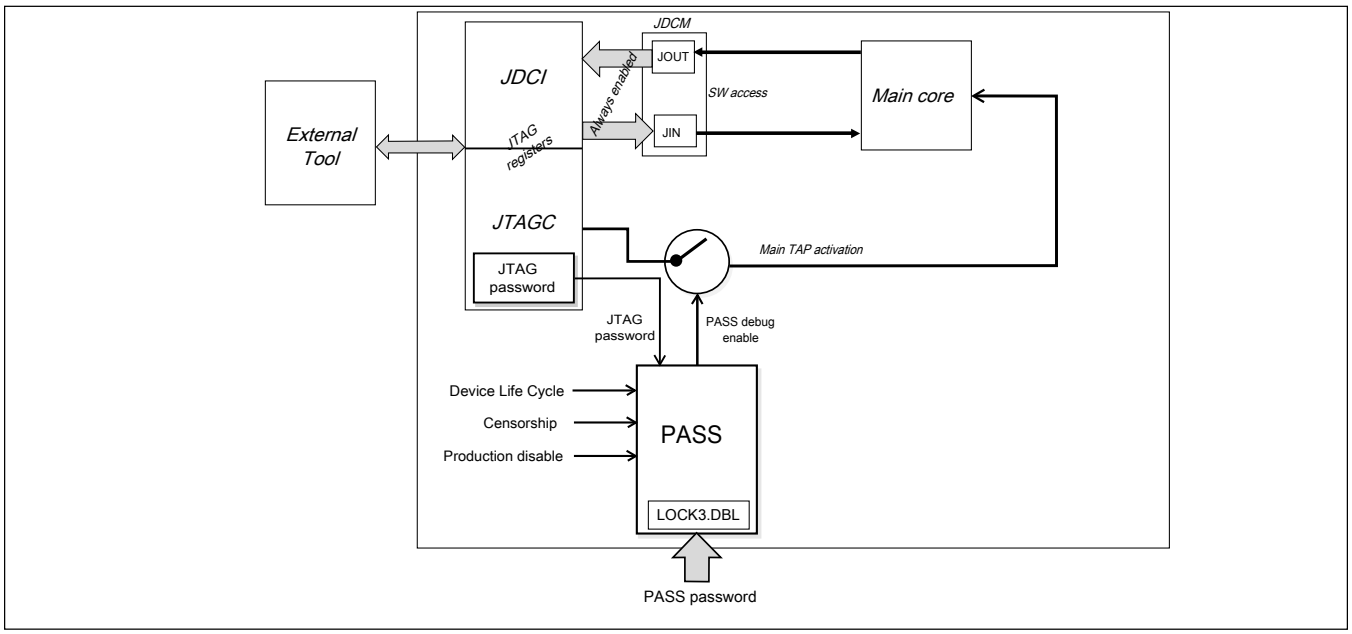


Figure 81-1. JTAG protection architecture

81.3.2.3 Debug protection truth tables

shows the truth tables that regulates the MainTAP Controller debug activation. The PASS module drives an output signal, which is used to enable the Debug TAP Controllers. The PASS Debug Enable output signal depends on the following inputs:

- Censorship
- Debug lock bit
- JTAG password

The censorship works in conjunction with the PASS DBL lock bit. The censorship information is a DCF record loaded during boot and has priority on temporary unprotection. In this way, the debugger can be configured as locked/unlocked multiple times as there are no restrictions on setting and clearing the censorship status in corresponding DCF, apart from the available space for DCF records. The DBL bit protection is used to lock or unlock the debug access once the censorship is activated. The DBL bit can be changed via software, but the writing of the bit can be conditioned by the PASS module password matching (see the PASS chapter for more information). The debug interface is disabled according to the censorship control word settings. The main debug interface can be temporarily re-enabled by the Main cores, following dedicated procedures or exchanging correct passwords.

The JTAG password is used to bypass the DBL bit status. The value sent via JTAG is compared with the password stored inside the Flash.

See Table 82-1.

81.3.2.4 External Tool Authentication protocol

An external tool may be required to authenticate itself with a Challenge/Response protocol before the Debug interface is enabled. This protocol can be implemented in software by any of the MCU CPU.

The Challenge/Response protocol requires that a communication channel between the external tool, requesting the access, and the internal device CPUs exist, independently from the debug interface enable status. For this purpose, a Debug/Host communication interface is implemented to allow Main CPUs to communicate with the external tool. Two 32-bit registers are provided in the Debug/Host communication interface, called JDC (JTAG Data Communication). These registers are always active and allow the requested communication between CPUs and external tool:

- JIN for Debug2HOST communication: writable by the external tool via JTAG and readable by the Main CPUs via software on peripheral bus (AIPS bus)
- JOUT for HOST2Debug communication: writable by Main CPUs via software on peripheral bus (AIPS bus) and readable by the external tool via JTAG

These two registers plus the associated control and status registers are implemented in a module called JDC. JIN register supports an interrupt generation when the debugger has written a data versus the main CPUs. JOUT register support a ready bit, set when the register is written via IPS bus and cleared when debugger reads the data.

An internal CPU can use these communication registers to implement a software authorization protocol, which can be simply a password comparison or a more complex challenge/response protocol.

81.3.3 JTAGM and Debug Over CAN

The MCP574xR embeds the JTAGM module, which makes it possible to inject JTAG commands via software or CAN messages. The JTAG control via CAN is automatically protected by the same mechanism described in chapter Debug access protection. As a results of this, in order to be able to control the TAP controller via CAN, the TAP controller shall be enabled.

81.3.4 Serial bootloader protection

The BAF code include a specific routine, called BAF bootloader (see the BAF chapter for more information), meant to upload data into the system RAM. This feature is intended to let the customer upload into the RAM some code, in order to be able to program the Flash content during their production and development phase. Usually, the BAF Bootloader is executed in case no valid application entry point is found (see the BAF chapter for more information). In order to meet the security requirements, the BAF Bootloader execution is conditioned by the Life Cycle: in case no valid application entry point is found, the BAF code reads the Life Cycle from a read-only register of the SSCM and triggers a destructive reset if the Life Cycle is In Field or OEM_PROD.

81.3.5 BAR execution protection

The BAR (Boot Assist ROM) is ROM-based code, mapped within the AIPS space, which is used during FSL production phase. The BAR code makes it possible to upload data into the system RAM by reading the data directly from a GPIO port. During FSL Production test, the boot CPU1 processor starts executing from the BAR code. After FSL Production test, the CPU1 always starts executing the BAF code. Nevertheless, after FSL Production test, it is still possible to accidentally or maliciously, jump to the BAR code. It is advisable to prevent execution of the BAR code.

The BAR ROM code hardware is designed to always generate an Illegal Peripheral Access Error when it is accessed. This is the same error that is caused by accessing a 'reserved' location in any peripheral. But by default this error is masked by a configuration bit in the SSCM. The Register bit is SSCM_ERROR[RAE]. If this bit is set to '1', any access to BAR code ROM, or any other peripheral reserved location, causes a Prefetch or Data Abort exception to the CPU. The SSCM_ERROR register can also be locked, using REG_PROT from any changes till the next reset.

It is recommended for safe and secure operation, that SSCM_ERROR[RAE] bit is set and the register is locked.

81.4 Flash Memory Protection

This section describes the flash memory protection.

81.4.1 Overview

The Flash memory protection consists of four main features:

- Protection to prevent Flash memory from being erased or programmed by unauthorized users; either malicious or accidental
- Protection to prevent Flash memory from being read by unauthorized users; mainly malicious
- Protection to ensure an authorized user has unimpeded access to programme and erase Flash memory
- Protection to prevent test features of the device from being used to maliciously change the contents of the Flash memory
- Protection for selected flash block from ever being erased; or to log any erase event

These features are described in the sections below.

81.4.2 Flash memory program and erase protection

This section describes flash memory program and erase protection.

81.4.2.1 Password Protection

The Flash memory is protected from Programme and Erase operations by 4 × 256 bit passwords. These passwords are programmed into fixed locations in UTEST flash while Life Cycle = Customer_Delivery. Once Life Cycle is advanced beyond Customer_Delivery, all read accesses to these passwords are blocked.

All the Flash memory blocks can optionally be assigned to one or more of the 4 passwords, by writing DCF Records to UTEST memory.

The passwords and assignment of flash blocks to the passwords is implemented in the PASS module. For full details, refer to the PASS module chapter.

81.4.2.2 One Time Programmable (OTP) Protection

The flash blocks can also be optionally assign as 'OTP'. A flash block assigned as OTP has two attributes

- the block can never be erased.
- a page (64 bits) in the block can only be programmed if it is already erased (0xFFFF_FFFF_FFFF_FFFF). Over programming an page already programmed is not possible

There are no backdoors or passwords to bypass these attributes in User mode; but Flash Test Mode can still erase the flash block - unless the flash block is 'sealed' For more details of flash sealing, refer to Flash Block Sealing.

Flash blocks are assigned as OTP by writing DCF Records to UTEST flash. For mode details of OTP assignment, refer to the Tamper Detect Module (TDM) chapter.

81.4.2.3 Erase Counter Protection

The flash blocks can also optionally be assigned to a mechanism that allows the logging of the number of times a flash block is erased.

This erase counter mechanism uses a log file to record each erase operation. The log file is called a 'Diary' and is maintained in flash memory that has been assigned as OTP. The user is responsible for writing to the diary. The hardware prevents erase operation until the diary has been written.

There are no backdoors or passwords to temporarily bypass the TDM diaries in User mode; but Flash Test Mode can still erase the flash block - unless the flash block is 'sealed'. For more details of flash sealing, refer to Flash Block Sealing.

The flash blocks can be assigned to a Diary using DCF Records written to UTEST flash. There are 6 completely independent Diaries. For mode detail, refer to Tamper Detect Module (TDM) chapter.

81.4.2.4 Flash Programme and Erase Implementation

The Flash memory contains a group of registers called LOCK0, LOCK1, LOCK2 & LOCK3. These registers contain a bit for each individual flash block. To programme or erase a flash block, the relevant bit in one of these registers must first be cleared. Each bit in these LOCKx registers is further qualified by 3 signals per bit from outside the flash. These signals are:

- plock - when asserted, prevents a block being programmed
- elock - when asserted, prevents a block being erased
- opplock - when asserted, prevents a block being over-programmed

These signals come from the PASS module and the TDM module as shown below. The logic is repeated for every bit in all four LOCKx registers

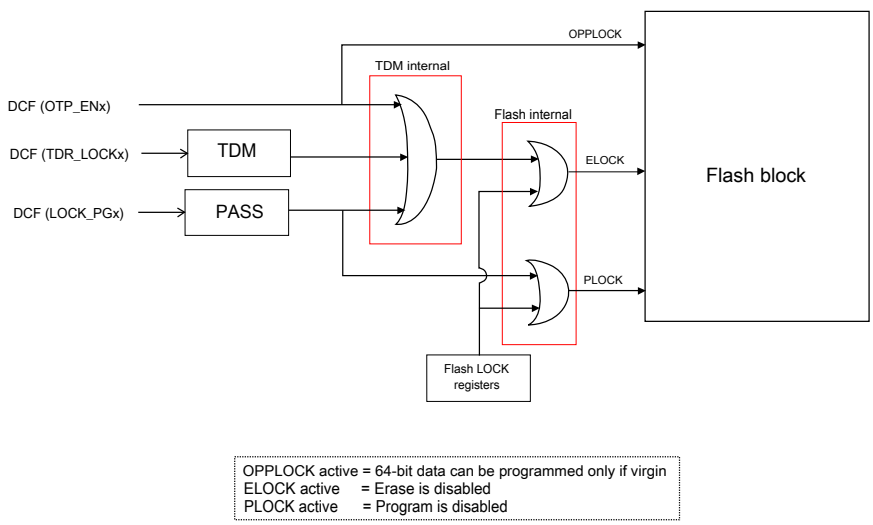


Figure 81-2. Flash Prog/Erase Protection Glue Logic Diagram

81.4.3 Block Read Protection

This section describes block read protection.

81.4.3.1 General Read Protection

The Flash blocks can optionally be configured to have all read accesses blocked in certain conditions.

There are two conditions that cause this flash block read protection:

- Lifecycle = OEM_Production or In_Field; Device is Censored; Debug Interface is Enabled.
- Life Cycle = FA; Device is censored.

The hardware monitors the enablement of the debug interface, and if Debug is enabled AND the device is Censored, the pflash controller is instructed to block read (and write) accesses from all bus masters, as configured by the PASS_LOCK3[RLx] bits.

See Table 82-2.

The purpose of this is to prevent unauthorized reading of the flash contents. The side affect of this is that if a debugger is 'Hot Attached' while the application is running, and debug is enabled, the application will probably crash because the CPUs can no longer fetch code from flash.

To prevent this crash from occurring, there are two options:

- Enable debug access while the device is held in reset. Then use the debugger to write to the PASS module LOCK3[RLx] bits to temporarily remove the read protection. Then release reset to allow the application to run
- Use one of the running CPUs to write to the PASS module LOCK3[RLx] to temporarily remove read protection before enabling debug access

The complete flash array address range is divided into the following read protection regions:

1. UTEST Flash
2. Main code flash (most of the flash)
3. Main EEPROM data

These regions can be configured independently through the PASS_LOCKn_PGm register of the PASS module.

81.4.4 Test Feature Protection

Flash Test mode is a mode of the flash memory that allows direct access to the internal operation of the flash memory read and write mechanism. Therefore flash test mode has the capability read and to disturb the data stored in the flash. As most of the security

features are either configured from flash memory or software stored in flash memory, the Flash Test mode is considered a security vulnerability. Following independent features are used to protect from Flash Test Mode attacks

- Flash block sealing is a feature that allows any flash block to be marked as 'sealed from flash test mode'. Flash sealing is described in the next section.

81.4.4.1 Flash Block Sealing

Sealing a block has no affect on normal user operation but it means that in Flash Test Mode, no test features can be used on the sealed block. Sealing is very good for preventing flash content from being read or changed in Flash Test mode, but it has the side-effect that no FA can be performed either. So care should be taken in selecting blocks to seal as this reduces the FA capability of the device.

81.4.4.1.1 Assigning Flash blocks as sealed

Within UTEST Flash are two regions called 'Test Mode Disable Block Select Group A' and 'Test Mode Disable Block Select Group B'. Each of these two regions contains 4 × 32 bit registers that contain a bit for each flash block. Flash blocks are assigned as seal by writing the relevant bit in either Group A or Group B. Note: UTEST is OTP. The smallest data size that can be written to UTEST is 8 bytes - 2 × 32bit registers. Therefore it is only possible to assign blocks in the same register pair as sealed, twice. Once for Group A and once for Group B.

81.4.4.1.2 Test Mode Override Password

The Flash block sealing mechanism also includes a password protected override called 'Test Mode Override Passcode.

This is a 32 bit password that the customer programmes into UTEST flash during module production.

If a device is returned to FSL for FA, this password can be used to unlock the sealing mechanism and allow full flash test capability on sealed blocks.

The Test Mode Override Passcode has 3 illegal values: 0x0000_0000, 0xFFFF_FFFF, 0x5555_5555. If the customer programmes the Test Mode Override Passcode in UTEST to any of these values, no sealing override is possible. Flash blocks marked as 'sealed' are permanently sealed with no override possible.

81.4.4.1.3 Sealing Truth Tables

The table below show the effect of sealing blocks.

Table 81-4. Flash Block Sealing

Block	Test Mode Disable Block Select	Flash Block Access through		
		Flash Test Mode Register Interface	Main/Alternate Register Interface	Read Bus
Test Flash	Don't care	Read and Write	No access	No access
UTest	Don't care	No access	Program operation is allowed only on nonprogrammed words. Erase is blocked. User test operations (e.g. Margin Read) are blocked	Read is allowed
BAF	Not sealed	Read and Write	Program operation is allowed only on nonprogrammed words. Erase is blocked. Program and user test operations are allowed	Read is allowed
	Sealed	No access		
Others	Not sealed	Read and Write	Program, erase, and user test operations are allowed	Read is allowed
	Sealed	No access		

81.4.5 Flash overlay mechanism restrictions

Load and Instruction-fetch accesses from any bus master to flash, can be configurably diverted to overlay RAM or system RAM.

The flash is memory mapped twice: one memory map allows overlay remap; the other doesn't.

No overlay memory region: 0x00000000 - 0x07FFFFFF

Mirrored region, that allows overlay remap: 0x08000000 - 0x0BFFFFFF

Remappable flash blocks are mapped in both memory map regions at the same address, with only different offset: 0x0800000 offset has to be applied to access flash blocks in mirrored overlay region.

OTP blocks should be used in the no-remap region.

If code is to be executed, it should be in the no-remap region.

Caches and RAMs can not be remapped anyway because the access is not through the flash controller.

81.5 End of Life Protection

This section describes End of Life protection.

81.5.1 Overview

After the module has been in the field for some time, it is possible that a defect may occur and the module is returned to the customer for further debugging and analysis. The MCP574xR includes several mechanisms that make this possible, but at the condition (under customer's choice) that the device will not be working any more in the customer application. The following sections describes each of these mechanisms.

81.5.2 BAF code in Failure Analysis

The BAF code checks at the very beginning the Life Cycle value and, in case it is acknowledged as Failure Analysis, terminates the BAF execution in an endless loop.

81.5.3 Protection of Flash Test Mode activation

Flash Test mode provides access to the hardware that controls the reading and writing of the individual flash bit cells and includes several modes for measuring and stressing the flash bit cells. For highly secure applications, this makes Flash Test Mode a security vulnerability and merits special features to protect it.

Flash Test mode is required during initial manufacture and testing of the device (Lifecycle < Customer_Delivery) and also during Failure Analysis of the device (Lifecycle = Failure_Analysis).

When Lifecycle = FSL_Production, the test mode is always available.

Chapter 82
Password and Device Security Module (PASS)

82.1 Password and Device Security Module (PASS) chip-specific module information

The PASS module is used to implement password-based read and write protection for flash blocks. Up to four levels of password protection can be implemented for each flash block.

NOTE

The registers discussed in this section are initialized via DCF records, so the reset values depend on user settings.

Each password group defined in the DCF records has a set of four LOCK*n* registers association with it—PASS_LOCK*n*_PG*n*. The bits of these registers are associated with specific flash blocks. The mapping is shown in the attached Flash Block Assignment spreadsheet.

NOTE

Password Challenge passwords and JTAG passwords that are all 0's or all 1's are invalid.

The following table shows the the Debug Interface Access scenarios on this chip.

Table 82-1. Debug Interface Enable Truth Table

Input Conditions					Outcome
Lifecycle	**Production Disable mode**	**DCF Censorship**	**Debug Enable (LOCK3[DBL] bit)**	**JTAG Password**	**Debug Interface Enable**
Customer Delivery	Don't Care	Don't Care	Don't Care	Don't Care	Enabled
OEM Production	Don't Care	Not censored	Don't Care	Don't Care	Enabled
		Censored	Unlocked (DBL=0)	Don't Care	Enabled
			Locked (DBL=1)	Matched	Enabled
				Not Matched	Blocked

Table continues on the next page...

Table 82-1. Debug Interface Enable Truth Table (continued)

Input Conditions					Outcome
Lifecycle	**Production Disable mode**	**DCF Censorship**	**Debug Enable (LOCK3[DBL] bit)**	**JTAG Password**	**Debug Interface Enable**
In Field	Off	Not censored	Don't Care	Don't Care	Enabled
		Censored	Unlocked (DBL=0)	Don't Care	Enabled
			Locked (DBL=1)	Matched	Enabled
				Not Matched	Blocked
	On	Don't Care	Don't Care	Don't Care	Blocked
Failure Analysis	Don't Care	Not censored	Don't Care	Don't Care	Enabled
		Censored	Unlocked (DBL=0)	Don't Care	Enabled
			Locked (DBL=1)	Matched	Enabled
				Not Matched	Blocked

The following table summarizes the scenarios of read lock (RL) region protection on this chip.

Table 82-2. Flash memory read protection truth table

Inputs				Outputs	
Life Cycle	**Censorship DCF client**	**Debug Interface**	**LOCK3[RLx]**	**Read Protected RL[0] (UTest)**	**Read Protected RL[3:1] (other regions)**
Customer Delivery	Don't care	Don't care	Don't care	Readable	Readable
OEM Production or In Field	0x55AA	Don't care	Don't care	Readable	Readable
	Any other value	Blocked	Don't care	Readable	Readable
		Enabled	Locked	Read Blocked	Read Blocked
			Unlocked	Readable	Readable
Failure Analysis	0x55AA	Don't care	Don't care	Readable	Readable
	Any other value	Don't care	Locked	Readable	Read Blocked
			Unlocked	Readable	Readable

82.2 Introduction

82.2.1 Overview

The Password and device security (PASS) module shown in the following figure, receives password challenges and determines their validity. It also maintains the device security and access states.

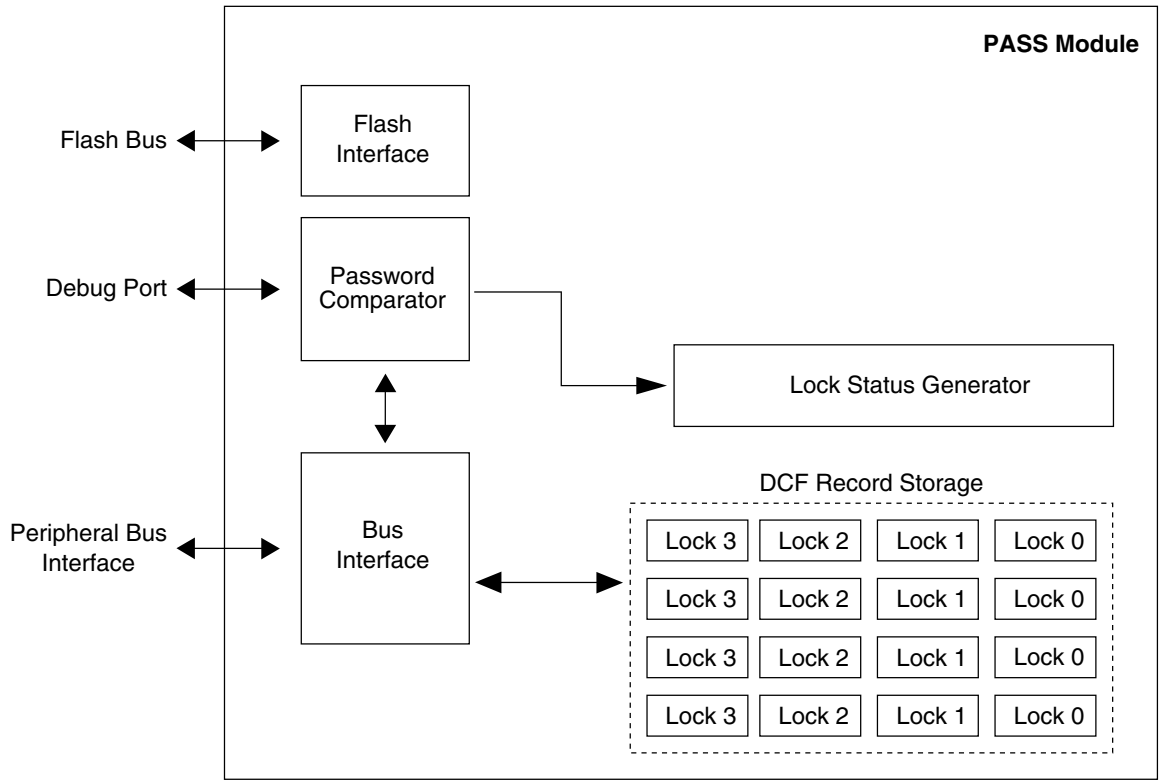


Figure 82-1. PASS module block diagram

82.2.2 Features

The PASS includes these distinctive features:

- Life Cycle status
- Password comparison
- JTAG password comparison
- Production Disable control
- Flash read, write, and erase control

82.2.3 Modes of operation

The PASS operates identically in all system modes.

82.3 External signal description

The PASS has 1 external pin that is used for Production Disable control.

82.4 Memory map and register definition

This section provides a detailed description of all memory-mapped registers and DCF clients in the PASS.

Where a DCF client is listed, the reset value of that register may automatically be pre-loaded during reset based on the data stored in DCF Records in flash UTEST region.

The following table shows the memory map for the PASS. Note that all addresses are offsets; the absolute address for registers may be calculated by adding the specified offset to the base address of the PASS. The address of the DCF client is local to the PASS module and must be qualified with the appropriate Client Select (CS[14:0]) for the PASS module, as defined in the DCF Record Section of the Reference Manual.

NOTE

Non-implemented registers generate bus aborts if enabled.

The following registers are available in the PASS. The shaded bits are reserved for future use. For future compatibility, only write ‘0’ to these bits; they always read ‘0’.

PASS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Life Cycle Status Register (PASS_LCSTAT)	32	R	See section	82.4.1/3494
8	Challenge Selector Register (PASS_CHSEL)	32	R/W	0000_0000h	82.4.2/3495
10	Challenge Status Register (PASS_CSTAT)	32	R	0000_0000h	82.4.3/3495
20	Challenge Input Register (PASS_CIN0)	32	W (always reads 0)	0000_0000h	82.4.4/3496
24	Challenge Input Register (PASS_CIN1)	32	W (always reads 0)	0000_0000h	82.4.4/3496
28	Challenge Input Register (PASS_CIN2)	32	W (always reads 0)	0000_0000h	82.4.4/3496
2C	Challenge Input Register (PASS_CIN3)	32	W (always reads 0)	0000_0000h	82.4.4/3496
30	Challenge Input Register (PASS_CIN4)	32	W (always reads 0)	0000_0000h	82.4.4/3496

Table continues on the next page...

PASS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
34	Challenge Input Register (PASS_CIN5)	32	W (always reads 0)	0000_0000h	82.4.4/3496
38	Challenge Input Register (PASS_CIN6)	32	W (always reads 0)	0000_0000h	82.4.4/3496
3C	Challenge Input Register (PASS_CIN7)	32	W (always reads 0)	0000_0000h	82.4.4/3496
100	Password Group n - Lock 0 Status Register (PASS_LOCK0_PG0)	32	R/W	See section	82.4.5/3497
104	Password Group n - Lock 1 Status Register (PASS_LOCK1_PG0)	32	R/W	See section	82.4.6/3498
108	Password Group n - Lock 2 Status Register (PASS_LOCK2_PG0)	32	R/W	See section	82.4.7/3499
10C	Password Group n - Lock 3 Status Register (PASS_LOCK3_PG0)	32	R/W	See section	82.4.8/3501
110	Password Group n - Lock 0 Status Register (PASS_LOCK0_PG1)	32	R/W	See section	82.4.5/3497
114	Password Group n - Lock 1 Status Register (PASS_LOCK1_PG1)	32	R/W	See section	82.4.6/3498
118	Password Group n - Lock 2 Status Register (PASS_LOCK2_PG1)	32	R/W	See section	82.4.7/3499
11C	Password Group n - Lock 3 Status Register (PASS_LOCK3_PG1)	32	R/W	See section	82.4.8/3501
120	Password Group n - Lock 0 Status Register (PASS_LOCK0_PG2)	32	R/W	See section	82.4.5/3497
124	Password Group n - Lock 1 Status Register (PASS_LOCK1_PG2)	32	R/W	See section	82.4.6/3498
128	Password Group n - Lock 2 Status Register (PASS_LOCK2_PG2)	32	R/W	See section	82.4.7/3499
12C	Password Group n - Lock 3 Status Register (PASS_LOCK3_PG2)	32	R/W	See section	82.4.8/3501
130	Password Group n - Lock 0 Status Register (PASS_LOCK0_PG3)	32	R/W	See section	82.4.5/3497
134	Password Group n - Lock 1 Status Register (PASS_LOCK1_PG3)	32	R/W	See section	82.4.6/3498
138	Password Group n - Lock 2 Status Register (PASS_LOCK2_PG3)	32	R/W	See section	82.4.7/3499
13C	Password Group n - Lock 3 Status Register (PASS_LOCK3_PG3)	32	R/W	See section	82.4.8/3501

82.4.1 Life Cycle Status Register (PASS_LCSTAT)

The Life Cycle Status Register reflects the production status and DCF Censorship client status of the device. It is possible to mature the device (move the device life cycle forward), but never to revert the life cycle to an earlier state.

The reset value of the LCSTAT register LIFE field derives from an external life cycle management module (like SSCM). It is a 3-bit life cycle. This register simply displays the life cycle input from external blocks.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CNS	JUN	1	0												
W																
Reset	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0													LIFE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

* Notes:
- CNS field: Reset value depends on the Censorship DCF client

PASS_LCSTAT field descriptions

Field	Description
0 CNS	This field indicates whether the DCF Censorship client has been written with a DCF record to 0x55AA or if the DCF Censorship client contains any other value than 0x55AA. The bit can be set (or cleared) via DCF Record in UTest Flash. **NOTE:** If no DCF record has been written to affect the value of the CNS bit, its reset value will be 1. 1 DCF Censorship Client value is anything other than 0x55AA. 0 DCF Censorship client value is 0x55AA.
1 JUN	This field indicates whether the device has been temporarily uncensored. Successful transmission of the JTAG password sets this bit to '1', otherwise it will be 0. (In life cycle FA, the JUN bit will always be 0.) Unsuccessful transmission clears this bit.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
3–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 LIFE	The life cycle setting is manipulated externally (SSCM). Pass displays a 3-bit input life cycle 000 Failure Analysis 001 Reserved

Table continues on the next page...

PASS_LCSTAT field descriptions (continued)

Field	Description
	010 OEM Production 011 Customer Delivery 100 Reserved 101 Reserved 110 Freescale Production 111 In Field

82.4.2 Challenge Selector Register (PASS_CHSEL)

This register determines which password group is challenged via the Challenge Input Register.

When the Master Only (MO) bit of the PASS_LOCK3_PGn register is set, only the master that unlocked access to the passgroup settings will be able to access these registers. The CMST field of the Challenge Status Register will reflect the locking master. When the Master Only (MO) bit of the PASS_LOCK3_PGn register is not set, the passgroup settings can be accessed by any master.

When a master writes this register, its ID sets the CMST field of the Challenge Status register.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																														GRP	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PASS_CHSEL field descriptions

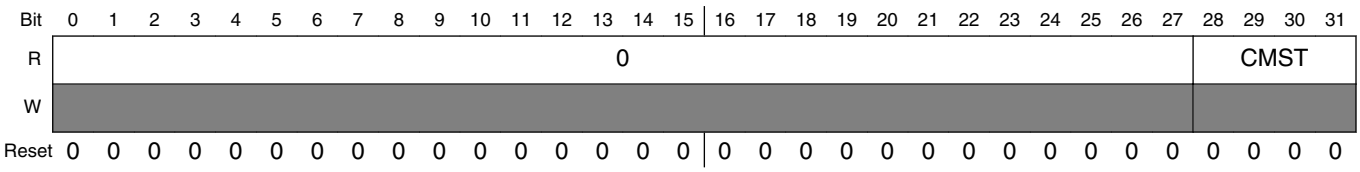
Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 GRP	Password Group to unlock 00 Password Group 0 01 Password Group 1 10 Password Group 2 11 Password Group 3

82.4.3 Challenge Status Register (PASS_CSTAT)

This register provides status information for the password challenge.

Only the master shown in this register can access the CINn registers - access by other masters result in a comparison fail regardless of the values written.

Address: 0h base + 10h offset = 10h



PASS_CSTAT field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 CMST	ID of the master which has last written the CHSEL register

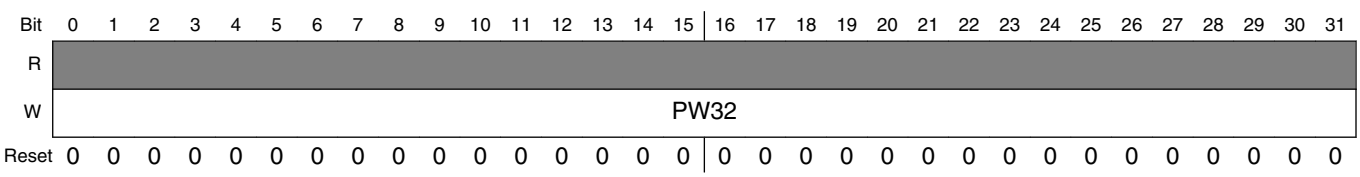
82.4.4 Challenge Input Register (PASS_CIN*n*)

This register is used to provide the Challenge word *n* for a password group as selected by the Challenge Selector Register.

After programming the Challenge Selector Register, the password challenge needs to be written with eight 32-bit writes from CIN0 to CIN7 in that order. The most significant 32 bits of the password challenge must be written to CIN0. Once CIN7 is written, the password is compared. The write access only completes after the comparison has been executed. In case of a successful password comparison, the PASS module stores the ID of the master which has written CIN0 to CIN7 in the PASS_LOCK3_PGn register of the selected password group, as well as clears the PGL bit of that register.

All accesses to these registers need to be made by the bus master that wrote the CHSEL register previously. If another master writes any of the CIN registers before the original master has written CIN7, the comparison fails.

Address: 0h base + 20h offset + (4d × i), where i=0d to 7d



PASS_CIN*n* field descriptions

Field	Description
0–31 PW32	32 bits of the 256-bit password challenge. Only 32-bit writes may be used.

82.4.5 Password Group *n* - Lock 0 Status Register (PASS_LOCK0_PG*n*)

Each Password Group has one Lock 0 Status register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked the register becomes writable. For additional details of the LOCK bits, refer to the Flash memory description.

The LOCK bits do not come into effect before the life cycle has matured to *OEM Production* or older.

LOCK0_PG*n* register functions are as shown in this section.

NOTE

Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during functional reset. The default value before any DCF record is written is '1', indicating that a block is locked.

During the Customer Delivery life cycle, the user must program a valid password or a DCF record that writes to this register to unprotect the UTEST for each password group in order to prevent permanent and irreversible locking of Flash blocks.

Warning

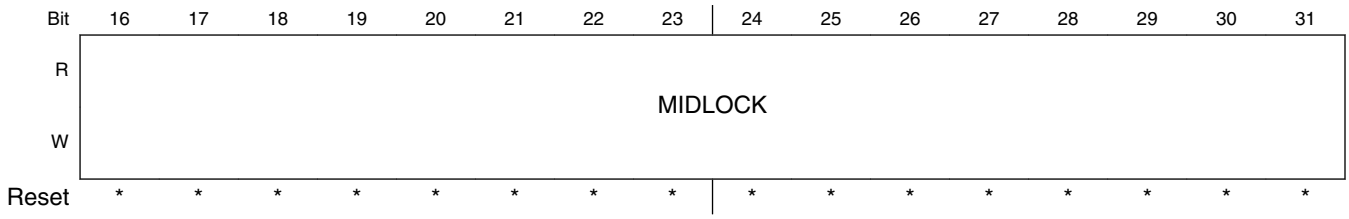
Failure to program valid passwords or DCF records for all password groups before the life cycle matures to OEM Production renders the part unusable.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the life cycle is OEM Production or older, the Flash blocks are always unlocked, regardless of the values in the password groups.

Address: 0h base + 100h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TSLOCK	ATSL	LOWLOCK													
W																
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*



* Notes:
- MIDLOCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit will reset to ‘1’.
- LOWLOCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit will reset to ‘1’.
- ATSL field: If no DCF records have been written to change the ATSL bit state, the bit will reset to ‘1’.
- TSLOCK field: If no DCF records have been written to change the TS bit state, the bit will reset to ‘1’.

PASS_LOCK0_PG*n* field descriptions

Field	Description
0 TSLOCK	UTest NVM Lock. This bit is used to lock the UTest NVM block from programs (erase protection not needed since UTest NVM is OTP and not erasable).
1 ATSL	Alternate Interface UTest NVM Lock. This bit is used to lock the UTest NVM block from programs on the alternate interface (erase protection not needed since UTest NVM is OTP and not erasable).
2–15 LOWLOCK	Low Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.
16–31 MIDLOCK	Mid Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

82.4.6 Password Group *n* - Lock 1 Status Register (PASS_LOCK1_PG*n*)

Each Password Group has one Lock 1 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable . For additional details of the LOCK bits, refer to the Flash description.

The LOCK bits do not come into effect before the life cycle matures to OEM Production or older.

LOCK1_PG*n* register functions are as shown in this section.

NOTE

Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during functional reset . The default value before any DCF record is written is '1', which indicates a block is locked. If no DCF records have been written to change the lock state, the bits will reset to '1'.

During the Customer Delivery life cycle, the user must program a valid password or a DCF record that writes this register to unprotect the UTEST for each password group to prevent permanent and irreversible locking of Flash blocks.

CAUTION

Failure to program valid passwords or DCF records for all password groups before the life cycle matures to OEM Production renders the part unusable. However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the life cycle is older than OEM Production, the Flash blocks are always unlocked, regardless of the values in the password groups.

Address: 0h base + 104h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HIGHLOCK															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:

- HIGHLOCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit will reset to ‘1’.

PASS_LOCK1_PG*n* field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHLOCK	High Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

82.4.7 Password Group *n* - Lock 2 Status Register (PASS_LOCK2_PG*n*)

Each Password Group has one Lock 2 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable . For additional details of the LOCK bits, refer to the Flash description.

The LOCK bits do not come into effect before the life cycle is matured to OEM Production or older.

NOTE

Lock registers can only be accessed with 32-bit operations.

LOCK2_PG*n* register functions are as shown in this section.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during functional reset. The default value before any DCF record is written is '1', which indicates a block is locked.

During the Customer Delivery life cycle, the user must program a valid password or a DCF record that writes this register to unprotect the UTEST for each password group in order to prevent permanent and irreversible locking of Flash blocks.

Warning

Failure to program valid passwords or DCF records for all password groups before the life cycle matures to OEM Production renders the part unusable. However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the life cycle is older than OEM Production, the Flash blocks are always unlocked, regardless of the values in the password groups.

Address: 0h base + 108h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	L_256LCK																															
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:

- L_256LCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit resets to '1'.

PASS_LOCK2_PG*n* field descriptions

Field	Description
0–31 L_256LCK	Lower 256KByte Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

82.4.8 Password Group *n* - Lock 3 Status Register (PASS_LOCK3_PG*n*)

Each Password Group has one Lock 3 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable.

The LOCK bits do not come into effect before the life cycle matures to OEM Production or older.

LOCK3_PGn register functions are as shown in this section.

NOTE

Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during functional reset. The default value before any DCF record is written is '1', which indicates a block is locked.

During the Customer Delivery life cycle, the user must program a valid password or a DCF record that writes this register to unprotect the UTEST for each password group in order to prevent permanent and irreversible locking of Flash blocks.

Warning

Failure to program valid passwords or DCF records for all password groups before the life cycle matures to OEM Production renders the part unusable. However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the life cycle is older than OEM Production, the Flash blocks are always unlocked, regardless of the values in the password groups.

Once a censored device enters the Failure Analysis life cycle, read protection is active regardless whether the system is in debug mode or not. It is still possible to unlock Flash regions with the correct passwords, however. Flash Utest region controlled by LOCK3[0] is read unlocked in Failure Analysis life cycle.

NOTE

Do not toggle DBL and RLx bits with the same write command. Use one instruction to program the DBL bit and a separate instruction to toggle the RLx bits.

Address: 0h base + 10Ch offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PGL	DBL	MO	0	MSTR				0			1	1	RL2	RL1	RL0
W																
Reset	*	*	*	0	1	1	1	1	0	0	0	1	1	*	*	*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	U_256LCK															
W																
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:

- U_256LCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit will reset to '1'.
- RL0 field: If no DCF records have been written to change the RL0 bit lock state, the bit will reset to a logic 1. This Flash block region is always readable if the life cycle is Customer Delivery or earlier. To determine if this region is readable for Life Cycle states later than Customer Delivery, consult the Flash Memory Read Protection Truth Table.
- RL1 field: If no DCF records have been written to change the RL1 bit lock state, the bit will reset to a logic 1. This Flash block region is always readable if the life cycle is Customer Delivery or earlier. To determine if this region is readable for Life Cycle states later than Customer Delivery, consult the Flash Memory Read Protection Truth Table.
- RL2 field: If no DCF records have been written to change the RL2 bit lock state, the bit will reset to a logic 1. This Flash block region is always readable if the life cycle is Customer Delivery or earlier. To determine if this region is readable for Life Cycle states later than Customer Delivery, consult the Flash Memory Read Protection Truth Table.
- MO field: If no DCF records have been written to change the MO bit lock state, the bit will reset to ‘1’.
- DBL field: If no DCF records have been written to change the DBL bit lock state, the bit will reset to ‘1’.
- PGL field: Reset value depends on life cycle. If life cycle is Customer Delivery or earlier, the value will be 0, otherwise 1.

PASS_LOCK3_PG*n* field descriptions

Field	Description
0 PGL	Password Group Lock. This bit may be set in software to lock the password group, when the device is older than Customer Delivery. Before that the value of the bit is always 0. This bit is only updatable from SW path. It cannot be updated from DCF load from Flash. This bit would be automatically locked once Flash Scanning is finished after functional reset. This bit can be cleared by writing a valid password to the Password Challenge Input Registers. 0 Password group registers are unlocked. All four registers in the Password group may be read and written without restriction. 1 Password group registers are locked. Writes to the password group registers have no effect. Read accesses work normally. (If life cycle is Customer Delivery or earlier, this bit has no effect and the registers are always unlocked.)

Table continues on the next page...

PASS_LOCK3_PG*n* field descriptions (continued)

Field	Description
1 DBL	Debug Interface Lock. The default value of this bit would be '1' if not updated from flash. At power on, this bit would be updated from DCF records in flash. This bit cannot be updated from FLASH after first functional reset. After first functional reset, it can only be updated by writing to it once the passgroup has been unlocked via the PASS Password challenge mechanism. 0 Debug interface is unlocked. Accessed to all JTAG clients is allowed 1 Debug interface is locked. Only access to JTAG password challenge register is possible. Access to all other JTAG clients is blocked
2 MO	Master Only. 0 All masters can modify the passgroup settings if PGL is cleared 1 Only the Master which unlocked the passgroup can change the passgroup settings
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 MSTR	Master Access. Only the Master with this matching master ID can change the passgroup settings. This field is loaded with the ID of the master which unlocked the passgroup once the correct password is provided.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
13 RL2	Region 2 is protected. The default value of this bit would be '1' if not updated from flash. At power on, this bit would be updated from records loaded in flash. This bit cannot be updated from FLASH after first functional reset. After first functional reset, It can only be updated by Test Access Port or Through PASS Password challange mechanism. 0 Region 2 is not protected. 1 Region 2 is protected.
14 RL1	Region 1 is protected. The default value of this bit would be '1' if not updated from flash. At power on, this bit would be updated from records loaded in flash. This bit cannot be updated from FLASH after first functional reset. After first functional reset, It can only be updated by Test Access Port or Through PASS Password challange mechanism. 0 Region 1 is not protected. 1 Region 1 is protected.
15 RL0	Region 0 is protected. The default value of this bit would be '1' if not updated from flash. At power on, this bit would be updated from records loaded in flash. This bit cannot be updated from FLASH after first functional reset. After first functional reset, It can only be updated by Test Access Port or Through PASS Password challange mechanism. 0 Region 0 is not protected. 1 Region 0 is protected.

Table continues on the next page...

PASS_LOCK3_PG*n* field descriptions (continued)

Field	Description
16–31 U_256LCK	Upper 256 KB Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

82.5 DCF clients

The following table shows the DCF clients for the PASS.

Where a DCF client is listed, the reset value of that register may automatically be pre-loaded during reset based on the data stored in DCF Records in Flash UTEST region.

The address of the DCF client is local to the PASS module and must be qualified with the appropriate Client Select (CS[14:0]) for the PASS module, as defined in the DCF Record Section of the Reference Manual or in DCF sheet attached to this document.

Table 82-36. PASS DCF clients

Offset	Register	Access	Reset Value	DCF Client Address (ADDR[14:0])	Section
Censorship					
None	Censorship	None	0x0000_0000	0x00B0	Censorship
Production Disable					
None	Production Disable	None	0x0000_0000	0x00C0	Production Disable

82.5.1 Censorship

Censorship is a DCF client that is *not* accessible by software. It is automatically pre-loaded during reset with data stored in DCF Record format in the Flash UTEST region.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1 This bit is not defined

2 This value is loaded by DCF Record during reset

Field	Description
16:31	Censorship If this value is 0x55AA the MCU is NOT censored If this value is anything other than 0x55AA, the MCU is censored

82.5.2 Production Disable

Production Disable is a DCF client that is not accessible by software. It is automatically pre-loaded during reset with data stored in DCF Record format in the Flash UTEST region. The Production disable DCF client defines the Debug Disable aspects of the Production Disable feature.

The DCF client implements 1 bit with a default value of ‘0’. The DCF client can only be written once and writes from subsequent DCF records are ignored.

For more information on Production disable, see Production disable

DCF Client Address: 0x00C0 Production Disable — Access: DCF Record Write Once Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W																DD[1]
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

[1] This value is loaded by DCF Record during reset

Figure 82-34. Production Disable DCF client

Table 82-37. Production Disable DCF Client field descriptions

Field	Description
0-30	This bit is reserved and always has the value 0.
31 DD	Debug disable. '0' = Debug Interface is Enabled and Disabled using Censorship, JTAG password and PASS_LOCK3[DBL] bit (see chip-specific PASS information). '1' = Debug Interface is disabled

82.5.2.1 Debug Disable (DD)

Debug Disable is an optional feature that, once the life cycle is In Field or beyond, overrides all other debug controls and disables the debug interface. The Debug Disable feature has a higher priority than Censorship, JTAG password and the PASS LOCK3[DBL] bit.

82.6 Functional description

The primary purpose of the PASS is to provide fine-grained access control to the Flash and debugger interface. It utilizes the DCF mechanism, as described in the SSCM section.

82.6.1 Censorship

Censorship is the overall control that enables and disables two independent security features: Debug Interface access and flash memory read protection.

The Debug Interface access is controlled by the CNS and LIFE bits of the LCSTAT register according to the Debug Interface Access truth table. The flash memory read protection is also controlled by the CNS and LIFE bits of the LCSTAT register according to the Flash memory read protection truth table.

The CNS bit is set or cleared by executing a DCF record that writes to the Censorship DCF client (See Censorship). The LIFE bits of the LCSTAT register are controlled by the SSCM using user programmed values from specific locations in the UTEST flash.

82.6.2 Debug Interface Access

The debug interface is fully enabled in Customer Delivery life cycle. But from OEM Production life cycle, it can be blocked using censorship and unlocked by either entering a 256 bits password via JTAG or by software clearing the DBL bit the PASS_LOCK3 registers (See Password Group *n* - Lock 3 Status Register (PASS_LOCK3_PG*n*)). Both these debug enabling options are further conditioned by life cycle and Production Disable.

The debug interface enable truth table for this chip is shown in the device specific PASS section of this chapter.

82.6.2.1 JTAG Password Challenge

To enable the Debug Interface Access, the JTAG password challenge register in the JTAGC, needs to be written by an external tool. If that password matches the JTAG password stored in UTEST flash, Debug Interface Access is enabled until the next destructive reset.

82.6.3 Flash Memory Read Protection

Flash Memory Read Protection is a security feature that can selectively make regions of the flash memory unreadable. Any attempt to read a region of flash that is read protected results in the read access terminating with a transfer error. This is consistent for all bus masters including debug bus masters.

The assertion of flash memory region read protection is controlled by Life Cycle, Censorship and whether the Debug Interface access is enabled or blocked. The Flash memory read protection truth table describes the combination of conditions that drive Flash Memory Read Protection.

The intention of the feature is to protect the contents of the flash from being read. The flash memory is readable in normal operation, allowing the application to run, but becomes read protected when a Debug tool is attached and enabled (JTAG password or LOCK3[DBL] bit).

The flash memory is also read protected when the life cycle status is Failure Analysis. This life cycle is used for a module field failure, where the device is returned for failure analysis. This protection prevents the contents of the flash memory from being read.

The flash memory is divided into a number of Regions (3 to 5 depending on the device). The assignment of the regions is defined in the device reference manual.

The memory read protection truth table for this chip is shown in the device specific PASS information.

82.6.4 Production Disable

82.6.4.1 Overview and rationale

Production Disable is an optional feature that, once enabled, allows the MCU to function completely normally without any constraints, except the Debug interface is disabled.

82.6.4.2 Enabling

The feature is enabled via the Production Disable DCF client in the PASS module. Bit 0 enables the Debug Interface Disable. See Production Disable for details.

This information can be written by adding a DCF Record to the DCF record list in Flash UTEST. The DCF Client is configured to accept only one write. Subsequent DCF records that write to this DCF client are ignored.

If no DCF Record writes to this client, the default state of this feature is disabled. If, for a particular application, there is no intention of ever using this feature, it would be good practice to add a DCF Record that writes this client to 0x0000_0000. This is not essential, but ensures the feature can not be enabled accidentally.

82.7 Initialization/application information

82.7.1 Reset

The reset state of each individual bit is shown within the Register Description section. However, many bits receive their values based on DCF records stored in the UTEST Flash memory block.

82.7.2 Setting lock bits in a password group

There is no restriction on setting and clearing lock registers in the UTEST Flash, apart from the available space for DCF records. It is possible to set and clear them multiple times.

Table 82-38. CS/ADDR Values for Lock Registers

Register	DCF Command Word
PW Group 0	
Lock0	0xxxxx[1, 1] _0100
Lock1	0xxxxx[1] _0104
Lock2	0xxxxx[1] _0108
Lock3	0xxxxx[1] _010C
PW Group 1	
Lock0	0xxxxx[1] _0110
Lock1	0xxxxx[1] _0114
Lock2	0xxxxx[1] _0118
Lock3	0xxxxx[1] _011C
PW Group 2	
Lock0	0xxxxx[1] _0120
Lock1	0xxxxx[1] _0124
Lock2	0xxxxx[1] _0128
Lock3	0xxxxx[1] _012C
PW Group3	
Lock0	0xxxxx[1] _0130
Lock1	0xxxxx[1] _0134
Lock2	0xxxxx[1] _0138
Lock3	0xxxxx[1] _013C

1. Value depends on the DCF Client Select assigned to the PASS module in the device. See chip specific PASS information for the Client Select assignment

In order to set the MIDLOCK field in the Lock0 register of password group 1 to the value 0x00FF and other fields to 0, the DCF record has to be added as shown in Table 82-39.

Table 82-39. Setting some bits in the MIDLOCK field

ADDR offset	DATA	Description
0x00	0x05AA_55AF	Start Record
0x04	0x0000_0000	
0x08	0x0000_00FF	PW Group 1, Lock0
0x0C	0xxxxx[1]_0110	Set MIDLOCK to 0xFF
0x10	0xFFFF_FFFF	Non-Programmed Addresses
0x14	0xFFFF_FFFF	
…	…	

1. Life cycle is governed by external life cycle management (SSCM).

Chapter 83
Tamper Detection Module (TDM)

83.1 Tamper Detection Module (TDM) chip-specific module information

The Tamper Detection Module provides a type of flash memory write protection mechanism that forces software to write a record associated with one or more blocks in a Tamper Detection Region (TDR) before the block(s) can be erased. An additional feature is provided to enable customers to override the protection mechanism for any TDR so that no record is required to be written before a block within the TDR can be erased.

Additionally, the TDM provides a mechanism for configuring individual flash memory blocks as One Time Programmable (OTP), that is, write-once, blocks.

The following sections provide details on the TDM configuration registers and DCF clients.

NOTE

The TDM configuration registers detailed in this section are DCF clients. In general, DCF clients can only be written once —their intended values must be carefully selected beforehand.

83.1.1 TDR*n*/Lock*m* OTPEN*m* register bit map

See the attached Flash Block Assignment spreadsheet for block naming and register bit maps.

83.1.2 OTPEN0 value

On this chip OTPEN0 is factory-programmed to 0x00010000.

83.2 Overview

The Tamper Detection Module provides a type of flash memory erase protection mechanism that requires software to write a record associated with one or more blocks in a Tamper Detection Region (TDR) before the block(s) can be erased. The mechanism does not *prevent* an erase operation on any block within a TDR—it requires a record to be written before the operation can execute. Collectively, the records are referred to as a "diary". At minimum, the diary serves as an erase counter. Because the diary record content is customer-defined, the diary can also serve as a history.

Up to 6 TDRs can be defined via DCF records.

The following figure shows the block diagram of the TDM and its interface to other system components.

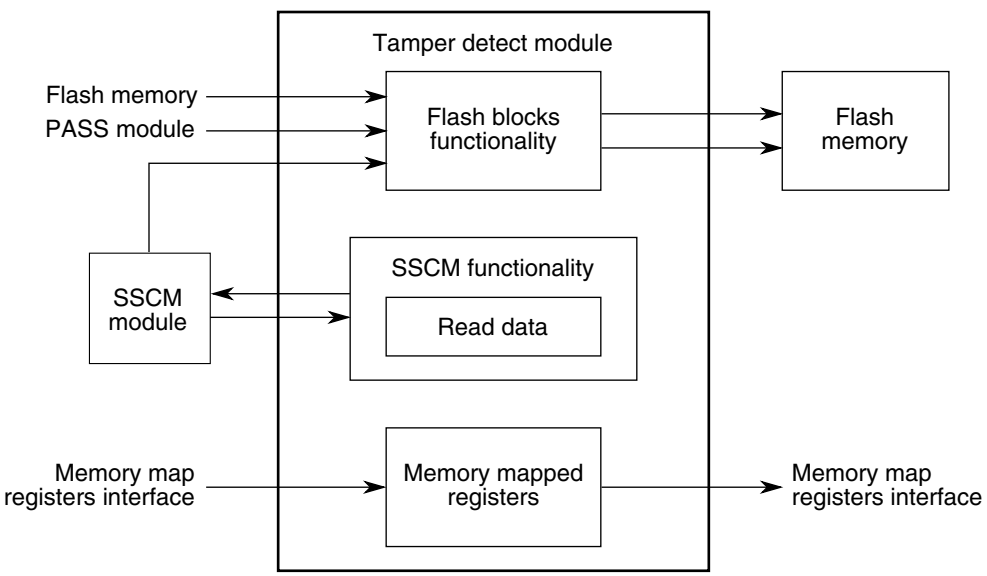


Figure 83-1. Tamper Detection Module block diagram

Security information is passed to the TDM by the SSCM module via the DCF bus. Among the data are:

- OTP (One Time Programmable) settings for each of the flash blocks
- Base address for the diary in the flash
- Block assignments for the TDRs
- Tamper region override information
- Software tamper region disable

The TDM module controls the flash blocks for erase operations and sets them as OTP.

The two flash buses are used to control the flash blocks for erase operation and set them as OTP. OTP means that flash erase of the entire block is disabled and the only words that are already erased can be programmed. Over-programming is not possible.

NOTE

The pass module provides the control for erase operation for each flash block as well as for each region of flash memory, and the PASS settings have priority over the settings of TDM.

The flash block indicates to the TDM the address of the programmed word. This address is internally decoded and provides a means to verify whether the block is part of any TDRs. The memory map registers interface provides a means to the user to read information of the TDM registers.

83.3 Features

The Tamper Detection Module supports these distinctive features:

- Erase counter (diary) for each TDR to record up to 256 write attempts on the flash blocks
  - The diary is divided into 6 parts (TDRs), each consisting of 256 64-bit records.
  - The diary block should be assigned as OTP region in flash.
  - The diary area can store any type of relevant information.
- Tamper Detection Region Block Assignment
  - Any flash block can be assigned to any TDR and be controlled.
- 32-bit Erase Lock DCF clients for each flash memory TDR defined.
- 32-bit OTP Enable DCF clients to disable the erase operation for an entire flash block
- Override TDR via DCF records[1]
  - Allows block assigned to a TDR to be unlocked for erase even if the TDR is locked for this operation.
- Memory mapped registers interface for reading via software of specific registers.

83.4 External signal description

There are no external signals driving or being driven by the TDM.

1. The Tamper Region Override DCF record is a "write one only" DCF record. If an override is applied to a TDR it is permanent and cannot be reversed.

83.5 DCF clients

TDM-related DCF records perform the following functions:

- Establish a permanent diary base address
- Define Tamper Detect Regions (TDRs) to monitor on-chip flash memory program/ erase activity
- Permanently disable one or more Tamper Regions
- Permanently disable ability for software to permanently override one or more TDRs
- Configure flash memory as One Time Programmable (OTP) on a per-block basis

The following table gives an overview on the TDM DCF clients implemented.

Table 83-1. DCF clients

DCF Client Name	Description	Strategy
Diary Base Address	Stores the base address of the tamper detection diary	write once, write 1 to 0 only
Tamper Region Override	Stores information of which TDR is to be overridden	write 0 to 1 only
Software Tamper Region Override Disable	Stores information to disable the software mechanism to override the TDR	write once, write 0 to 1 only
OTPEN[nb] (OTP Enable)	Stores information of which flash blocks is to be assigned as OTP (One Time Programmable) - nb means one register for each Lock register of the flash memory	write 0 to 1 only
TDR[n]_LOCK[m] Region Assignment	Stores information of which flash block is assigned to a TDR. • n specifies a TDR • m specifies the flash blocks as per the LOCK registers of the flash memory.	write 0 to 1 only

83.5.1 Diary Base Address (DBA) DCF client

This DCF client holds base address information of the diary. The following diagram depicts this client.

CAUTION

- This write-once DCF client can be written only one time. After one DCF record has written to this client, all subsequent writes are ignored.
- This DCF client is writable from 1 to 0 only.

DCF Client Address: 0x0000 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	DBA																															
Default	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 83-2. Diary Base Address (DBA) DCF client field description

Field	Description
0–31 DBA	Diary Base Address. This field holds the base address of the diary. This information is then used to verify if the diary being programmed matches any TDR region. **NOTE:** Diary Base Address must be on a 4 KB boundary.

83.5.2 Tamper Region Override (TO) DCF client

This DCF client holds override information for each TDR. The following figure shows the diagram for this register.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the override of a TDR permanent.

DCF Client Address: 0x0001 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TOE5	TOE4	TOE3	TOE2	TOE1	TOE0
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 83-3. Tamper Region Override (TO) DCF client field descriptions

Field	Description	
0–25 Reserved	This field is reserved and should be written with all zeroes when programming the TOE fields.	
26–31 TOE5 to TOE0	TDR Override Enable. Each TOE*n* field, where *n* represents the TDRs, holds information of the Override Enable status for each TDR.	
	0	Normal operation. Blocks assigned to this TDR can NOT be erased until the diary is written.
	1	Diary override. The blocks assigned to this TDR can be erased WITHOUT writing to the diary.

83.5.3 Software Tamper Region Override Disable (STO_DIS) DCF client

This DCF client holds information to disable the software mechanism to override the tamper detect regions.

CAUTION

- This write-once DCF client can be written only one time. After one DCF record has written to this client, all subsequent writes are ignored.
- This DCF client is writable from 0 to 1 only.

DCF Client Address: 0x0002 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	STO_DIS5	STO_DIS4	STO_DIS3	STO_DIS2	STO_DIS1	STO_DIS0
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 83-4. Software Tamper Region Override Disable (STO_DIS) DCF client field descriptions

Field	Description	
0–25 Reserved	This field is reserved and should be written with all zeroes when programming the STO_DIS fields.	
26–31 STO_DIS5 to STO_DIS0	Software TDR Override Disable TDR*n*. Each STO_DIS*n* field, where *n* represents the TDR number, holds information to disable the software mechanism to override TDR*n*.	
	0	Software tamper detect override mechanism supported
	1	Software tamper detect override mechanism disabled. Any attempt to override TDR*n* by loading a service key value in STO_KEY*n* register is ignored.

83.5.4 One Time Programmable Enable (OTPEN0) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK0 register of the flash memory. The following diagram depicts this client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the OTP assignment of a flash memory block irreversible.

DCF Client Address: 0x0008 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	LOWOTPEN[11:0]												MIDOTPEN[15:0]															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 83-5. One Time Programmable Enable (OTPEN0) DCF client

Field	Description	
0–3 Reserved	This field is reserved and should be written with all zeroes when writing to this client.	
4–15 LOWOTPEN	Low Block OTP Enable. This field affects flash memory blocks in Low address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module.	
	0	Corresponding flash blocks are not assigned as OTP
	1	Corresponding flash blocks are assigned as OTP

Table continues on the next page...

Table 83-5. One Time Programmable Enable (OTPEN0) DCF client (continued)

Field	Description	
16–31 MIDOTPEN	Mid Block OTP Enable. This field affects flash memory blocks in Mid address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module.	
	0	Corresponding flash blocks are not assigned as OTP
	1	Corresponding flash blocks are assigned as OTP

83.5.5 One Time Programmable Enable (OTPEN1) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK1 register of the flash memory. The following diagram depicts the client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the OTP assignment of a flash memory block irreversible.

DCF Client Address: 0x0009 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	HIGHOTPEN															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 83-6. One Time Programmable Enable (OTPEN1) DCF client field descriptions

Field	Description	
0–15 Reserved	This field is reserved and should be written with zeroes when writing to this client.	
16–31 HIGHOTPEN	High Block OTP Enable. This field affects flash memory blocks in High address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module.	
	0	Corresponding flash blocks are not assigned as OTP
	1	Corresponding flash blocks are assigned as OTP

83.5.6 One Time Programmable Enable (OTPEN2) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK2 register of the flash memory. The following diagram depicts the client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the OTP assignment of a flash memory block irreversible.

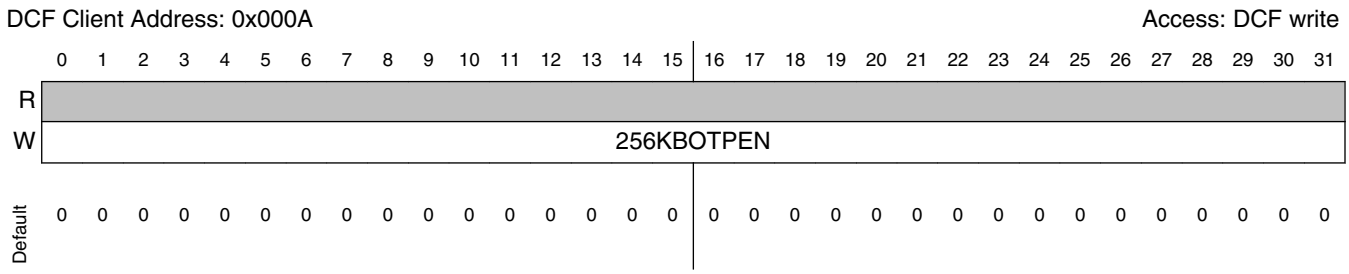


Table 83-7. One Time Programmable Enable (OTPEN2) DCF client field descriptions

Field	Description	
0–31 256KBOTPEN	256 KB Block OTP Enable. This field affects flash memory blocks in 256KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module.	
	0	Corresponding flash blocks are not assigned as OTP
	1	Corresponding flash blocks are assigned as OTP

83.5.7 One Time Programmable Enable (OTPEN3) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK3 register of the flash memory. The following diagram depicts the client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the OTP assignment of a flash memory block irreversible.

DCF Client Address: 0x000B — Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	256KBOTPEN															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 83-8. One Time Programmable Enable (OTPEN3) DCF client field descriptions

Field	Description	
0–15 Reserved	This field is reserved and should be written with zeroes when writing to this client.	
16–31 256KBOTPEN	256 KB Block OTP Enable. This field affects flash memory blocks in the upper range of 256KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module.	
	0	Corresponding flash blocks are not assigned as OTP
	1	Corresponding flash blocks are assigned as OTP

83.5.8 Tamper Region Assignment DCF client (TDRn_LOCK0)

This DCF client holds the assignment information of flash memory blocks to a specific TDR. Each flash memory block can be assigned to any TDR (*n* varies from 0 to 5). This DCF client holds the assignment information for only those blocks specified in the LOCK0 register in the flash memory. For that reason, the layout of this DCF client mirrors the LOCK0 register in the flash memory. The following diagram depicts the TDRn_LOCK0 DCF client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash memory block to a TDR irreversible unless the entire TDR is overridden.

DCF Client Address: 0x0014 + ((n x 4) + 0) Access: DCF write

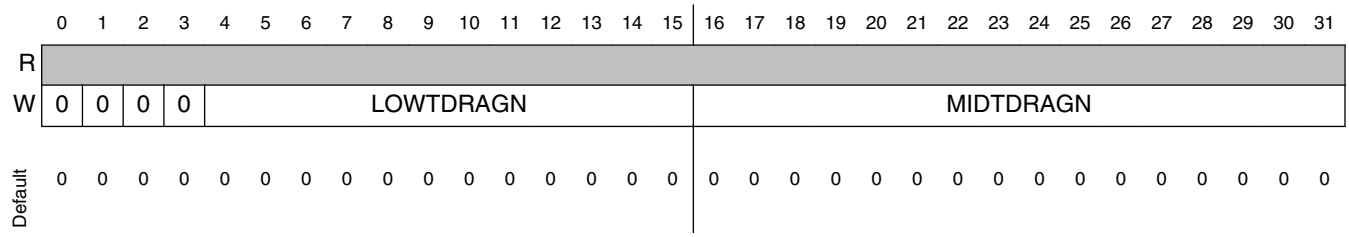


Table 83-9. Tamper Region Assignment DCF client (TDRn_LOCK0)

Field	Description	
0–3 Reserved	This field is reserved and should be written with zeroes when writing to this client.	
4–15 LOWTDRAGN	Low Block Tamper Region Assignment. This field affects flash memory blocks in Low address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module.	
	0	Corresponding flash blocks are not assigned to the TDRn
	1	Corresponding flash blocks are assigned to the TDRn
16–31 MIDTDRAGN	Mid Block Tamper Region Assignment. This field affects flash memory blocks in Mid address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module.	
	0	Corresponding flash blocks are not assigned to the TDRn
	1	Corresponding flash blocks are assigned to the TDRn

83.5.9 Tamper Region Assignment DCF client (TDRn_LOCK1)

This DCF client holds the assignment information of flash memory blocks to a specific TDR. Each flash memory block can be assigned to any TDR (*n* varies from 0 to 5). This DCF client holds the assignment information for only those blocks specified in the LOCK1 register in the flash memory. For that reason, the layout of this DCF client mirrors the LOCK1 register in the flash memory. The following diagram depicts the TDRn_LOCK1 DCF client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash memory block to a TDR irreversible unless the entire TDR is overridden.

DCF Client Address: 0x0014 + ((n x 4) + 1) Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	HIGHTDRAGN															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 83-10. Tamper Region Assignment DCF client (TDRn _LOCK1)

Field	Description	
0–15 Reserved	This field is reserved and should be written with zeroes when writing to this client.	
16–31 HIGHTDRAGN	High Block Tamper Region Assignment. This field affects flash memory blocks in High address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module.	
	0	Corresponding flash blocks are not assigned to the TDRn
	1	Corresponding flash blocks are assigned to the TDRn

83.5.10 Tamper Region Assignment DCF client (TDRn_LOCK2)

This DCF client holds the assignment information of flash memory blocks to a specific TDR. Each flash memory block can be assigned to any TDR (*n* varies from 0 to 5). This DCF client holds the assignment information for only those blocks specified in the LOCK2 register in the flash memory. For that reason, the layout of this DCF client mirrors the LOCK2 register in the flash memory. The following diagram depicts the TDRn_LOCK2 DCF client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash memory block to a TDR irreversible unless the entire TDR is overridden.

DCF Client Address: 0x0014 + ((n x 4) + 2) Access: DCF write

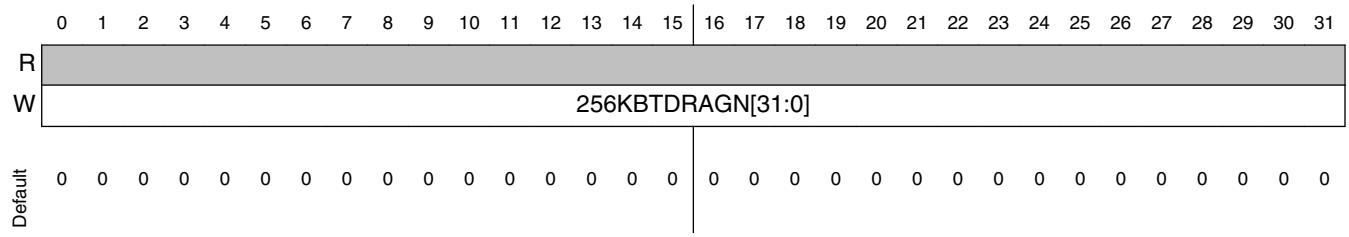


Table 83-11. Tamper Region Assignment DCF client (TDRn_LOCK2)

Field	Description	
0–31 256KBTDRAGN[31:0]	256 KB Block Tamper Region Assignment. This field affects flash memory blocks in 256KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module.	
	0	Corresponding flash blocks are not assigned to the TDRn
	1	Corresponding flash blocks are assigned to the TDRn

83.5.11 Tamper Region Assignment DCF client (TDRn_LOCK3)

This DCF client holds the assignment information of flash memory blocks to a specific TDR. Each flash memory block can be assigned to any TDR (*n* varies from 0 to 5). This DCF client holds the assignment information for only those blocks specified in the LOCK3 register in the flash memory. For that reason, the layout of this DCF client mirrors the LOCK3 register in the flash memory. The following diagram depicts the TDRn_LOCK3 DCF client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the assignment of a flash memory block to a TDR irreversible unless the entire TDR is overridden.

DCF Client Address: 0x0014 + ((n x 4) + 3) Access: DCF write

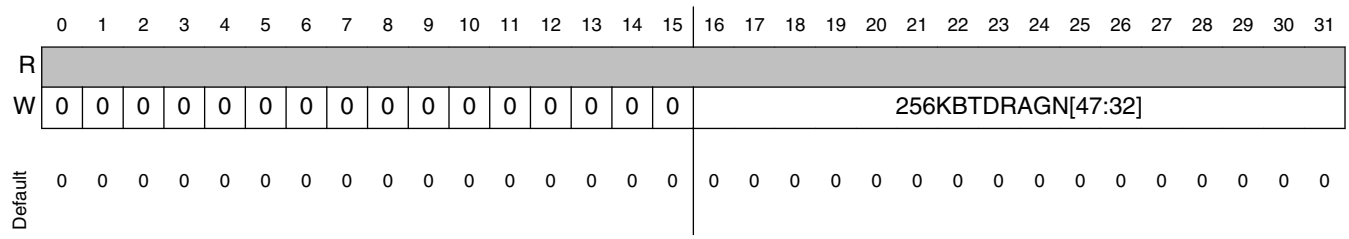


Table 83-12. Tamper Region Assignment DCF client (TDRn_LOCK3)

Field	Description	
0–15 Reserved	This field is reserved and should be written with zeroes when writing to this client.	
16–31 256KBTDRAGN[47:32]	256KB Block Tamper Region Assignment. This field affects flash memory blocks in 256KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module.	
	0	Corresponding flash blocks are not assigned to the TDRn
	1	Corresponding flash blocks are assigned to the TDRn

83.6 Memory Map/Register Definition

TDM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	TDR Status Register (TDM_TDRSR)	32	R	See section	83.6.1/3525
4	Last Flash Programmed Address Register (TDM_LFPAR)	32	R	0000_0000h	83.6.2/3527
8	Diary Base Address (TDM_DBA)	32	R	See section	83.6.3/3528
10	Software Tamper Override Key Region (TDM_STO_KEY0)	32	R/W	0000_0000h	83.6.4/3528
14	Software Tamper Override Key Region (TDM_STO_KEY1)	32	R/W	0000_0000h	83.6.4/3528
18	Software Tamper Override Key Region (TDM_STO_KEY2)	32	R/W	0000_0000h	83.6.4/3528
1C	Software Tamper Override Key Region (TDM_STO_KEY3)	32	R/W	0000_0000h	83.6.4/3528
20	Software Tamper Override Key Region (TDM_STO_KEY4)	32	R/W	0000_0000h	83.6.4/3528
24	Software Tamper Override Key Region (TDM_STO_KEY5)	32	R/W	0000_0000h	83.6.4/3528

83.6.1 TDR Status Register (TDM_TDRSR)

This register contains the current status, i.e., locked or unlocked, of each tamper region, i.e., TDR.

- If a tamper region is locked, the blocks assigned to that tamper region cannot be erased.
- If a region is unlocked, the blocks within the tamper region can be erased.

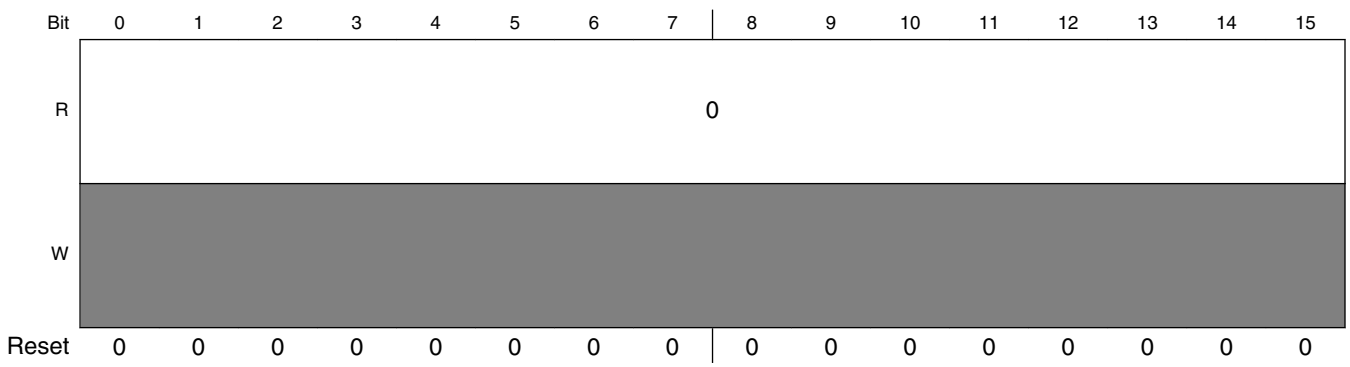
The register contains a status bit for each TDR. Status is determined as follows:

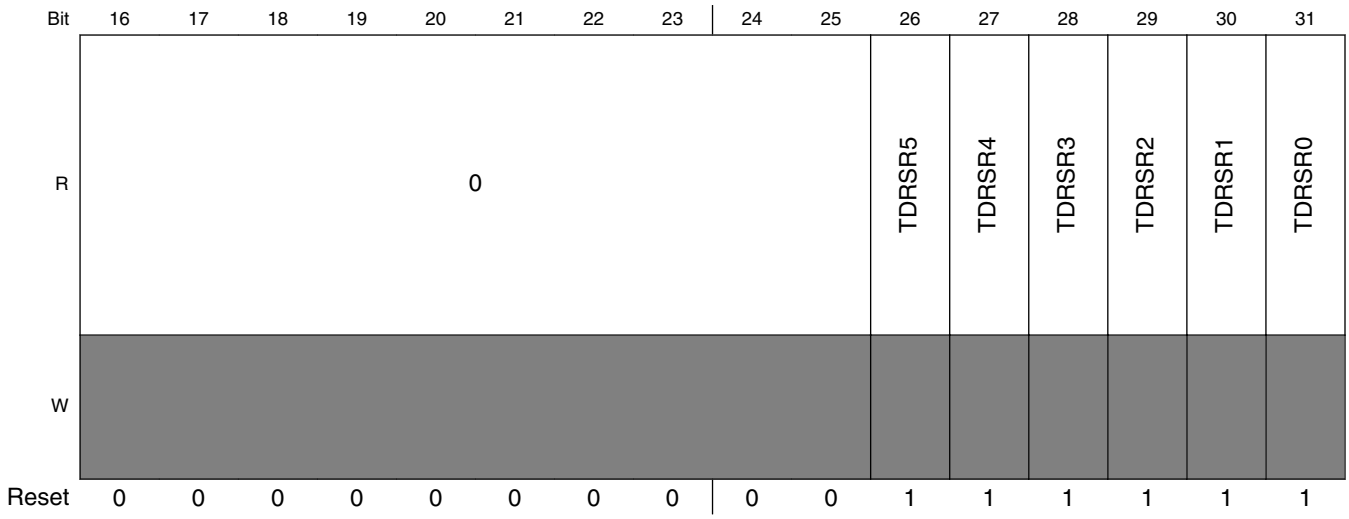
1. The default status for each TDR is ‘1’, which indicates that the erase operation is disabled for all flash blocks associated with the TDR.
2. The status bit for a TDR is set to ‘0’ (indicating that the erase operation for all flash blocks associated with the TDR is enabled) if either a successful programming operation to the TDR diary region has been performed or the TDR Override bit for that TDR is set.

NOTE

- TDRSR is a read-only register and writes have no effect on it. Write operations result in a transfer error.
- If the reset value for this register is determined by a DCF record, it will be noted in the chip-specific information at the beginning of this chapter. Otherwise, the reset value is as shown.

Address: 0h base + 0h offset = 0h





TDM_TDRSR field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 TDRSR5	Represents the status of TDR 5 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
27 TDRSR4	Represents the status of TDR 4 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
28 TDRSR3	Represents the status of TDR 3 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
29 TDRSR2	Represents the status of TDR 2 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
30 TDRSR1	Represents the status of TDR 1 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR

Table continues on the next page...

TDM_TDRSR field descriptions (continued)

Field	Description
	has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
31 TDRSR0	Represents the status of TDR 0 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.

83.6.2 Last Flash Programmed Address Register (TDM_LFPAR)

This register holds information of the address of the last successful programming operation provided from the flash memory to the TDM. Writes to this register are ignored and will generate a transfer error.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								LFPAR																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM_LFPAR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 LFPAR	Last Flash Programmed Address. These bits represent the address of the last successful programming operation.

83.6.3 Diary Base Address (TDM_DBA)

This register holds base address information of the diary. Any write to this register is ignored and generates a transfer error.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DBA																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value of this register is set via DCF record.x = Undefined at reset.

TDM_DBA field descriptions

Field	Description
0–31 DBA	Diary Base Address. These bits represent the address of the diary location in the flash. **NOTE:** The diary base address is established via the DBA DCF record. The reset value of this register is determined by that record.

83.6.4 Software Tamper Override Key Region (TDM_STO_KEY*n*)

The STO_KEYn register holds the service key for overriding the Tamper Detect Region n, where n denotes the Tamper Detect Region number.

Caution

1. The STO_KEYn registers can only be written once.
2. The associated DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored.

Address: 0h base + 10h offset + (4d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	STO_KEY																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM_STO_KEY*n* field descriptions

Field	Description
0–31 STO_KEY	Software Tamper Override Key Diary n. This field holds the service key value for overriding Tamper Detect Region n. By loading STO_KEYn with the correct signature of 0x55AA5A5A, Tamper Detect Region n is overridden, and the flash blocks assigned to Tamper Detect Region n are open for erase. This field has no effect when STO_DISn field of STO_DIS DCF client is set. **Caution:** The STO_KEYn registers can only be written once.

TDM_STO_KEY*n* field descriptions (continued)

Field	Description

83.7 Functional description

The primary purpose of the TDM is to provide a Tamper Detection mechanism by enabling an Erase Counter (or diary) for regions within the flash. These regions are called Tamper Detection Regions. The following sections cover the entire functionality of this module.

83.7.1 Flash erase counter and tamper detection

A flash region block or part of it is assigned for implementing the diary. The base address for the diary is defined by a DCF Record.

The diary is divided into 6 parts where each part consists of 256 × 64-bit records, to record up to 256 write attempts on the flash blocks assigned to each Tamper Detection Region. For correct operation, the diary should reside in a flash block that is assigned as OTP.

Each flash block can be assigned to any of the TDRs. Related to each TDR is a group of 4 x 32-bit internal hardware registers that are initialized by DCF records that define the tamper region. These registers mirror exactly the LOCK registers bits from the flash module.

Each TDRn_LOCK*m* register is loaded at boot time by the SSCM, which defines if the associated flash block is Locked for Erase in a particular TDR:

- ’1’ means the flash block is locked for erase
- ’0’ means the flash block is not locked for erase

Before performing an erase in one of the blocks locked for erase and associated to a TDR, a program operation has to be performed on the diary part for that TDR. As the diary should be OTP, successive writes are allowed on unprogrammed double words only. This ensure that diary records can only increase and that data recorded in the diary is not modifiable by software.

At the end of a successful double word program operation, the flash module indicates to the TDM the address of the programmed double word. TDM determines whether the programmed data was written to the correct TDR diary location and then unlocks the

blocks for erase operation for the corresponding TDR. TDM ensures that this erase operation on unlocked blocks is allowed until the completion of the erase sequence or until next desctructive reset.

NOTE

All blocks associated to a TDR where an erase operation is allowed can be erased all together.

The data programmed in the diary location is flexible, and may include count information or other important customer information.

NOTE

There is no restriction on the type or format of data that can be programmed in the diary, so software must verify whether the diary is full. If a TDR's diary area is full, the tamper detect mechanism can be disabled for that TDR by creating a Tamper Region Override (TO) DCF record with the TDR's corresponding TDR Override Enable (TOE) bit set to '1'. This allows erase operations to be performed to the TDR without any diary entry. Alternatively, if another TDR is available it can be defined to contain the same blocks to extend the diary length for that TDR.

The setting from “override tamper region lock” DCF record (Diary) has priority over the settings in the TDRn_LOCK0- 3 registers. In other words, if a TDR is unlocked for erase by the override tamper region lock, the TDR*n*_LOCK*m* bits have no effect on the TDR that has been overridden, but the tamper detection remains active for all TDRs not having the override bit set.

Moreover, if a block is locked by the PASS_LOCK*x*_PG*n* registers or by TDR*n*_LOCK*m*, the block is locked for erase.

NOTE

The settings in the Tamper Region Override Register have priority over the TDR*n*_LOCK*m* registers. The PASS module also has priority over the settings done in the TDM. For instance, if blocks in a TDR are unlocked for erase operation via TDM, but the same blocks are programmed to be locked for erase operation in PASS, the blocks end up being locked for erase operation.

83.7.2 Assigning blocks to Tamper Detect Regions (TDR)

Each TDR is defined by a group of 4 x 32-bit DCF records and clients, with each containing bits that map to specific blocks in a flash region. These mappings mirror exactly the LOCK registers bits from the flash module. Hence, the block assignment registers are named as TDR*n*_LOCK*m* where *n* is the number of a tamper region, that is, 0- 5 and *m* varies from 0-3 as flash contains four Lock registers.

In these DCF records:

- '1' means that the corresponding flash block is assigned to this TDR
- '0' means that the corresponding flash block is not assigned to this TDR (Default)

DCF clients are not visible to software. They can not be read nor written. They can only be initialized by DCF record within the UTest flash block during reset. The value of these registers can be evaluated by reading all the UTest DCF records and searching for records that correspond to these clients.

NOTE

No flash block should be assigned to more than one TDR. Each block should correspond to a single TDR. For example, if a single block is assigned to TDR0 and TDR1, then even if the Diary is programmed for both TDR0 and TDR1, this flash block will never be unlocked for erase operation.

83.7.3 Diary

The diary is a region of the flash memory where records of block erases for each TDR are stored. The format and size of the records are not fixed. The only requirements are that each diary has a maximum size of 2 KB, and the minimum size of any programming operation is 8 bytes to honor OTP restrictions. Therefore, each diary can hold a maximum of 256 64-bit entries.

A flash region block or part of it must be assigned for implementing the diary. The base address for the diary is determined by a value written in a DCF record. The diary can be placed within any flash block in flash array, and history records in it can be read by any core.

NOTE

To maintain the security of the TDR, the block where the diary is placed must be assigned as OTP within the OTP registers.

The base address of the diary is implemented using a DCF Client. The DCF client is only writable once and the default bit state is 1.

The DCF clients can not be read nor written by software. They can only be initialized by DCF record from within UTEST block during reset. The value of these registers can be evaluated by reading all the UTEST DCF records and searching for records that write these DCF clients.

83.7.4 Specifying the diary base address

The diary is a region of flash memory where records of block erases for each TDR are stored. The format and size of these records is defined by the customer. The only hardware constraints are that each diary has a maximum size of 2 KB and the minimum size of any programming operation is 8 bytes. Therefore, each diary region limits the erases of the assigned blocks to 256 erases.

The diary can be placed within any flash block in the flash array. To maintain the security of the TDR, the block where the diary is placed, must be assigned as OTP within the OTP registers, while in customer delivery or OEM production life cycle states.

The base address of the diary is implemented with a 20-bit DCF client that can be initialized by the SSCM during reset. The DCF client is writable once and the default bit state is ‘1’.

The base address register is not visible to software. It cannot be read nor written. It can only be initialized by DCF record within UTest flash during reset. The value of this register can be evaluated by reading all the UTest flash DCF Records and searching for record that writes this register.

Each TDR is assigned a 2 KB section of the diary. With 6 TDRs, 12 KB of diary space is required.

The diary base address must be at a 4 KB boundary. Therefore the least significant 12 bits of the base address must all be ‘0’.

The Base address of the diary is represented by:

Diary_Base_Address = 0bxxxx_xxxx_xxxx_xxxx_xxxx_0000_0000_0000

The base address of each TDR is shown below:

- TDR0 = Diary_Base_Address + 0b00_0000_0000_0000
- TDR1 = Diary_Base_Address + 0b00_1000_0000_0000
- TDR2 = Diary_Base_Address + 0b01_0000_0000_0000
- TDR3 = Diary_Base_Address + 0b01_1000_0000_0000
- TDR4 = Diary_Base_Address + 0b10_0000_0000_0000
- TDR5 = Diary_Base_Address + 0b10_1000_0000_0000
- End of the diary = Diary_Base_Address + 0b11_0000_0000_0000

83.7.5 TDR lock status

The status of whether a TDR is locked for erase is maintained by the TDM. This is accomplished by implementing a register with as many bits as the number of TDRs. The default state of these bits is 1, which is defined as "TDR Erase Blocked". See TDR Status Register (TDM_TDRSR) for further details about this register.

When the flash memory module has completed a programming operation, the address of the location programmed is output from the flash module. The TDM in turn compares that address to determine if it falls within the range of any TDR diary regions. If it is within any TDR, the corresponding status bit is set to 0 meaning "TDR Erase Enabled".

These status bits are all reset to 1 when the flash module signals an erase complete by asserting the PEC (Program Erase Complete) signal.

The state of the 6 status bits can be read through the TDRSR. See TDR Status Register (TDM_TDRSR). The address of the last successful programming operation provided from the flash memory to the TDM can be read through the LFPAR. See Last Flash Programmed Address Register (TDM_LFPAR).

83.7.6 TDR override

Before a block protected by a tamper detect region can be erased, a record must be written to the associated diary. Software must search through the diary to find the end of the records, where the next available diary location that can be written. After a maximum of 256 records are programmed to the diary, a TDR diary is full. Full means that at least 1 bit has been programmed to ‘0’ in each 64 bit double word within the 2 KB diary.

If software determines that a particular TDR diary is full it may override the diary to allow erasing to continue or it may choose to stop further erases and to flag an error.

Diary Override bits are implemented with a DCF client consisting of 6 bits, one for each TDR. See the device configuration information at the beginning of this chapter for TDM DCF client details.

The Diary Override bits are by default set to ‘0’; which means the diary must be written before an erase can occur. The DCF client can be written many times but the bits can only be written to ‘1’. Writes to ‘0’ are ignored. In this way, by default, all Tamper Detect Regions are active. The Tamper detect region diaries can be individually overridden by writing a DCF record that sets that corresponding bit to ‘1’. In this mode, the TDR status bit is permanently set to 0 to allow flash erase.

The application can also override the diary by loading a service key to the STO_KEY*n* program-visible register, where n denotes the tamper detect region. By loading the value 0x55AA5A5A into STO_KEYn register, the corresponding tamper detect region is overridden. This software override mechanism is disabled if the STO_DISn field of the STO_DIS DCF Client is set.

83.7.7 One time programmable (OTP)

83.7.7.1 Overview

Any flash block within the flash array can be assigned, at any time, to be OTP. Once a flash block is assigned OTP, it cannot be changed back.

OTP means that flash erase of the entire block is disabled and only 64-bit double words that are already erased (that is, flash content is 0xFFFF_FFFF_FFFF_FFFF) can be programmed. Over-programming is not possible.

Flash blocks are assigned as OTP by writing a DCF record. The block will become OTP after the next functional reset.

Appendix A
Release Notes for Revision 6

A.1 General changes

- In the OSC Digital Interface (XOSC) chapter:
  - In section "Oscillator bypass mode", changed "The power-down mode has priority over the bypass configuration for FXOSC." to "The bypass configuration is independent of the power-down mode of the oscillator." and removed the NOTE on FXOSC, as the feature is not supported.
- In the Flash memory controller chapter, removed descriptions on flash remap to flash:
  - In section Features, removed the phrase "on-chip flash memory" in the second from last bullet.
  - In section PFlash calibration remap support, removed sentence "Additionally, support is provided to reroute flash accesses to other on-chip flash memory addresses" and bullet "Addresses within any non-dedicated block of on-chip flash memory. Secure flash, UTEST flash, EEPROM, and BAF cannot be used as overlay targets" .
  - Removed section "PFlash calibration remap to flash memory".
- In the Embedded Flash Memory (c55fmc) chapter, added test mode disable seal information:
  - In section UTest NVM block, changed last sentence of first paragraph.
  - Added section Test mode disable seal.
  - Added register Test Mode Disable Password Check register (C55FMC_TMD).
  - In section Erase, updated step 3 of the erase sequence.
  - In section Array integrity self check, added note to step 2 of the array integrity sequence.
  - In section User margin read, added note to step 2 of the margin read procedure.
- In the Sigma Delta Analog-to-Digital Converter (SDADC) Digital Interface chapter:
  - Removed all descriptions referencing the external modulator feature:
    - [Addendum Rev.1] In sections "Features" and "External signal description" removed mention of external modulator feature.
    - In sections "Overview", "Functional description", "High-pass filter support", "Hardware triggering", and "Initialization information" removed mention of external modulator feature.
    - [Addendum Rev.1] Removed section "External modulator mode".
    - [Addendum Rev.1] In the MCR register changed EMSEL field to Reserved.
    - In SDADC_MCR[TRIGEN], SDADC_MCR[PDR], and SDADC_OSDR[OSD] field descriptions, removed mention of external modulator feature.
  - In the FCR register changed FOWEN field to Reserved.
- In the Analog-to-Digital Converter (ADC) chapter, removed ADC_ALTR register.
- In the Temperature Sensor (TSENS) chapter:
  - Removed section "Temperature sensor chip-specific module information" as all references to the high temperature digital output 2 signals (165°C operation) have been removed in the rest of the chapter.
- In the CAN (FlexCAN) chapter removed all references to CAN FD.
- In the Fast Ethernet Controller (FEC) chapter:
  - Added MII-Lite feature descriptions in the following sections:
    - Section Introduction
    - Section Block diagram
    - Section Features
    - Section External signal description
  - Added section 10 Mbps and 100 Mbps MII-Lite interface
- In the Boot Assist Flash (BAF) chapter, added NO_ECHO feature descriptions in the following sections:
  - Section Serial boot mode protocol
  - Section LINFlexD download protocol
  - Section FlexCAN download protocol
  - Section Download start address, VLE bit, NO_ECHO bit and code size
  - Section Protocol summary
- In the Memory Error Management Unit (MEMU) chapter: In the MEMU memory map, corrected the MEMU_PERIPH_RAM_OFLW instances to show only one instance, MEMU_PERIPH_RAM_OFLW0. In the MEMU_PERIPH_RAM_OFLWn register, corrected the "i" value in the address description.
- In the Fault Collection and Control Unit (FCCU) chapter:
  - [Addendum Rev.1] Removed section "NVM interface"
  - [Addendum Rev.1] Removed mention of "NVM" throughout the chapter, in the following sections: "Introduction", "Acronyms and abbreviations", "Main features", "EOUT interface", "Control Register (FCCU_CTRL)", "Configuration Register (FCCU_CFG)", and "CFG Timeout Register (FCCU_CFG_TO)".
- In the Password and Device Security Module (PASS) chapter:
  - In section "Flash Memory Read Protection", removed table "Flash Memory Read Protection Truth Table". The table has been moved to section Password and Device Security Module (PASS) chip-specific module information with modified description in the Censorship column and addition of the Debug Interface column.

A.2 Preface changes

- Added section Special terms.

- In section Acronyms and abbreviations :
  - "LFAST" changed to "Zipwire (LFAST/SIPI)".
  - Removed "SIPI".
  - "LINFlexD" changed to "LINFlex".
  - Added "STAC".

A.3 Introduction changes

- In section Introduction, removed "new" in first sentence.
- In section Target applications, changed "mid-end" to "mid-range" and changed "applications" to "target applications".
- In section Block diagrams :
  - Removed mention of "Dual Interrupt Controllers".
  - In figure "Peripherals allocation":
    - "DECIM" changed to "DECFILTER".
    - "SIPI" changed to "Zipwire".
    - I/O lines added to Zipwire, SIUL2, REACM, eTPU, eMIOS, IGF, and XOSC.
- In section Features summary :
  - Removed references to CAN "FD" modules.
  - In the first bullet for the Main CPUs, removed "identical".
  - In the fifth bullet for the Main CPUs, changed "Local" to "local".
  - In the bullet for the SRAM, removed the parenthetical phrase.
  - In the bullet for INTC, removed "serving two computational cores".
  - Rephrased bullet for UART.
  - Rephrased second bullet for LINFlexD instances that they are Masters "only".
  - Rephrased bullet for eTPU.
  - Added bullet for IGF.
  - Rephrased bullet for eMIOS
  - Added bullet for REACM.
- [Addendum Rev.1] In section Feature list, Table 2-1, MPC5743R column DSPI row, "3+2 MSB" changed to "4+2 MSB", and footnote "DSPI 4 and 3 are not supported" corrected to "DSPI 4 is not supported".
- In section Software debug and calibration, changed "Interprocessor bus" to "Zipwire" and changed "eTPU" to "eTPUs".

A.4 Embedded memories changes

- No substantial content changes

A.5 Signal Multiplexing and Signal Description changes

- In section System pin functions table "System pin functions":
  - Made editorial corrections to table header.
  - Added 144QFP column.
  - Corrected pin numbers for 176QFP.

A.6 Memory Map changes

- No substantial content changes

A.7 Device Configuration Format Records changes

- In section DCF record structure, Table 6-1 Parity bit description:
  - [Correction to change in Addendum Rev.1] Added "Even parity"
  - [Addendum Rev.1] Removed Note that the bit is not implemented for DCF Clients written from UTEST.
- In section DCF clients :
  - Unders subsection "DCF Client Special Strategy", modified Parity description to state even parity is expected of all DCFs which require parity evaluation.
  - In Table 6-5 :
    - [Addendum Rev.1] Corrected DCF Client Special Strategy value for "DCF_UTEST_Miscellaneous" to "SPRD, Triple voting, and PARITY".
    - Corrected DCF Client Special Strategy value for "ERROR0_PAD_SELECT" and "ERROR1_PAD_SELECT" to "SPRD, Triple voting, and PARITY".
- In DCF_UTEST_Miscellaneous DCF client :
  - [Addendum Rev.1] In stcu_CFB field description, corrected "online LBIST" with "offline LBIST".
  - Changed bit 15 (formerly "self_test_failure") to Reserved Read-only 0.
- In DCF_IVPR1 DCF client, added Note that this DCF client is put into effect only after BAF is executed and the application code starts executing, and that the value gets left-shifted by 8 bits when programmed in to the Core_1 IVPR.
- [Addendum Rev.1] In DCF_SELFTEST_CONFIG_IPS_2 DCF client, corrected System clock divider assignment to the bit fields:
  - 29 to 31: System clock divider 0
  - 26 to 28: System clock divider 1
  - 23 to 25: System clock divider 2
  - 20 to 22: System clock divider 3
- In section DCF_SELFTEST_CONFIG_IPS_3 DCF client :
  - Bits [5:7]: Removed "Aux_Mux_11_Divider_0_Division_Value" from the field name. The field name is now "Aux_Mux_10_Divider_0_Division_Value".
  - Bits [14:16]: Removed "Aux_Mux_7_Divider_0_Division_Value" and made the field Reserved Read-only 0.

A.8 e200z4d Core Complex Overview changes

- No substantial content changes

A.9 Core Description changes

- In section Local Instruction and Data Memory Overview, added a sentence each to the second and third paragraphs to clarify the "frontdoor access" and "backdoor access" references.

A.10 System Integration Unit Lite2 (SIUL2) changes

A.10.1 Chip-specific SIUL2 information changes

- No substantial content changes

A.10.2 SIUL2 module changes

- In the SIUL2_SCR0n register, changed FEC_MODE field bit 1 description from "MII" to "MII-Lite".
- In the SIUL2_MSCRn register, removed reference to "Automotive" pads in the field description and changed bit value "00" to Reserved.

A.11 Crossbar Switch (AXBS) changes

A.11.1 Chip-specific AXBS information changes

- Added section Hardware Implementation Dependent Register (HID1).

A.11.2 Crossbar switch module changes

- Master General Purpose Control Register (XBAR_MGPCR*n*) : Added note to register description re. unimplemented master ports.
- General operation : Removed phrase, ", other than the flash (if present),". Removed last paragraph that began with "If present, the flash slave port..."
- Features :
  - Replaced "64-bit data bus" with "Up to single-clock 64-bit transfer".
  - Replaced "Support for byte, 2-byte, 4-byte, 8-byte, and 32-byte burst transfers" with "Support for burst transfers of up to 16 beats of data".
- Memory Map / Register Definition :Edited introductory text and note. Added paragraph to note beginning with , "All references to the crossbar switch..."

• Removed bullet beginning with, "Operation at a 1-to-1 clock frequency..." from Features.

A.12 XBIC module changes

• Overview : Removed content which is not useful to customer.
• Features : Removed bullet beginning with "hready, hresp0, and hresp2..."
• Functional description : Section has been rewritten. Removed "XBIC H Matrix Definition" table.
• Block diagram :
  • Replaced cross references to figures with "the following diagram."
  • In XBIC system block diagram, corrected label on top green signal to "hready, hresp0, hresp2".
  • Removed figure, "XBIC block diagram".

• Changed "last transfer" to "first transfer" in XBIC_EAR register description.
• Minor editing. • XBIC Error Status Register (XBIC_ESR) : Previous errata e8310 integrated into the RM: Added note to register description. • XBIC Error Address Register (XBIC_EAR) : Previous errata e8310 integrated into the RM: Added note to register description.

A.13 Peripheral Bridge (AIPS-Lite) changes

A.13.1 Chip-specific AIPS-Lite information changes

• No substantial content changes

A.13.2 AIPS module changes

• Edited General operation.

A.14 System Memory Protection Unit (SMPU) changes

A.14.1 Chip-specific SMPU information changes

• No substantial content changes.

A.14.2 SMPU module changes

- In Figure 13-2 of Block diagram section: in upper right corner, changed "Internal peripheral bus" to "Register access"
- In Features section:
  - Removed what was the first list item: "Support for a maximum of 12 program-visible 128-bit region descriptors, accessible as four 32-bit words each. "
  - Changed first (formerly second) list item from "Each region descriptor defines an arbitrarily-sized space, aligned anywhere in memory" to "Arbitrarily sized protection regions alignable anywhere in memory."
  - Raised these list items one level in list hierarchy:
    - "Arbitrarily sized protection regions alignable anywhere in memory.
      - Region sizes can vary from a minimum of 1 byte to a maximum of 4 GB.
    - Read/write access control permissions are defined in the region descriptor.
    - Global cache-inhibit attribute indicator.
    - Hardware-assisted maintenance of the descriptor valid bit minimizes coherency issues.
    - Priority given to granting permission over denying access for overlapping region descriptors."
- In Hit determination section:
  - Replaced Boolean equation for hit determination with new sentence: "When RGD*n*_Word3[VLD] is 1, a region hit occurs when the requested address is between the start address (RGD*n*_Word0[SRTADDR]) and the end address (RGD*n*_Word1[ENDADDR]), inclusive."
  - Changed first Note from "The SMPU does not verify that ENDADDR >= SRTADDR" to "If ENDADDR < SRTADDR, no hit occurs."
- Changed title of Final evaluation and error terminations section (formerly was "Putting it all together and error terminations")

A.15 Intelligent AHB Gasket (IAHBG) changes

A.15.1 Chip-specific IAHBG information changes

- No substantial content changes

A.15.2 IAHBG module changes

- No substantial content changes

A.16 Platform Configuration Module changes

- No substantial content changes

A.17 SEMA42 module changes

- Edited two citations in Multi-core programming 101: software gates.

- Edited Introduction and removed note.
- Removed "Multi-core programming 101: software gates" section.
- Edited Features and block diagram.
- Memory map/register definition :
  - Edited introductory paragraph.
  - Changed access of Reset Gate Write (SEMA42_RSTGT_W) to W.
  - Changed access of Reset Gate Write (SEMA42_RSTGT_R) to R.
- Reset Gate Write (SEMA42_RSTGT_W) :
  - Edited register description and added note.
  - Added footnote to reset value.
  - Changed access of RSTGTN field to write only.
  - Edited description of RSTGDP field and changed access to write only.
- Reset Gate Read (SEMA42_RSTGT_R) :
  - Edited register description.
  - Changed access of RSTGTN field to read only.
  - Two most-significant bits of register:
    - Changed name to ROZ.
    - Changed field description.
    - Changed access to read only.

- Functional description : Removed references to internal signals and changed "AIPS" to "peripheral bridge".

A.18 Interrupt Controller (INTC) changes

A.18.1 Chip-specific INTC information changes

- No substantial content changes

A.18.2 INTC module changes

- In Memory map and register definition :
  - Added Figure 17-2 and surrounding text explaining write restrictions for PSR*n* and SSCIR*n*.
  - In INTC Priority Select Register (INTC_PSR*n*) and INTC Software Set/Clear Interrupt Register (INTC_SSCIR*n*), added "See Figure 17-2 for limitations on writing to this register."

A.19 Enhanced Direct Memory Access (eDMA) changes

- No substantial content changes

A.20 Direct Memory Access Multiplexer (DMACHMUX) changes

A.20.1 Chip-specific DMACHMUX information changes

- No substantial content changes

A.20.2 DMAMUX module changes

- Removed the address information of CHCFG1 register (base address + 0x01) and CHCFG8 (base address + 0x08) from the section "Enabling and configuring sources."

A.21 WKPU (non-MEDC18 SC) changes

- In the WKPU_NCR register NDSS0 field description, corrected "Set HID0[MCP_EN] to 1." to "Set HID0[EMCP] to 1."

A.22 Error Injection Module (EIM) changes

A.22.1 Chip-specific EIM information changes

- No substantial content changes

A.22.2 EIM module changes

- Memory map and register definition
  - Minor editorial changes: in each list item of first two unordered lists, capitalized first letter of first word
- Memory map and register definition
  - Edited and reorganized text in Error injection channel descriptor: function and structure

A.23 Clocking changes

- [Addendum Rev.1] In section Clock generation :
  - In figure Clock generation :
    - Added “LINCLK (System Clock Synchronous domain)” output to System Clock Selector's Divider 1.
    - Added “(System Clock Asynchronous domain)” text to the existing LINCLK output from Auxiliary Clock Selector 0's Divider 4.
  - Added NOTE: "Do not bring out the ADC clock (XOSC + PLL0) or its component (XOSC and PLL0) on the CLKOUT pad when the ADC is in use using the same clock."
- In section System clock frequency limitations, Table 22-2 :
  - Added "LINCLK (System Clock Synchronous domain)" to "SXBAR, System Ram Array".
  - Changed "LINCLK" to "LINCLK (System Clock Asynchronous domain)".
- In section Ethernet clocking, changed all instances of "MII" to "MII-Lite".
- [Addendum Rev.1] Previous errata ERR002345 integrated into the RM: Added section LINFlex clocking.

A.24 PLLDIG changes

- PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)
  - Removed the text "the binary equivalent of" from the field descriptions of INCSTP and MODPRD.

- PLLDIG PLL0 Control Register (PLLDIG_PLL0CR)
  - Updated reset value of reserved field at location 21(Reserved), and changed the access to RO.

- Frequency modulation
  - Updated the sentence "If center modulation is selected..." to "If center modulation is used..." in the Note "The device maximum operating frequency..."

- Register PLLDIG PLL1 Divider Register (PLLDIG_PLL1DV)
  - Replaced the register description with updated content.

- PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)
  - Updated MODSEL bitfield value descriptions for both 0 and 1.

- PLLDIG PLL1 Frequency Modulation Register (PLLDIG_PLL1FM)
  - Removed the sentence "Modulation period (PLL1FM[MODPRD]) and increment step..." from the register description.
- PLLDIG PLL1 Fractional Divide Register (PLLDIG_PLL1FD)
  - Removed the sentence "The dither disable (PLLDIG_PLL1FD[DTHDIS]) and..." from the register description.

- Frequency modulation
  - Updated figure Figure 23-14 to show correct modulation depth.

- In PLLDIG PLL1 Fractional Divide Register (PLLDIG_PLL1FD)
  - Replaced DTHDIS field with Reserved field at bit locations 14:15.

- Clock configuration
  - Updated equations f_{PLL1_PHI} and f_{pll1_VCO}.

- Minor editorial updates throughout.

A.25 CMU changes

• Main features
  • Updated second bullet to "CLKMN0_RMT monitoring with respect to CLKMT0_RMN ÷ $2^{CSR[RCDIV]}$ clock."
• In CMU Control Status Register (CMU_CSR)
  • Updated RCDIV field description.
• Added CMU_ prefix in front of register/field names that did not previously include it.
• CMU Frequency Display Register (CMU_FDR)
  • Updated register description placing items in a bullet list.
• CMU Interrupt Status Register (CMU_ISR)
  • Updated Note in register description, "All flags in the CMU_ISR are set..."
  • Updated bitfield description FHHI
  • Updated bitfield description FLLI
• Frequency meter
  • Added "[FD]" to end of CMU_FDR throughout where it was not previously included.

• CMU Interrupt Status Register (CMU_ISR)   • Updated the Note in the OLRI bitfield description, "When attempting to enter STOP mode..."
• Editorial updates.
• CMU Interrupt Status Register (CMU_ISR)   • Updated the Note in the OLRI bitfield description.
• CMU Frequency Display Register (CMU_FDR)   • Added NOTE in register description, "The CMU_FDR should be read only when CMU_CSR[SFM] = 0."

A.26 Clock Generation Module changes

• No substantial content changes

A.27 XOSC changes

• Editorial updates throughout.
• XOSC Control Register (XOSC_CTL)   • Updated XOSC_CTL[I_OSC] field access.

A.28 IRCOSC changes

• Frequency trimming calculation
  • Updated table contents replacing δF_{var_SW} with δf_{TRIM}.
  • Updated all instaces of "F" with "*f*" to represent frequency.

A.29 PRAMC module changes

- Read and write operations (formerly "Read / Write introduction" :
  - Section completely rewritten

A.30 Flash memory controller changes

A.30.1 Chip-specific PFLASH information changes

- No substantial content changes

A.30.2 Flash memory controller module changes

- [Addendum Rev.1]ECC on data flash accesses :
  - Clarification: If a non-correctable error is detected, the flash memory controller returns a value of 1555_1555h to the requesting master.

- Access protections :
  - Content previously located in the "Error termination" and "Flash error response operation" sections moved to this section.
- Error termination :
  - Section deleted
- Flash error response operation :
  - Section deleted
- Read cycles - buffer miss :
  - Section deleted
- Read cycles - buffer hit :
  - Section deleted

- Throughout:
  - Editorial changes

- Platform Flash Configuration Register 1 (PFLASH_PFCR1) :
  - Clarification: Following statement added to PFLASH_PFCR1[APC] field description: "For valid RWSC and APC combinations please refer to the device data sheet.
  - Following note in PFLASH_PFCR1[APC] field description reformatted into note list: "A value of '1' in the most significant bit causes the flash controller to function in retrograde/legacy mode, in which there is no pipelining between the sampling of flash read data and the presentation of the next address for flash lookup."
  - Added to note list in PFLASH_PFCR1[APC] field description: "Flash operation is not guaranteed for RWSC/APC combinations other than those specified in the data sheet."
  - Clarification: Updated portion of PFLASH_PFCR1[RWSC] field description that points to data sheet for valid values to state, "For valid RWSC and APC combinations please refer to the device data sheet. (was "The required settings are documented in the device data sheet")
  - Clarification: Following statement in PFLASH_PFCR1[RWSC] field description reformatted as item in note list for emphasis: "Higher operating frequencies require non-zero settings for this field for proper flash operation"
  - Clarification: Added to note list in PFLASH_PFCR1[RWSC] field description: "Flash operation is not guaranteed for RWSC/APC combinations other than those specified in the device data sheet"

A.31 Embedded Flash Memory (c55fmc) changes

A.31.1 Chip-specific C55FMC information changes

- No substantial content changes

A.31.2 c55fmc module changes

- Over-Program Protection 0 register (C55FMC_OPP0) :
  - Correction: Two most significant bits of the register always reset to 0 (previously shown as dependent on chip configuration)
  - Clarification: Footnote in register diagram replaced with footnotes stating that the reset value of each field (LOWOPP and MIDOPP) varies among devices
- Over-Program Protection 1 register (C55FMC_OPP1) :
  - Correction: Sixteen most significant bits of the register always reset to 0 (previously shown as varying among devices)
  - Clarification: Footnote in register diagram replaced with footnote stating that the reset value of the HIGHOPP field varies among devices
- Over-Program Protection 2 register (C55FMC_OPP2) :
  - Clarification: Footnote in register diagram replaced with footnote stating that the reset value of the A256KOPP field varies among devices
- Over-Program Protection 3 register (C55FMC_OPP3) :
  - Correction: Sixteen most significant bits of the register always reset to 0 (previously shown as varying among devices)
  - Clarification: Footnote in register diagram replaced with footnote stating that the reset value of the A256KOPP field varies among devices
  - Minor editorial fix

A.32 Decorated Storage Memory Controller (DSMC) changes

A.32.1 Chip-specific DSMC information changes

- No substantial content changes

A.32.2 DSMC module changes

- No substantial content changes

A.33 Flash Memory Programming and Configuration changes

- In section Creating the tamper detect diary, updated cross references to table "DCF Clients" in the DCF chapter.

A.34 Analog Peripheral Modules

- No substantial content changes.

A.35 Sigma Delta Analog-to-Digital Converter (SDADC) Digital Interface changes

A.35.1 Chip-specific SDADC information changes

- In section Clock source removed mention of external modulator feature.
- In section DMA/interrupt gating input from ETPU, modified description in second and third paragraphs.
- Added new section Sigma-Delta interface to eMIOS module with subsection DMA/interrupt gating input from eMIOS.

A.35.2 SDADC module changes

- Module Configuration Register (SDADC_MCR)
  - Updated the access rights to Reserved field (at bit location 13) to RW (was ROZ).
  - Updated the field description for Reserved field (at bit location 13), "This field is reserved and always has the value 0. Only the reset value shall be written to this register field."
  - Updated the access rights to Reserved field (at bit location 12) to RW (was ROZ).
  - Updated the field description for Reserved field (at bit location 12), "This field is reserved and always has the value 0. Only the reset value shall be written to this register field."

A.36 Analog-to-Digital Converter (ADC) changes

A.36.1 Analog-to-Digital Converter (ADC) chip-specific information

- [Addendum Rev.1] In SARADC channel map, Table 35-3 :
  - Added footnote to ADC0 and ADC1 channel 33 (ADC_BGREF): "This channel is functional from 4.5V to 5.5V of the corresponding reference (VDD_HV_ADR_SAR)."

A.36.2 ADC module changes

- Presampling channel enable
  - Updated formatting of table Table 35-160, added table title, and added a cross reference to the table in the preceding text.
  - Updated formatting of table Table 35-159, added table title, and added a cross reference to the table in the preceding text.
  - Added cross reference in first paragraph to the PSCR register.

- Main Configuration Register (ADC_MCR)
  - Updated the field access of Reserved field at bit location 6 (Reserved) to RW. Field description updated accordingly.

- Calibration (High Accuracy mode)
  - Updated content throughout.

- Parts of the ADC
  - Removed "TEST Interface" bidirectional i/o from Figure 35-3 .

- Calibration, BIST Control and status Register (ADC_CALBISTREG)
  - Replaced ACC_*x* fields with Reserved fields (bits 5,6,7,29,30).
  - Change access of Reserved bits 1,2.

[Addendum Rev.1] Added back ADC_MCR[TRGEN] and ADC_MCR[EDGE] fields.

- [Addendum Rev.1] Conversion time
  - Updated content of "Compare phase time (CT):" bullet.
  - Updated "Examples" excluding CALBISTREG[OPMODE] since that field is no longer implemented.

- Removed the text "CTU" throughout section "Memory Map/Register Definition".

- Crosstriggering Unit interface
  - Changed the signal name 'ctu_push' to 'ctu_EOC' in Figure 35-158.
- Conversion time
  - Changed section title from "Compare phase time" to "Conversion time".

- Features
  - Reformatted bullet "Modes of operation".
- Main Status register (ADC_MSR)
  - Replaced BCTU/CTU with BCTU, and CTU with BCTU.

• Self Test Analog Watchdog Register 4 (ADC_STAW4R)
  • Updated field description of THRH.
  • Updated field description of THRL.
• Self Test Analog Watchdog Register 5 (ADC_STAW5R)
  • Updated field description of THRH.
  • Updated field description of THRL.
• Removed section "User Defined Offset and Gain values".

• Calibration, BIST Control and status Register (ADC_CALBISTREG)
  • Added the Note "This setting is used only when the application..." to the NR_SMPL field description.
• Self Test Analog Watchdog Register 4 (ADC_STAW4R)
  • Reformatted Note in THRL field description.

• Register DMA Enable register (ADC_DMAE)   • In the DCLR bitfield value description for the case when DCLR=1, added note "When DCLR is set, ADC DMA channels should only execute read accesses on ADC_CDR*n* registers."
• Register Offset and Gain User Register (ADC_OFSGNUSR)   • Removed Note "A detailed description of the..." from register description.
• Calibration, BIST Control and status Register (ADC_CALBISTREG)   • Updated the Note in the NR_SMPL field description to "The 16 samples setting is used only when the application..."
• Editorial updates throughout.
• Added subsection Conversion time for BCTU interface to section Conversion time.

A.37 Decimation Filter (DECFILTER) changes

A.37.1 Chip-specific DECFILTER information changes

• No substantial content changes

A.37.2 DECFILTER module changes

• No substantial content changes

A.38 Temperature Sensor changes

• In Introduction and Functional Description changed number of digital outputs from 3 to 2. • In Introduction, Functional Description and Temperature threshold detection (digital output generation) changed number of high temperature thresholds from 2 to 1. • In Figure 37-1 deleted high temperature threshold 2. • Editorial changes and rephrased text in Introduction and Temperature threshold detection (digital output generation).
• "Temperature Sensor" title is changed as "Temperature Sensor (TSENS)".

A.39 Timers Overview changes

- No substantial content changes

A.40 System Timer Module (STM) changes

A.40.1 Chip-specific STM information changes

- No substantial content changes

A.40.2 STM module changes

- For reserved fields of following registers, added text to each field description: "This field is reserved."
  - STM Control Register (STM_CR)
  - STM Channel Control Register (STM_CCR*n*)
  - STM Channel Interrupt Register (STM_CIR*n*)

A.41 Software Watchdog Timer (SWT) changes

A.41.1 Chip-specific SWT information changes

- Added section SWT time-out events: configuring reset and interrupt requests.

A.41.2 SWT module changes

- In SWT Control Register (SWT_CR) changed Reserved from read-only and always zero to read/write

- In SWT Service Register (SWT_SR) description of WSC field, changed beginning of second sentence from "If the SWT_CR[KEY] bit is set" to "If the SWT_CR[SMD] field is 01b"
- In SWT Service Key Register (SWT_SK) description of SK field, changed beginning of second sentence from "If SWT_CR[KEY] is set" to "If SWT_CR[SMD] is 01b"

- In Overview changed final three sentences:
  - From: "If this servicing action does not occur before the timer expires the SWT generates an interrupt or hardware reset. The SWT can be configured to generate a reset or interrupt on an initial time-out. A reset is always generated on a second consecutive time-out."
  - To: "If this servicing action does not occur before the timer expires, the SWT generates an interrupt or hardware reset request. The SWT can be configured to generate a reset request or interrupt on an initial time-out. The SWT always generates a reset request on a second consecutive time-out."
- In SWT Control Register (SWT_CR) description of ITR field
  - Added a Note: "For a description of how this chip implements SWT reset requests resulting from SWT time-outs, see the chip-specific SWT information."
  - In descriptions of field values, changed "reset" to "reset request"
- In Time-out
  - Changed the description of a reset time-out result from "reset" to "reset request"
  - Edited and reorganized entire section

A.42 Periodic Interrupt Timer (PIT-RTI) changes

A.42.1 Chip-specific PIT information changes

- No substantial content changes

A.42.2 PIT module changes

- Previous Errata err7905 moved into documentation: In PIT_MCR register, added a note to MDIS bit field -"Always write....clock generation module."

A.43 Enhanced Modular IO Subsystem (eMIOS200) changes

A.43.1 Chip-specific eMIOS information changes

- No substantial content changes

A.43.2 eMIOS module changes

Changed the field description of eMIOS_MCR[0] to “Only write this field with the reset value.” See eMIOS200 (eMIOS)

- Removed the redundant footnote from the eMIOSx_UCDIS register.

A.44 Enhanced Time Processing Unit (eTPU) changes

A.44.1 Chip-specific eTPU2 Plus information changes

- No substantial content changes

A.44.2 eTPU module changes

- Changed the reset value of eTPU Engine 1 Time Base Configuration Register (eTPU_TBR_TBCR_ENG1) and eTPU Engine 2 Time Base Configuration Register (eTPU_TBR_TBCR_ENG2) (was 0000_0000h, is 2000_0000h).

A.45 Reaction Module 2 (REACM2) changes

A.45.1 Chip-specific REACM2 information changes

- No substantial content changes

A.45.2 REACM2 module changes

- No substantial content changes

A.46 Input Glitch Filter (IGF) changes

A.46.1 Chip-specific IGF information changes

- No substantial content changes

A.46.2 IGF module changes

- In Prescaler Register (IGF_PRESR*n*), changed description of FPRE field by adding formula to calculate rate of filter counter clock.

• In Filter types, changed "The Input Glitch Filter implements four main filter types" to "The Input Glitch Filter implements three main filter types."

A.47 Body Cross-triggering Unit (BCTU) changes

A.47.1 Chip-specific BCTU information changes

• No substantial content changes

A.47.2 BCTU module changes

• In TRG_FLAG, combined field setting descriptions into a single entry for 0 and a single entry for 1 (previously two entries for 0 and two entries for 1). Made editorial changes to the wording of these descriptions. • In SFTRG1 and SFTRG65, changed field setting for "Set software trigger" to 1 (was 0). • Editorial changes throughout.

A.48 CAN (FlexCAN) changes

A.48.1 Chip-specific FlexCAN3 information changes

• Throughout the subsections of "CAN chip-specific module information", removed all references to CAN FD.

A.48.2 FlexCAN module changes

• Minor editorial changes in register information.
• In CAN_ERRIAR register description, removed table showing RAM to memory map mapping, as the Chip-specific FlexCAN module section now shows the information for both FD-enabled and non-FD instances of the chip.

• In Module Configuration Register (CAN_MCR), added "This bit is not affected by soft reset" to the LPMACK, FRZACK, SOFTRST, and NOTRDY field descriptions. Changed sentence in description of field MDIS from "This is the only bit within this register not affected by soft reset" to "This bit is not affected by soft reset."
• In Memory Error Control Register (CAN_MECR) :
  • Revised the note in the register description.
  • Revised the ECRWRDIS field description.
• In Memory map/register definition, deleted the "FlexCAN Memory Partition for CAN FD" section.

• Removed line in FlexCAN module features about programmable clock source. • In FlexCAN memory mapping, changed reference to address offset in second paragraph. • In Table 47-6, prefixed "CAN_" to all register names.
• Changed "MCU" to "chip" throughout the chapter.

• Stop mode   • Removed the sentence "As a consequence, it will not...".
• FlexCAN module features   • Changed 64 to 96 in bullet "Flexible message buffers (MBs), totaling..."

• Memory Error Control Register (CAN_MECR)
  • Updated access of Reserved bit 12 to RW.
  • Updated access of Reserved bit 13 to RW.
  • Updated access of Reserved bit 15 to RW.
• Error Status Register (CAN_ERRSR)
  • Removed the sentence "If CAN_MECR[CEI_MSK] is..." from the field description of CEIF.

• Changed bit values for fields in Table 47-140.
• In Message buffer structure, removed note about configuring CAN FD frames.
• Changed access type and descriptions for reserved fields at bit locations 5, 9, 12, and 13 in CAN_MCR.
• In CAN_CTRL2, changed access type and descriptions for two new reserved fields at bit locations 30 and 31.
• Changed access type and description for reserved field at bit location 16 in CAN_CTRL2.
• Changed access type and description for reserved field at offset 15 in CAN_MECR. Changed description for reserved fields at offsets 12 and 13 in same register.

A.49 De-serial Serial Peripheral Interface (DSPI) changes

A.49.1 Chip-specific DSPI information changes

• No substantial content changes

A.49.2 DSPI chapter changes

• No substantial content changes

A.50 Zipwire changes

• No substantial content changes

A.51 SIPI module changes

• Updated bit field access for SIPI_ACR[ADCNT] and SIPI_ACR[Reserved] to read write.

A.52 LFAST module changes

- In Memory map and register definition, added note "Read/Write accesses to all unimplemented registers and write accesses to Read only register will return a Transfer Error.".
- Updated the register-access value of LFAST Rx ICLC Interrupt Status Register (LFAST_RIISR) register from R/W to W1C.

A.53 Fast Ethernet Controller (FEC) changes

A.53.1 Chip-specific FEC information changes

- In section FEC interface selection, table "FEC interface selection", changed "MII" to "MII-Lite". In the footnote added that full MII is not supported on this chip.

A.53.2 FEC module changes

- In Frames received with CRC error (FEC_IEEE_R_CRC), ensured that the reset value is shown as Undefined.

A.54 SENT Receiver (SRX) changes

A.54.1 Chip-specific SENT-SRX information changes

- [Addendum Rev.1] Added section SRX_GBL_CTRL[NIB_LEN_VAR_LIMIT] setting.

A.54.2 SRX/SENT module changes

- In Global Control Register (SRX_GBL_CTRL), added a note in bitfield NIB_LEN_VAR_LIMIT.
- In Global Control Register (SRX_GBL_CTRL) changed range of values from "1 to 255" to "0 to 255" in GBL_CTRL[TSPRC] field description.
- Previous errata e8082 integrated into reference manual: In Fast messages buffers added the note "In the case of a fast message overflow and continuous reading of register …".

A.55 LINFlexD changes

A.55.1 Chip-specific LINFlexD information changes

• No substantial content changes

A.55.2 LINFlexD module changes

• Updated section: Baud rate generation
• Existing note: 'The LINFlexD module in UART mode does not support communication ...' present in UART_CR register is now associated with Errata err6350.
• In section: Memory map and register description,   • In register 'LINESR' updated the bitfield description of 'OCF'
• Added section: LINFlexD Clock Tolerance
• In section Memory map and register description:, updated note 'In Master Mode, read access to IFMI register and ...'

• In section Timeout error, heading Response timeout mechanism:
  • Changed 'Nominal_time_out-Correction_factor' to 'Nominal_time_out + LINFlexD_LINTCSR[CNT]'
  • Added note 'The value of nominal_time_out ... '

• In section Memory map and register description: • Added note: "Any access to A0 location ... transfer error."

• Minor editorial updates throughout
• In section: Timeout, figure: Timeout, changed OC1 to OC2
• In section: Timeout error heading: Header timeout mechanism,
  • Updated bullet: "If the counter starts after 11 ... "
  • Updated paragraph: "The reset value of HTO ... "
• Updated note: "The LINFlexD module ..." present in UARTCR register.

• Updated section:Timer
• In section: Memory map and register description
  • Updated the access of register UARTCTO to R in table: LINFlexD memory map
  • Removed phrase 'free running' from bit LINTCSR[CNT].

A.56 Reset and Boot changes

- Added references to long external reset and made other corrections on references to phases.
  - In section *PHASE1[DEST]* Phase: temporization and monitoring setup, added that this phase can be entered from IDLE on an enabled, long external 'functional' reset event.
  - In section *PHASE1[FUNC]* Phase: temporization and monitoring setup, corrected description that this phase is "duplication of PHASE1[FUNC] after self-test" to "duplication of PHASE1[DEST] after self-test".
  - In section *PHASE2[FUNC]* Phase: flash initial configuration :
    - Corrected description that this phase is "entered on exit from PHASE2[DEST]" to "entered on exit from PHASE1[FUNC]".
    - Corrected description that the reset state machine "exits PHASE2[DEST] and enters PHASE3[DEST]" to "exits PHASE2[FUNC] and enters PHASE3[FUNC]".
    - Corrected description that this phase is "duplication of PHASE2[FUNC] after self-test" to "duplication of PHASE2[DEST] after self-test".
  - In section *PHASE3[FUNC]* Phase: device configuration monitoring, references to "shortened 'functional' reset event" modified to "shortened external 'functional' reset event" for consistency.
  - In section *IDLE[FUNC]* Phase figure "RGM state machine":
    - In the legend, clarified text description for pink and blue arrows.
    - In the legend, added new arrow named "long external 'functional' reset".
    - Added transition arrow from IDLE[FUNC] to PHASE1[DEST] for "long external 'functional' reset".
  - In section Module status during reset sequence table "Module status during reset phases":
    - Clarified text description for pink and blue arrows.
    - Added new arrow named "long external 'functional' reset".

A.57 Reset Generation Module changes

- No substantial content changes

A.58 Boot Assist Flash (BAF) changes

A.58.1 Chip-specific BAF information changes

- [Addendum Rev.1] In section Search boot header and boot options, corrected mention of "4 × 16 KB blocks" to "5 × 16 KB blocks" to match implementation.
- [Addendum Rev.1] Added section Serial boot password behavior.

A.58.2 BAF module changes

- Updated the field name in Table 57-23.
- Updated BAF NMI Event Handling.

• Editorial updates made throughout the chapter. • In section Resources, modified FCCU bullet item to refer to "SWT1 and SWT3".
• Updated Introduction. • Updated Download start address, VLE bit, NO_ECHO bit and code size.
• Added topic IVPR1_CPU1 - BAF set IVPR1 for CPU1 DCF Client. • In Table 57-13, added the row "IVPR1 for CPU1" and adjusted address range of the Reserved row.

A.59 System Status and Configuration Module (SSCM) changes

A.59.1 Chip-specific SSCM information changes

• No substantial content changes

A.59.2 SSCM module changes

• In SSCM System Status (SSCM_STATUS) register's CER field description, changed first sentence and added sentence:
  • was: "This field indicates that the SSCM has detected a configuration error during reset while loading DCF clients."
  • now: "This field indicates that the SSCM has detected a configuration error during reset while loading initial device configuration. See the chip-specific SSCM information for details about sources of the error."

• Editorial changes.
• In Memory map and register definition, added additional text ("that is, you should not..") to note regarding register accesses.
• Revised CERS-DCF Mechanism. • In CER, added cross-reference to CERS-DCF Mechanism.
• In Life Cycle, added "Life cycle sequence" figure.
• Note updated in Memory map and register definition from "SSCM does not support transfer errors in memory holes" to "SSCM does not generate transfer errors in all memory holes".
• In SSCM System Status (SSCM_STATUS), added the note "Reset value of 1 is correct in case the device is secured" in reset value of field SSCM_STATUS[5]. • In SSCM System Memory and ID Register (SSCM_MEMCONFIG), changed all instances of "JTAG ID" changed to "JTAG Part ID" • In Life Cycle, deleted the sentence "The priority of the entries is from top to bottom, so the first row that applies determines the resulting Life Cycle" and added in note the sentence "Any other combination of values will result in a system reset." • Updated Table 58-11.

A.60 Power Management changes

- In secction Supply Concept, Figure 59-2 :
  - Removed blocks LVD_FEC, LVD_JTAG, LVD_MSC_3V3, LVD_MSC_5V0 and the pads and lines connecting to them (VDD_HV_IO_MSC, VDD_HV_IO_FEC, and VDD_HV_IO_JTAG).
  - Corrected "VRC_CTL" to "VRC_CTRL".

A.61 Power Management Controller digital interface (PMC_dig) changes

A.61.1 Chip-specific PMC information changes

- In section PMC registers whose reset values can be defined by DCF clients, modified first sentence that implied that DCF control could be disabled.
- Added section PMC registers with chip-specific reset values.

A.61.2 Power Management Controller changes

- In section ADC Interface table "ADC Interface", changed the lvd_fec, lvd_msc, and lvd_jtag rows (S. No. 29, 30, 33 to 36) to Reserved.
- In the PMC_ESR_TD register:
  - Modified first sentence of the register description.
  - In the TEMP1_2, TEMP1_0, TEMP0_2, TEMP0_0 field descriptions, clarified descriptions on assertion and clearing of the bit, and removed the Notes.
- Updated PMC_REE_TD, PMC_REE, PMC_RES, and PMC_ADC_CS reset values to footnotes referring to the chip-specific information section.
- In the PMC_RES register, in the NOTE in the register description, removed sentence "If this register is programmed through DCF client…".

A.62 Power Control Unit changes

- No substantial content changes

A.63 Mode Entry Module changes

- In the MC_ME_CCTL2 register STOP0 field access to read-only.
- In section Software Considerations for Preventing the Blocking of Mode Transitions, in the sentence following "Service any pending wakeup events when transitioning to low power modes", removed "RTC", as the feature is not supported.

A.64 Calibration and Debug changes

- No substantial content changes

A.65 Core Debug Support changes

- No substantial content changes

A.66 DCI changes

- No substantial content changes

A.67 JTAGC module changes

A.68 CJTAG changes

- No substantial content changes

A.69 JDC changes

- No substantial content changes

A.70 SPU changes

• No substantial content changes

A.71 JTAGM changes

• In figures "JTAGM block diagram" and "JTAGM to DCI serial interface diagram" changed "DCI_TDO" to "TDO".
• In Memory mapped registers, added text saying that JTAGM supports only 32-bit write accesses to its registers.

A.72 DTS changes

• No substantial content changes

A.73 NAR changes

A.73.1 Chip-specific NAR information changes

• No substantial content changes

A.73.2 NAR changes

• No substantial content changes

A.74 Nexus Module changes

• No substantial content changes

A.75 NXMC changes

A.75.1 Chip-specific NXMC information changes

- No substantial content changes

A.75.2 NXMC module changes

A.76 Functional Safety changes

- No substantial content changes

A.77 CRC module changes

- In CRC_CFG*n*[SWAP], updated bit description text from "The swap operation is a bit-by-bit rotation of the content" to "The swap operation is a bit-by-bit swapping of the content".
- In CRC_CFG*n*[INV], removed the bit-description text "in case of CRC-CCITT polynomial, the inversion operation is applied on the 16 least-significant bits.".
- Updated title of Signal description section to External signal description.
- In Block diagram, updated reference to Main features section to see the number of contexts of CRC.

A.78 Memory Error Management Unit (MEMU) changes

A.78.1 Chip-specific MEMU information changes

- In section Reporting table implementation, modified description on errors that are reported to System RAM reporting tables and Peripheral RAM reporting tables.

A.78.2 MEMU changes

- In Design overview, deleted "The MEMU block has three instantiations of the basic reporting block along with a common CPU interface for the user software to access the storage block.".
- In Overview, deleted the paragraph beginning with "The Control, Error Flag, and Debug".

A.79 Fault Collection and Control Unit (FCCU) changes

A.79.1 Chip-specific FCCU information changes

- Added section FCCU fault recovery after NMI reaction explaining handling of NMI ISRs.
- In section FCCU chip-specific register reset values
  - Changed section title. Section title was previously "FCCU register availability and default values". Now the section only shows registers that are available for this chip. Also changed the lead paragraph and the table title to match.
  - In Table 78-1 :
    - [Addendum Rev.1] Removed the registers that are unimplemented on this chip.
    - Added rows for the FCCU_CFG, FCCU_CFG_TO, and FCCU_NCF_TO registers as their reset values are chip-specific.
    - Corrected mnemonics for FCCU_NCF_En registers.
    - Row for FCCU_MCS was added in Addendum Rev.1 but removed from this release, as the reset value is now described in the register section.
- In section Error signal flow diagram, Figure 78-1, added footnote to the "FCCU_EOUT_SIG_EN[0:3]: corresponding enable bit ==1?" block: "This condition must be satisfied only when FCCU is configured for Bistable fault-output mode (FCCU_CFG[FOM])."
- In section FCCU failure inputs from other modules :
  - [Addendum Rev.1] Updated NOTE to include channels 51 and 52, and clarified to mention both MBIST and LBIST.
  - In Table 78-3 :
    - Removed the entire "Fault input" column.
    - Modified "Description" for the following NCFs: 0 to 3, 6 to 8, 28 to 30, 35 to 37, 49, 50, 88, and 89.
    - [Addendum Rev.1] In NCF 25 and 26, modified "Source" and "Description".
    - In NCF 51 and 52 updated "Error Reaction Path Check" and "Description".
    - Removed rows for NCFs 65 to 68.
- [Addendum Rev.1] Added section FCCU_CFG[FCCU_SET_AFTER_RESET] explaining that the bit is not used on this chip.

A.79.2 FCCU module changes

- In Definitions, after the list of applicable reset types, added this information: *For more information on each type of reset, see the MC_RGM and reset chapters.*
- Editorial changes throughout.

- [Addendum Rev.1] In the FCCU_CFG_TO register description, added this note to the register's function description: *When the configuration watchdog timer expires, FCCU changes the value of this register to its reset value.*
- [Addendum Rev.1] In the FCCU_EINOUT register description, added this note to the description of the EOUT1 and EOUT0 fields: *When the configuration watchdog timer expires, FCCU changes the value of this field to its reset value.*
- In the FCCU_STAT register description, changed the access type of the ESTAT field from *RW* to *RO*.

• In FSM description, added the paragraph that begins with the sentence *Multiple faults can occur at the same time.*
• In FSM description, added the paragraph that begins with the phrase *After moving to FAULT state*. • In OPR, added statement about OP15.
• In the FCCU_CFG register description, in the description for the FCCU_SET_CLEAR field, changed *by writing into these fields with 00 or 10 or by going to CONFIG state* to *by writing into these fields with 00 or 10*. • In the FCCU_CFG register description, for the FCCU_SET_CLEAR field, changed the meaning of the value 01 from *Error Pin goes in Faulty state* to *Both error signals are set to low*. • In the FCCU_CFG register description, for the FCCU_SET_CLEAR field, changed the meaning of the value 11 from *write: Error Pin goes OK state (write not possible when FCCU already in the T min timer phase or if FSM enters in FAULT state by capture of a fault)* to *Both error signals are set to high (write not possible when FCCU is already in the T_min timer phase or if FCCU enters Fault state by capture of a fault)*.
• In FSM description, added two paragraphs that begin with the phrase *All pending faults that occur during CONFIG state*.
• In Main features, changed *Fault detection collection* to *Fault detection and collection*. • In Block diagram, changed the stem sentence for the table to *This table describes the FCCU submodules.*
• [Addendum Rev.1] In FCCU Output Supervision Unit, changed the first sentence of the third paragraph to *There is a 'do nothing' input coming from the FCCU that indicates that the FCCU is programmed for no reaction for ALL FAULTS*
• In Definitions, changed the descriptions of the reset types from a list to a table, clarified the description of the destructive reset, and corrected the descriptions of the long and short functional resets.
• In the FCCU_CFG register description, added this note: *If you specify a new value for any of the fields in this register that affect the EOUT signals while the EOUT fault-indication timer is running (FCCU is indicating a fault on the EOUT signals), FCCU doesn't use the new settings you specified until after the EOUT fault-indication timer expires (FCCU stops indicating a fault on the EOUT signals).*
• Changed all references to the IRCOSC and RCOSC clocks so they are in terms of CLKSAFE, which represents either CLKSAFE0 or CLKSAFE1, both of which appear on the boundary of the module. • In EOUT interface, added this sentence: *The EOUT frequency is generated by dividing the CLKSAFE frequency by a fixed factor of 2^{18}.*
• Rewrote Signals (previously titled "Pinout"). • In the FCCU_DELTA_T register description, rewrote the field description for DELTA_T.
• In Introduction, changed *when the system clock is faster than the IRC clock* to *when CLKSYS is faster than the CLKSAFE clocks*. • In Acronyms and abbreviations, removed the row for IRCOSC. • In Acronyms and abbreviations, added a row for CF. • Rewrote NMI/WKPU interface. • Revised Figure 78-8 to use signal abbreviations that match those used in NMI/WKPU interface and now described in Signals. Also changed *All NMI signals are shown active-low* to *All NMI signals shown are active-low*.
• In Figure 78-72, added the following note: *The fault sources shown are examples only.*
• In the Noncritical Fault Timeout (FCCU_NCF_TO) register description, deleted the register reset value and replaced it with a footnote to indicate that it is chip-specific. Also edited the register description for grammar and style. • In the Noncritical Fault Configuration (FCCU_NCF_CFG*n*), Noncritical Fault State Configuration (FCCU_NCFS_CFG*n*), Noncritical Fault Enable (FCCU_NCF_E*n*) and Noncritical Fault Timeout Enable (FCCU_NCF_TOE*n*) registers, deleted the register reset value and replaced it with a footnote to indicate that it is chip-specific. Deleted the following redundant note: *The reset value is chip-specific. See the chip-specific FCCU information.* Edited the registers description for grammar and style.
• In the FCCU_MCS register description, changed the reset value of the register to 0 with a footnote to indicate that the MC_ME module changes the value of the register immediately after the chip leaves RESET mode. Rewrote the rest of the register description and bits description for clarity.

• For Reserved field of Configuration (FCCU_CFG)
  • Changed the field access from read-only and always zero to read/write.
  • In field description, added text: "Always write the reset value to this field."

• In the FCCU_CFG register description, added a footnote to the diagram to indicate that the entire register reset value is chip-specific.Removed the footnote for the field that occupies bits 26–31 and changed that field's description to *Always write the reset value to this field.* Changed the access type of some reserved fields. Edited the register description for grammar, style, and clarity. • In the Memory map, changed the register access type of Noncritical Fault Timeout Enable (FCCU_NCF_TOE*n*) to *R/W*. • In the FCCU_CFG_TO register description, deleted the register reset value and replaced it with a footnote to indicate that it is chip-specific.
• In the FCCU_STAT register description, in the description of the ESTAT field, deleted this paragraph: *SW can write this bit with a different value. The new value is valid for 1 clock cycle, thereafter it returns to indicate the actual value being driven from FCCU. For example in non-faulty states, this bit reads 0 and can be set for one cycle to 1. Similarly in faulty state, this bit reads 1 and SW can write it to 0 and after one cycle the bit reads 1 again.* Edited the ESTAT description for style.
• In the FCCU_CTRL register description, rewrote the description of the DEBUG field. • In Register descriptions, replaced the two paragraphs that begin with *The FCCU generates an access error in the following cases* with a paragraph that begins with *Follow these register-access guidelines*.
• In Use cases and limitations, changed all occurrences of *Tmin* to *T_min*. • In the FCCU_DELTA_T register description, added this note to the end of the description of the DELTA_T field: *The specified values for DELTA_T may vary ±2% due to the post-trimming inaccuracy of the CLKSAFE clocks.* • In Introduction, removed this sentence from the list because it duplicates information in the description of FCCU_DELTA_T[DELTA_T] and provides the wrong maximum value for DELTA_T: *T_min = 250 µs + delta_T, with delta_T parameter being configurable by SW up to 16.67 ms*. • In STCU interface, changed *This feature is optionally programmable inside the STCU.* to *This feature is optionally programmable using MC_RGM.*

• In Introduction :
  • In the sublist for the list item *Configurable internal reactions for each NCF*, changed *No reset reaction* to *No reaction*.
  • Changed the list item *After power-on the error out pins have high impedance. FCCU goes to operational state only on software request.* to *After power on, the EOUT signals have high impedance. They show operational state only on software request.*

• In Definitions, in the list item *HW recoverable fault*:
  • Changed *until the fault cause is asserted* to *until the fault cause is deasserted*.
  • Changed *That is, if 0 on the fault signal indicates fault, then the status flags are valid until the fault line is '0'* to *That is, if logical 0 on the fault signal indicates fault, then the status flags are valid as long as the fault line stays at 0*.
  • Changed *Typically the fault signal is latched in an external module at the FCCU* to *Typically the fault signal is latched external to the FCCU in the module where the fault occurred*.

• In FSM description :
  • Edited the content for style and usage.
  • In the list item *Configuration*, changed *The configuration state is used only to modify the default configuration of the FCCU* to *Used only to modify the configuration of FCCU from its default*.
  • In the stem sentence for the second list, changed *The transition from NORMAL/ALARM state goes along with the generation of* to *The transition from Normal/Alarm to Fault state goes along with the generation of*.

• In section Block diagram, Figure 78-2 :
  • Corrected bidirectional arrow between "FAULT" and block "FAULT intf" to a unidirectional arrow from "FAULT" to "FAULT intf".
• In section FSM description in the second paragraph of the first bullet, modified sentence "The permanent configuration lock can be disabled by a reset of the FCCU".
• In section FCCU Output Supervision Unit :
  • In the third paragraph, corrected "an enabled fault" to "a disabled fault".
  • In the fourth paragraph, corrected "the FCCU does not check..." to "the FOSU does not check..."

- In EOUT interface :
  - Edited the section for style, usage, grammar, clarity, and structure, which included reordering some information and moving some information to the appropriate register descriptions.
  - Changed *Destructive reset* to *Destructive reset (including when an FOSU timeout occurs due to FCCU failing to react)*.
  - Deleted the list item *FOSU timeout reset*.
  - Changed *The FCCU provides up to two bidirectional signals (EOUT interface)* to *The FCCU provides up to two bidirectional signals (EOUT[1:0])*.
  - Changed *These signals need to be mapped to appropriate pins for external visibility* to *For information on the availability and names of these FCCU signals on the boundary of this chip, see the chip-specific FCCU information*.
  - In the list that begins with *These resets influence the state of the failure indication pins*, changed *Pin reset (RESET_b)* to *Chip pin reset (RESET_B)*.
- In the FCCU_CFG register description, edited the following field names and descriptions for clarity:
  - SM
  - PS
  - FOM
- Updated bit description of FCCU_CFG[FCCU_SET_AFTER_RESET], FCCU_CFG[RCCE1] and FCCU_CFG[RCCE0].

- Rewrote the FCCU_EOUT_SIG_EN register description for clarity.
- Added the following note to the FCCU_EOUT_SIG_EN[EOUTENx] field description: *To ensure that the FOSU module is aware that fault-output signaling is enabled as a reaction for the associated noncritical fault channel (so that FOSU doesn't mistakenly generate a destructive chip reset), you must set this field to enabled when FCCU is configured for any mode other than Bistable fault-output mode.*

- In FSM description, in the list item *Configuration*, added the following: *The locking feature only restricts the FSM movement into Configuration state. Once the user enters Configuration state and then tries to lock the configuration by writing either to TRANS_LOCK or PERMT_LOCK, the locking of configuration will be effective after FCCU moves to Normal state, it will not be effective in the current Configuration state.*

- In Definitions
  - Replaced types of reset table with two bullet lists.

- In FSM description, in the list item *Configuration*, deleted *by writing either to TRANS_LOCK or PERMT_LOCK* because these locking mechanisms are discussed elsewhere.

- In Register descriptions, in the first sentence of the second-to-last paragraph, edited text

A.80 Self-Test Control Unit (STCU2) changes

A.80.1 Chip-specific STCU2 information changes

- [Addendum Rev.1] Added sections Off-line LBIST and On-line LBIST.
- Renamed section AUTOLOCK_VALUE for register write access via STCU2_SKC (was "AUTOLOCK_VALUE") and expanded its content.
- Added section Wait time for writing to the online registers.
- Added sections On-Line reset generation (only MBIST) and On-Line reset generation (LBIST enabled).

A.80.2 STCU2 module changes

• Retitled the Built-in self-test scheduling section.
• In Watchdogs, retitled the following sections:
  • Initialization watchdog timer
  • Built-in self-test watchdog timer
  • Register write-access watchdog timer

• Edited Design implementation information for grammar and style. • In Online self-test sequence   • Updated Note "Flash memory must be in the idle..."
• [Addendum Rev.1] In Design implementation information, changed *Do not execute a* to *Do not execute LBIST*.
• In Design implementation information, corrected the grammar in the caution notice.
• In IPS bus interface, replaced the note with a reference to the chip-specific STCU2 information (which is where the correct contents now appear).
• In Reset management, replaced the subtopics pertaining to on-line reset generation with On-Line reset generation, which now states "See the chip-specific STCU2 information for details about On-Line reset generation.
• Editorial changes.
• In Register write-access watchdog timer, added "AUTOLOCK_VALUE is given in the chip-specific STCU2 information."
• In description of setting value 1 for STCU2_ERR_STAT[WDTO], changed "STCU2_CFG[BYP] bit are not set before the Watchdog reach..." to "STCU2_RUN[BYP] bit is not set before the Watchdog reaches..."
• In STCU2 Run Register (STCU2_RUN), in the BYP field description, changed "until the STCU2_SCK register" to "until the STCU2_SKC register".

• STCU2 Error Register (STCU2_ERR_STAT)
  • Updated the field description of UFSF
  • Updated the field description of RFSF

A.81 Register Protection (REG_PROT) changes

A.81.1 Chip-specific REG-PROT information changes

• No substantial content changes

A.81.2 REG_PROT module changes

• In Overview, added AIPS_LITE as an option to use the term PBRIDGE (PBRIDGE/ AIPS_LITE). • In General, modified the sentence "For all addresses that are protected …" to "For all addresses (peripheral registers) that are protected ….". • Removed topic "Initialization/application information".

A.82 Security changes

• Replaced whole chapter with more detailed descriptions on security features.

A.83 Password and Device Security Module (PASS) changes

A.83.1 Chip-specific PASS information changes

- In section Password and Device Security Module (PASS) chip-specific module information :
  - Table Table 82-1, corrected misalignment of table header.
  - Table Table 82-2 moved from section "Flash Memory Read Protection" with modified description in the Censorship column and addition of the Debug Interface column.

A.83.2 PASS module changes

- Revised Challenge Input Register (PASS_CIN*n*) description to indicate that a successful password comparison "clears the PGL bit of that register" (was "sets the PGL bit...").

- In section Features :
  - Added feature 'Flash read, write, and erase control'
- In register description of registers Password Group *n* - Lock 0 Status Register (PASS_LOCK0_PG*n*), Password Group *n* - Lock 1 Status Register (PASS_LOCK1_PG*n*), Password Group *n* - Lock 2 Status Register (PASS_LOCK2_PG*n*), and Password Group *n* - Lock 3 Status Register (PASS_LOCK3_PG*n*) :
  - Changed 'The initial value of the lock ... over during reset.' to 'The initial value of the lock ... over during functional reset.'
- In section DCF clients :
  - Added text to refer to DCF sheet attached with the Reference Manual.

- In section: Memory map and register definition, changed the width of CHSEL[GRP] to 2 (was 3), and updated bit description accordingly

- Updated topic: JTAG Password Challenge

A.84 Tamper Detection Module (TDM) changes

A.84.1 Chip-specific TDM information changes

- Added section OTPEN0 value.

A.84.2 TDM module changes

- Table 83-1 : Changed table title to "DCF clients" (was "DCF client address map").

• Editorial changes.
• Memory Map/Register Definition :
  • Changed title.
  • Removed note from beginning of the section.
• Overview : Updated and edited paragraph beginning with "Security information is passed...".
• Flash erase counter and tamper detection : Updated paragraphs beginning with:
  • "At the end of a successful double word program operation..."
  • "Moreover, if a block is locked by the PASS_LOCK*x*_PG*n* registers..."
• Overview : Changed last sentence from "The block will become OTP after the next reset." to "The block will become OTP after the next functional reset."

• TDR Status Register (TDM_TDRSR) : Removed footnotes from reset values of TDRSRn fields and changed value to 1.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. Freescale reserves the right to make changes without further notice to any products herein.

Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Freescale, the Freescale logo, Qorivva, SafeAssure, and the SafeAssure logo are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. All other product or service names are the property of their respective owners. The Power Architecture and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2012–2016 Freescale Semiconductor, Inc.

Document Number MPC5746RRM
Revision 6, 03/2016



MPC5746R Emulation and Debug Device Reference Manual

Document Number: MPC5746RBDRM
Rev. 6, 03/2016

Contents

Section number **Title** **Page**

Chapter 1
Emulation and Debug Device Introduction

1.1 Introduction....9

1.2 Functionality....9

1.3 Feature summary....9

1.4 Block diagram....10

1.5 Clocking....11

1.6 Reset logic....14

1.7 BD memory map....15

Chapter 2
Buddy Device—Crossbar and RAM (BD_XBS_RAM)

2.1 Introduction....17

2.2 3 x 5 Crossbar switch and attached controllers....18

2.3 BD_XBS_RAM End-to-End ECC32 Checkbit/Syndrome Coding Scheme....19

Chapter 3
Buddy Device—System Integration Unit Lite (BD_SIUL2)

3.1 Introduction....25

3.2 Device identification....26

3.3 Device status information....26

3.3.1 Initialization status....26

3.3.2 PD reset status....27

3.4 Memory mapped registers....27

3.4.1 Device Identification Register 1 (BD_SIUL2_DIDR1)....28

3.4.2 Device Identification Register 2 (BD_SIUL2_DIDR2)....29

3.4.3 Initialization Status Register (BD_SIUL2_ISR)....30

3.4.4 Reset Status Register (BD_SIUL2_RSR)....31

Chapter 4
Buddy Device—Debug and Calibration Interface (DCI)

4.1 Introduction....33
4.1.1 Features....33
4.1.2 Overview....34
4.1.3 Operating modes....36
4.2 External signal description....39
4.3 Register description....40
4.3.1 DCI control register (BD_DCI_CR)....40
4.3.2 DCI EVTx pin multiplexing control register (DCI_PINCR)....41

Chapter 5
Buddy Device—JTAG Controller (JTAGC)

5.1 Introduction....43
5.1.1 Block diagram....43
5.1.2 Features....44
5.1.3 Modes of operation....44
5.2 External signal description....45
5.2.1 TCK—Test clock input....46
5.2.2 TDI—Test data input....46
5.2.3 TDO—Test data output....46
5.2.4 TMS—Test mode select....46
5.2.5 JCOMP—JTAG compliancy....46
5.3 Register description....47
5.3.1 Instruction register....47
5.3.2 Bypass register....47
5.3.3 Device identification register....47
5.3.4 JTAG_PASSWORD register....48
5.3.5 Boundary scan register....49

5.4 Functional description....49
5.4.1 JTAGC reset configuration....49
5.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port....50
5.4.3 TAP controller state machine....50
5.4.4 JTAGC block instructions....52
5.4.5 Boundary scan....55
5.5 Initialization/Application information....56

Chapter 6
Buddy Device—Nexus Aurora Router (NAR)

6.1 Introduction....57
6.2 Overview....58
6.3 Features....58
6.4 Register definition....59
6.4.1 NAR control register (NAR_CR)....60
6.4.2 NAR status register (NAR_ST)....62
6.4.3 Message filtering registers....64
6.4.4 Trace memory bus configuration registers....67
6.4.5 Suppress mode triggers registers (NAR_STCR)....70
6.4.6 Client suppress status register (NAR_CSSR)....71
6.4.7 Receive queue client disable register (NAR_CDR)....73
6.4.8 Nexus Aurora Phy control register (NAR_NAPCR)....75
6.4.9 Trace Memory fill level and partition configuration register (NAR_AHFPAR)....76
6.5 Functional description....77
6.5.1 Reset/Startup....77
6.5.2 Operation modes....78
6.5.3 Data reception and arbitration....81
6.5.4 Suppress mode....83
6.5.5 Stall detection....84
6.5.6 Output arbitration....85

6.5.7 Queue partitioning....86
6.5.8 Message translation and formatting....87
6.5.9 Aurora interface....88
6.5.10 Trace memory bus output port usage....89
6.5.11 Internal message generation....91
6.5.12 Timestamp function....95
6.5.13 Global (Development tool client) stall control....96
6.5.14 Overlay and internal trace memory full status generation....96
6.5.15 Configuration/debug interface JTAG....97
6.5.16 Trace memory bus controller....101
6.5.17 Aurora client controller....103
6.6 Application information....104

Chapter 7
Buddy Device—Nexus Aurora Link (NAL)

7.1 Introduction....107
7.2 Transmit operation....108
7.3 Nexus Aurora formatting....109
7.4 Static training....109
7.5 Clock compensation considerations....110
7.6 Programmable registers....111
7.6.1 Nexus Aurora Link General Status Register (NAL_GSR)....112
7.6.2 NAL General Control Register (NAL_GCR)....113
7.6.3 NAL Training Control Register (NAL_TCR)....115

Chapter 8
Buddy Device—Nexus Aurora PHY (NAP)

8.1 Introduction....119
8.1.1 Features....120
8.1.2 Modes of operation....120
8.2 External signal description....121

8.3 Memory map and register definition....122
8.4 Functional description....122
8.4.1 Data symbol character accumulator....122
8.4.2 Serializer....122
8.4.3 Clock generation and control....123
8.5 Initialization information....123

Chapter 9
Buddy Device—Nexus Read Write Access (RWA)

9.1 Introduction....125
9.2 Register definition....125
9.2.1 Read/write access control/status (RWCS)....126
9.2.2 Read/write access data (RWD)....128
9.2.3 Read/write access address (RWA)....129
9.3 Nexus register access via JTAG....129
9.4 Memory access examples....130
9.4.1 Single write access....130
9.4.2 Block write access....131
9.4.3 Single read access....132
9.4.4 Block read access....132

Chapter 1
Emulation and Debug Device Introduction

1.1 Introduction

The emulation and debug device is a multi-die MCU that adds additional functionality specifically for ease in calibration and debug. The Emulation and debug device (ED) consists of the production die (PD) and a buddy die (BD). The BD for this chip does not implement the JTAG Master (JTAGM) functionality; disregard any references to JTAGM in the BD chapters.

1.2 Functionality

The ED is a PD packaged with a BD to provide additional capability during development. The ED is used for development, debug, and calibration of automotive powertrain controller applications. To enable these operations, the buddy device supports the following functions:

- Debug and calibration tool interface—Allows tool read/write access to buddy device resources even if main MCU supplies are removed as long as special ED debug supplies are maintained. The BD duplicates the Debug and Calibration Interface (DCI) functionality that is also available in the PD.
- High speed tool trace port (based on the IEEE-ISTO 5001-2012 Nexus Aurora standard)
- 1 MB on-chip SRAM usable as calibration overlay or trace RAM—entire SRAM contents are maintained as long as ED debug supply is maintained

1.3 Feature summary

On-chip modules available on the BD include the following features:

- 1 MB on-chip SRAM
  - Access timing matches internal PD flash accesses when used in conjunction with PD calibration remap logic
  - Includes ECC
  - Implemented as 4 × 256 KB partitions where each partition can be accessed by different masters without causing blockages to access of any other partition
- XBAR switch architecture
- XBAR slave interface
  - 64-bit data width with ECC checkbits
  - Supports "Early Write" AHB protocol (AHB-EW) for one clock trace data writes
  - Allows a PD with AHB master to support read and write to buddy device memory-mapped resources—Memory-mapped resources include buddy device SRAM and AIPS resources
- Nexus Aurora auxiliary trace port
  - 1.25 Gbit/s maximum data rate per lane
  - Four lanes supported in ED package
  - Port configurable to operate in 2- or 4-lane width
  - One lane LVDS reference clock in
  - Supports Nexus development interface (NDI) per IEEE-ISTO 5001-2003 standard, with some support for 2012 standard.
- Debug and Calibration Interface—Supports JTAG (IEEE 1149.1 and IEEE 1149.7) mode
- Independent high and low voltage supplies allow tool control of BD debug and calibration resources even when the main MCU supplies are off
- Device and board test support per Joint Test Action Group (JTAG) (IEEE 1149.1)

1.4 Block diagram

This figure shows the top-level block diagram of the emulation and debug device.

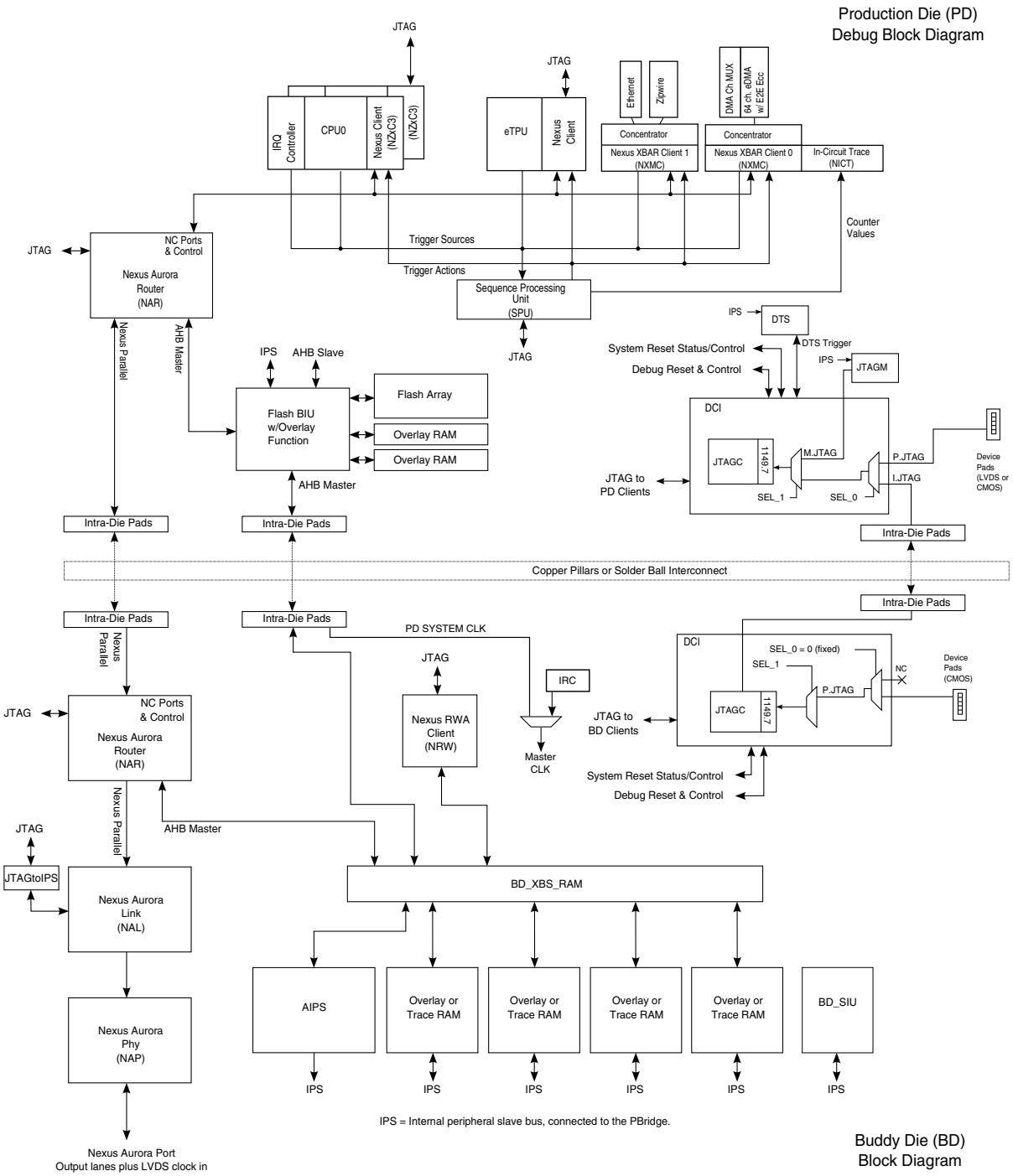


Figure 1-1. Emulation and debug device architecture

1.5 Clocking

The figure below shows the clocking structure of the BD. There is one internal clock source and four external clock sources:

- Internal RC oscillator (IRC) clock
- System clock coming from the PD (identified as BD_CLK clock in the clock generation figure of the Clocking chapter)
- JTAG clock coming from the TCK pin in the ED
- Nexus Aurora PHY clock coming from the differential LVDS pins in the ED

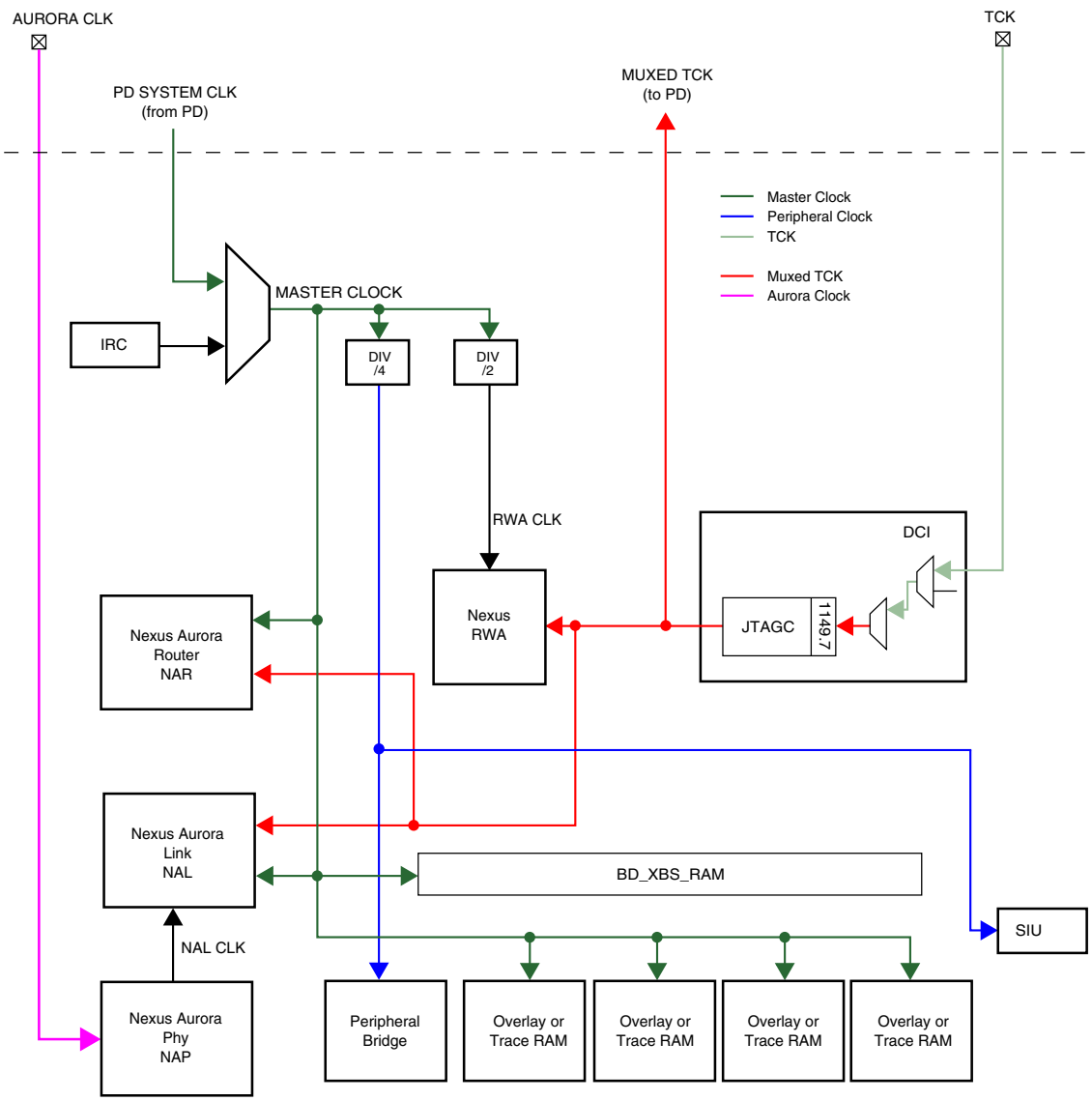


Figure 1-2. BD clock structure

The IRC is the source of the BD master clock when the PD is unpowered, or powered but still in reset state. It has a nominal frequency of 64 MHz, does not have any user accessible registers and is always enabled. It does not have any trimming capability, therefore the expected precision is on the order of ±30%.

The PD system clock is the source of the BD master clock when the PD is powered and out of reset state. When the PD PLL is engaged and locked this clock has a nominal frequency of 200 MHz, but while the PLL is not engaged the frequency is that of the PD internal RC oscillator.

The master clock directly feeds the XBAR, SRAMs and the Aurora sub-system. It is also divided down to generate the following internal clocks:

- Peripheral clock (divided by 4)—provides the clock to memory mapped registers and other low-speed logic
- Nexus RWA clock (divided by 2)—provides the clock to the Nexus Read Write Access module

The internal TCK clock (referred to as muxed clock in Figure 1-2) is used in the Nexus RWA, NAR, NAL, and it also goes to the PD to be used as the JTAG clock. (In the ED, the JTAG package pins are bonded to the BD JTAG interface; therefore, the PD receives the JTAG signals from the BD.) This clock comes from the TCK package pin, when using IEEE 1149.1 or IEEE 1149.7 access.

When the PD is unpowered, JTAG accesses can be done only by IEEE 1149.1 or IEEE 1149.7 modes accessing through the Nexus RWA module. In this mode, the TCK clock comes from the package pin and the RWA clock is a divided-by-2 version of the IRC; therefore, these clocks are asynchronous to each other. When clocks at the input of the RWA are asynchronous, there is a restriction that the RWA clock must be at least double the frequency of the JTAG clock. This means that the master clock frequency must be at least four times the JTAG clock frequency.

This table summarizes the frequency restrictions on the Nexus RWA accesses.

Table 1-1. JTAG operating frequencies

PD configuration	BD configuration	PD system clock frequency [MHz]	BD master clock frequency [MHz]	Max RWA JTAG frequency [MHz]	Notes
Unpowered or in reset state	1149.1 or 1149.7	—	44.8	11.2	Assuming 64 MHz BD IRC with 30% tolerance
Powered, out of reset, PLL not engaged, system clock at IRC frequency	1149.1 or 1149.7	15.52	15.52	3.88	Assuming 16 MHz PD IRC with 3% tolerance
Powered, out of reset, PLL engaged	1149.1 or 1149.7	200	200	27.7/50	JTAG speed is limited by TCK negedge to TDO delay (14 ns) plus external setup to TCK posedge (4 ns). Otherwise, the maximum Nexus RWA JTAG frequency is internally limited to 50 MHz.

1.6 Reset logic

There are five sources of reset that are combined to generate the reset of system portions as described in the following table:

1. BD power-on reset—asserts immediately upon assertion of BD POR, and remains asserted for 10 clock cycles after negation of POR.
2. PD reset—asserts two clock rising edges after assertion of reset signal coming from PD and remains asserted while the PD reset signal is asserted plus two clock rising edges after the PD reset signal is de-asserted (this reset source is reflected in the PD_RST bit of the BD_SIU_RSR register).
3. DCI FBD reset—asserts immediately when the FBD_RESET bit of the DCI Control Register is set, and remains asserted for one clock cycle after FBD_RESET is negated.
4. JTAG controller reset—asserts immediately when a reset request signal is received from the JTAG Controller by the execution of EXTEST, CLAMP, HIZ boundary instructions, and remains asserted for one clock cycle after the reset request is negated.
5. Nexus reset—asserts by DCI logic based on JCOMP state and JTAG source selected. DCI FBD also generates a Nexus reset.

In the following table, an 'X' indicates that the reset source generates a reset signal to the specified block. JTAG mode only indicates that those blocks are reset by the corresponding reset sources only in the case where the debug interface is running through the JTAG external pins.

Table 1-2. Reset sources and functional blocks

Blocks	BD POR	PD reset	DCI FBD reset	JTAGC reset	Nexus reset
DCI	X	—	—	—	—
Z0 Nexus RWA	X	—	X	—	X
BD XBAR	X	—	X	X	—
Overlay/Trace RAM	X	—	X	X	—
NAR	X	—	X	X	X
NAL	X	—	X	X	X
NAP	X	—	X	X	X
BD SIU	X	—	X	X	—
BD SIU GPIO registers	X	X	—	X	—

Table continues on the next page...

Table 1-2. Reset sources and functional blocks (continued)

Blocks	BD POR	PD reset	DCI FBD reset	JTAGC reset	Nexus reset
BD INIT flag	X	—	X	X	—
PD/BD interface logic	X	—	X	X	—

NOTE

The PD reset is not affected by the BD reset. The BD does not interfere with startup or shut down timing of the PD.

1.7 BD memory map

The BD memory map is contained within a 16 MB range. This entire range is accessed through the PD Flash BIU. The BD memory map is divided into two regions: overlay RAM and device registers.

The device register region allows the CPUs on the PD to access configuration registers of the peripherals in the BD, such as XBAR configuration, etc. The device register region are divided into 16 KB regions for assignment to each peripheral block.

The memory map for the BD is shown below as an absolute address relative to the PD address map. Within the BD, only a 24-bit address bus is required. The upper 8 bits of the complete 32-bit address are decoded and removed by the PD PFLASH controller.

Table 1-3. BD memory map

Start address	End address	Allocated size (KB)	Description
BD overlay RAM			
0x0C00_0000	0x0C0F_FFFF	1024	Extended overlay RAM
0x0C10_0000	0x0C1F_FFFF	1024	Reserved BD overlay RAM
0x0C20_0000	0x0C3F_FFFF	2048	Reserved BD overlay RAM
0x0C40_0000	0x0C5F_FFFF	2048	Reserved BD overlay RAM
0x0C60_0000	0x0C7F_FFFF	2048	Reserved BD overlay RAM
BD registers			
0x0C80_0000	0x0C80_3FFF	16	BD_SIUL2
0x0C80_4000	0x0C80_7FFF	16	Reserved
0x0C80_8000	0x0C80_BFFF	16	Reserved
0x0C80_C000	0x0C80_FFFF	16	Reserved
0x0C81_0000	0x0C81_3FFF	16	Reserved
0x0C81_4000	0x0C81_7FFF	16	Reserved
0x0C81_8000	0x0C81_BFFF	16	Reserved

Table continues on the next page...

Table 1-3. BD memory map (continued)

Start address	End address	Allocated size (KB)	Description
0x0C81_C000	0x0C81_FFFF	16	Reserved
0x0C82_0000	0x0CFF_FFFF	8064	Reserved

Chapter 2
Buddy Device—Crossbar and RAM (BD_XBS_RAM)

2.1 Introduction

Within the buddy die (BD), there is a crossbar switch that provides the required connections between multiple bus masters and the overlay RAM. For a diagram of the Emulation device (ED) debug architecture that shows the BD blocks and crossbar, see the Emulation device introduction chapter. This section provides details on the structure of this function.

Note

Throughout this description, the term "overlay RAM" is applied to the BD memory, even though this memory can be used for *both* overlay and debug trace information.

The 64-bit BD crossbar and RAM structure consists of:

- Input registers to capture transfers (address, attributes and write data) targeted at the overlay RAM or memory-mapped peripheral space
- A 3 x 5 crossbar switch to route requests from masters to either the overlay RAM or the system bus
    - Bus masters include the BD's Nexus Aurora Router (BD-NAR), the production device's flash controller (PFLASH) and the BD's Nexus Read/Write Access Client (RWA)
    - Bus slaves include 4 equal-sized regions of the overlay RAM and a peripheral bridge (AIPS-Lite) bus controller
        - 160-bit RAM arrays (128 bits of data, 32 bits of end-to-end ECC)
        - End-to-End ECC32 address and data protection

    - ECC scheme based on 29-bit address and 32-bit data
    - Overlay RAM supports a 64-bit "early write" datapath and a 128-bit read datapath
- Output alignment logic and registers to capture read data being driven to the requesting master

The result is an efficient structure supporting concurrent references of the overlay RAM for calibration accesses from the PFLASH controller on the Production Device (PD) and debug references from the other masters. The crossbar and RAM support 200 MHz operation.

Note

Calibration reads from the PFLASH have the same timing characteristics as normal data reads from the flash array.

2.2 3 x 5 Crossbar switch and attached controllers

In the same manner as the comparable crossbar switches in the PD, the BD crossbar concurrently supports up to 3 simultaneous connections between the master and the slave ports. It implements a 64-bit "early write" datapath and a 128-bit read datapath and fully supports concurrent accesses between calibration reads and debug trace writes to different overlay RAM regions.

This crossbar implements fixed priority arbitration on a per slave port basis where the master number defines the relative priority in a reverse numeric order, that is, m0 > m1 > m2 (PFLASH > BD-NAR > Nexus RWA). Each slave port is "parked" on the last master making a request.

The design supports {8, 16, 32, 64}-bit SINGLE reads, {8, 16, 32, 64}-bit SINGLE writes along with 64-bit WRAP4 burst reads and writes. Two consecutive sequential 128-bit accesses of the RAM are used (only) on the WRAP4 burst read.

2.3 BD_XBS_RAM End-to-End ECC32 Checkbit/Syndrome Coding Scheme

The e2eECC scheme used in the BD overlay RAM is a simplified derivative of the algorithm used in the Production Device. The only difference is the code is based on a 32-bit data field, but the same H-matrix is used with the appropriate 32-bit data upper or lower word zeroed.

Consider the following Boolean equation definitions:

e2eECC64[7:0] = ecc_addr[7:0] ^ ecc_data_upper[7:0] ^ ecc_data_lower[7:0]

where `ecc_addr` is H-matrix evaluation of the 29-bit access address (addr[31:3]), `ecc_data_upper` is the H-matrix evaluation of the upper data word (data[63:32]) and `ecc_data_lower` is the comparable evaluation of the lower data word (data[31:0]).

For the BD RAM definition, the following (re)definitions apply:

e2eECC32_upper[7:0] = ecc_addr[7:0] ^ ecc_data_upper[7:0]

e2eECC32_lower[7:0] = ecc_addr[7:0] ^ ecc_data_lower[7:0]

As a result, the PD's H-matrix is used for the BD RAM's e2eECC32 scheme with the appropriate 32 bit data word zeroed out:

e2eECC32_upper[7:0] = ecc_addr[7:0] ^ ecc_data_upper[7:0]; data_lower[31:00] = 0

e2eECC32_lower[7:0] = ecc_addr[7:0] ^ ecc_data_lower[7:0]; data_upper[63:32] = 0

In any case, the resulting ECC32 is calculated based on the appropriate 32 data bits plus 29 address bits (the upper bits of the 32-bit address field).

The complete H-matrix for this (101, 93) code is shown in the following table. A '*' in the table indicates the corresponding data or address bit is XOR'd to form the final checkbit value on the left. For 32-bit data writes, the table sections corresponding to D[63:32] or D[31:0], together with A[31:3] are logically summed (output of each table section is XOR'd) together to the final value driven on the hwchkbit[7:0] outputs. Note that this table uses *the AHB bit numbering convention where bit[0] is the least significant bit*.

Table 2-1. e2eECC Basic H-Matrix Definition

Checkbits [7:0]	Data Bit[1]																															
	Byte 7								Byte 6								Byte 5								Byte 4							
	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
7	*				*					*			*		*	*		*	*	*	*					*	*			*	*	

Table continues on the next page...

Table 2-1. e2eECC Basic H-Matrix Definition (continued)

6			*	*			*	*					*		*		*				*				*			*	*			*
5	*	*	*	*					*				*	*		*		*	*				*	*	*				*			
4	*		*			*		*					*		*		*			*			*	*			*				*	
3									*	*	*	*	*		*				*		*				*	*	*	*				
2						*		*					*		*			*	*				*	*							*	
1			*		*		*		*	*	*	*	*						*		*			*							*	
0			*			*	*	*				*			*	*	*			*				*				*			*	
			Byte 3								**Byte 2**									**Byte 1**									**Byte 0**			
Checkbits [7:0]	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
7					*		*				*		*				*	*			*	*			*				*	*		*
6	*	*		*	*				*				*				*	*		*	*					*	*				*	*
5			*	*			*	*	*			*	*	*					*		*	*		*								
4	*		*	*				*		*	*				*	*	*			*			*	*				*		*		
3	*	*	*			*						*	*		*	*			*				*		*	*	*			*		
2		*		*	*	*	*	*	*	*	*			*					*				*		*		*	*		*		
1		*						*		*			*	*	*			*				*				*			*	*	*	
0		*		*		*		*				*				*				*				*				*	*		*	*
	Address Bit																															
Checkbits [7:0]	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**			
7		*		*		*		*	*	*		*	*	*		*	*	*		*		*		*	*	*		*				
6		*		*	*	*	*			*	*			*	*	*	*		*		*	*		*	*	*		*				
5		*	*	*			*	*			*	*		*		*		*	*			*	*	*		*	*	*	*			
4	*	*	*		*	*		*		*	*	*	*		*					*	*		*	*	*		*	*				
3	*		*		*	*	*		*		*	*	*			*		*		*	*		*		*	*	*		*				
2	*	*			*		*	*	*	*				*	*	*	*	*	*	*		*	*		*		*		*				
1	*		*	*		*	*		*	*		*	*		*		*	*	*		*	*	*					*	*				
0	*		*	*	*			*	*		*		*	*	*		*		*	*	*			*		*	*		*				

1. Bit numbering is AHB convention, bit 0 is LSB. D[7:0] corresponds to byte at address 0. D[63:56] corresponds to byte at address 7.

Figure 2-1 through Figure 2-4 show an alternative representation of the ECC32 encode process, written as a C language function.

```
encodeEcc32_upper (addr, data_a2_is_zero)
    unsigned int        addr;                   /* 32-bit byte address */
    unsigned int        data_a2_is_zero;        /* 32-bit data upper, a[2]=0 */

{
    unsigned int        addr_ecc;               /* 8 bits of ecc for address */
    unsigned int        ecc;                    /* 8 bits of ecc codeword */

    /* the following equation calculates the 8-bit wide ecc codeword by examining
    each addr or data bits and xor'ing the appropriate H-matrix value if the bit = 1 */

    addr_ecc
        = (((addr           >> 31) & 1) ? 0x1f : 0x0)      /* addr[31] */
        ^ (((addr           >> 30) & 1) ? 0xf4 : 0x0)      /* addr[30] */
        ^ (((addr           >> 29) & 1) ? 0x3b : 0x0)      /* addr[29] */
        ^ (((addr           >> 28) & 1) ? 0xe3 : 0x0)      /* addr[28] */
        ^ (((addr           >> 27) & 1) ? 0x5d : 0x0)      /* addr[27] */
        ^ (((addr           >> 26) & 1) ? 0xda : 0x0)      /* addr[26] */
        ^ (((addr           >> 25) & 1) ? 0x6e : 0x0)      /* addr[25] */
        ^ (((addr           >> 24) & 1) ? 0xb5 : 0x0)      /* addr[24] */

        ^ (((addr           >> 23) & 1) ? 0x8f : 0x0)      /* addr[23] */
        ^ (((addr           >> 22) & 1) ? 0xd6 : 0x0)      /* addr[22] */
        ^ (((addr           >> 21) & 1) ? 0x79 : 0x0)      /* addr[21] */
        ^ (((addr           >> 20) & 1) ? 0xba : 0x0)      /* addr[20] */
        ^ (((addr           >> 19) & 1) ? 0x9b : 0x0)      /* addr[19] */
        ^ (((addr           >> 18) & 1) ? 0xe5 : 0x0)      /* addr[18] */
        ^ (((addr           >> 17) & 1) ? 0x57 : 0x0)      /* addr[17] */
        ^ (((addr           >> 16) & 1) ? 0xec : 0x0)      /* addr[16] */

        ^ (((addr           >> 15) & 1) ? 0xc7 : 0x0)      /* addr[15] */
        ^ (((addr           >> 14) & 1) ? 0xae : 0x0)      /* addr[14] */
        ^ (((addr           >> 13) & 1) ? 0x67 : 0x0)      /* addr[13] */
        ^ (((addr           >> 12) & 1) ? 0x9d : 0x0)      /* addr[12] */
        ^ (((addr           >> 11) & 1) ? 0x5b : 0x0)      /* addr[11] */
        ^ (((addr           >> 10) & 1) ? 0xe6 : 0x0)      /* addr[10] */
        ^ (((addr           >>  9) & 1) ? 0x3e : 0x0)      /* addr[ 9] */
        ^ (((addr           >>  8) & 1) ? 0xf1 : 0x0)      /* addr[ 8] */

        ^ (((addr           >>  7) & 1) ? 0xdc : 0x0)      /* addr[ 7] */
        ^ (((addr           >>  6) & 1) ? 0xe9 : 0x0)      /* addr[ 6] */
        ^ (((addr           >>  5) & 1) ? 0x3d : 0x0)      /* addr[ 5] */
        ^ (((addr           >>  4) & 1) ? 0xf2 : 0x0)      /* addr[ 4] */
        ^ (((addr           >>  3) & 1) ? 0x2f : 0x0);     /* addr[ 3] */
```

Figure 2-1. C language encodeECC32_upper function description (Part 1 of 4)

```
 ecc = (((data_a2_is_zero >> 31) & 1) ? 0xb0 : 0x0)      /* data[63] */
        ^ (((data_a2_is_zero >> 30) & 1) ? 0x23 : 0x0)      /* data[62] */
        ^ (((data_a2_is_zero >> 29) & 1) ? 0x70 : 0x0)      /* data[61] */
        ^ (((data_a2_is_zero >> 28) & 1) ? 0x62 : 0x0)      /* data[60] */
        ^ (((data_a2_is_zero >> 27) & 1) ? 0x85 : 0x0)      /* data[59] */
        ^ (((data_a2_is_zero >> 26) & 1) ? 0x13 : 0x0)      /* data[58] */
        ^ (((data_a2_is_zero >> 25) & 1) ? 0x45 : 0x0)      /* data[57] */
        ^ (((data_a2_is_zero >> 24) & 1) ? 0x52 : 0x0)      /* data[56] */

        ^ (((data_a2_is_zero >> 23) & 1) ? 0x2a : 0x0)      /* data[55] */
        ^ (((data_a2_is_zero >> 22) & 1) ? 0x8a : 0x0)      /* data[54] */
        ^ (((data_a2_is_zero >> 21) & 1) ? 0x0b : 0x0)      /* data[53] */
        ^ (((data_a2_is_zero >> 20) & 1) ? 0x0e : 0x0)      /* data[52] */
        ^ (((data_a2_is_zero >> 19) & 1) ? 0xf8 : 0x0)      /* data[51] */
        ^ (((data_a2_is_zero >> 18) & 1) ? 0x25 : 0x0)      /* data[50] */
        ^ (((data_a2_is_zero >> 17) & 1) ? 0xd9 : 0x0)      /* data[49] */
        ^ (((data_a2_is_zero >> 16) & 1) ? 0xa1 : 0x0)      /* data[48] */

        ^ (((data_a2_is_zero >> 15) & 1) ? 0x54 : 0x0)      /* data[47] */
        ^ (((data_a2_is_zero >> 14) & 1) ? 0xa7 : 0x0)      /* data[46] */
        ^ (((data_a2_is_zero >> 13) & 1) ? 0xa8 : 0x0)      /* data[45] */
        ^ (((data_a2_is_zero >> 12) & 1) ? 0x92 : 0x0)      /* data[44] */
        ^ (((data_a2_is_zero >> 11) & 1) ? 0xc8 : 0x0)      /* data[43] */
        ^ (((data_a2_is_zero >> 10) & 1) ? 0x07 : 0x0)      /* data[42] */
        ^ (((data_a2_is_zero >>  9) & 1) ? 0x34 : 0x0)      /* data[41] */
        ^ (((data_a2_is_zero >>  8) & 1) ? 0x32 : 0x0)      /* data[40] */

        ^ (((data_a2_is_zero >>  7) & 1) ? 0x68 : 0x0)      /* data[39] */
        ^ (((data_a2_is_zero >>  6) & 1) ? 0x89 : 0x0)      /* data[38] */
        ^ (((data_a2_is_zero >>  5) & 1) ? 0x98 : 0x0)      /* data[37] */
        ^ (((data_a2_is_zero >>  4) & 1) ? 0x49 : 0x0)      /* data[36] */
        ^ (((data_a2_is_zero >>  3) & 1) ? 0x61 : 0x0)      /* data[35] */
        ^ (((data_a2_is_zero >>  2) & 1) ? 0x86 : 0x0)      /* data[34] */
        ^ (((data_a2_is_zero >>  1) & 1) ? 0x91 : 0x0)      /* data[33] */
        ^  ((data_a2_is_zero        & 1) ? 0x46 : 0x0);     /* data[32] */

return(ecc);
}
```

Figure 2-2. C language encodeECC32_upper function description (Part 2 of 4)

```
C Language encodeECC32_lower Function Description
encodeEcc32_lower (addr, data_a2_is_one)
    unsigned int        addr;                   /* 32-bit byte address */
    unsigned int        data_a2_is_one;         /* 32-bit data lower, a[2]=1 */

{
    unsigned int        addr_ecc;               /* 8 bits of ecc for address */
    unsigned int        ecc;                    /* 8 bits of ecc codeword */

    /* the following equation calculates the 8-bit wide ecc codeword by examining
    each addr or data bits and xor'ing the appropriate H-matrix value if the bit = 1 */

    addr_ecc
        = (((addr           >> 31) & 1) ? 0x1f : 0x0)      /* addr[31] */
        ^ (((addr           >> 30) & 1) ? 0xf4 : 0x0)      /* addr[30] */
        ^ (((addr           >> 29) & 1) ? 0x3b : 0x0)      /* addr[29] */
        ^ (((addr           >> 28) & 1) ? 0xe3 : 0x0)      /* addr[28] */
        ^ (((addr           >> 27) & 1) ? 0x5d : 0x0)      /* addr[27] */
        ^ (((addr           >> 26) & 1) ? 0xda : 0x0)      /* addr[26] */
        ^ (((addr           >> 25) & 1) ? 0x6e : 0x0)      /* addr[25] */
        ^ (((addr           >> 24) & 1) ? 0xb5 : 0x0)      /* addr[24] */

        ^ (((addr           >> 23) & 1) ? 0x8f : 0x0)      /* addr[23] */
        ^ (((addr           >> 22) & 1) ? 0xd6 : 0x0)      /* addr[22] */
        ^ (((addr           >> 21) & 1) ? 0x79 : 0x0)      /* addr[21] */
        ^ (((addr           >> 20) & 1) ? 0xba : 0x0)      /* addr[20] */
        ^ (((addr           >> 19) & 1) ? 0x9b : 0x0)      /* addr[19] */
        ^ (((addr           >> 18) & 1) ? 0xe5 : 0x0)      /* addr[18] */
        ^ (((addr           >> 17) & 1) ? 0x57 : 0x0)      /* addr[17] */
        ^ (((addr           >> 16) & 1) ? 0xec : 0x0)      /* addr[16] */

        ^ (((addr           >> 15) & 1) ? 0xc7 : 0x0)      /* addr[15] */
        ^ (((addr           >> 14) & 1) ? 0xae : 0x0)      /* addr[14] */
        ^ (((addr           >> 13) & 1) ? 0x67 : 0x0)      /* addr[13] */
        ^ (((addr           >> 12) & 1) ? 0x9d : 0x0)      /* addr[12] */
        ^ (((addr           >> 11) & 1) ? 0x5b : 0x0)      /* addr[11] */
        ^ (((addr           >> 10) & 1) ? 0xe6 : 0x0)      /* addr[10] */
        ^ (((addr           >>  9) & 1) ? 0x3e : 0x0)      /* addr[ 9] */
        ^ (((addr           >>  8) & 1) ? 0xf1 : 0x0)      /* addr[ 8] */

        ^ (((addr           >>  7) & 1) ? 0xdc : 0x0)      /* addr[ 7] */
        ^ (((addr           >>  6) & 1) ? 0xe9 : 0x0)      /* addr[ 6] */
        ^ (((addr           >>  5) & 1) ? 0x3d : 0x0)      /* addr[ 5] */
        ^ (((addr           >>  4) & 1) ? 0xf2 : 0x0)      /* addr[ 4] */
        ^ (((addr           >>  3) & 1) ? 0x2f : 0x0);     /* addr[ 3] */
```

Figure 2-3. C language encodeECC32_upper function description (Part 3 of 4)

```
        ecc = (((data_a2_is_one  >> 31) & 1) ? 0x58 : 0x0)      /* data[31] */
            ^ (((data_a2_is_one  >> 30) & 1) ? 0x4f : 0x0)      /* data[30] */
            ^ (((data_a2_is_one  >> 29) & 1) ? 0x38 : 0x0)      /* data[29] */
            ^ (((data_a2_is_one  >> 28) & 1) ? 0x75 : 0x0)      /* data[28] */
            ^ (((data_a2_is_one  >> 27) & 1) ? 0xc4 : 0x0)      /* data[27] */
            ^ (((data_a2_is_one  >> 26) & 1) ? 0x0d : 0x0)      /* data[26] */
            ^ (((data_a2_is_one  >> 25) & 1) ? 0xa4 : 0x0)      /* data[25] */
            ^ (((data_a2_is_one  >> 24) & 1) ? 0x37 : 0x0)      /* data[24] */

            ^ (((data_a2_is_one  >> 23) & 1) ? 0x64 : 0x0)      /* data[23] */
            ^ (((data_a2_is_one  >> 22) & 1) ? 0x16 : 0x0)      /* data[22] */
            ^ (((data_a2_is_one  >> 21) & 1) ? 0x94 : 0x0)      /* data[21] */
            ^ (((data_a2_is_one  >> 20) & 1) ? 0x29 : 0x0)      /* data[20] */
            ^ (((data_a2_is_one  >> 19) & 1) ? 0xea : 0x0)      /* data[19] */
            ^ (((data_a2_is_one  >> 18) & 1) ? 0x26 : 0x0)      /* data[18] */
            ^ (((data_a2_is_one  >> 17) & 1) ? 0x1a : 0x0)      /* data[17] */
            ^ (((data_a2_is_one  >> 16) & 1) ? 0x19 : 0x0)      /* data[16] */

            ^ (((data_a2_is_one  >> 15) & 1) ? 0xd0 : 0x0)      /* data[15] */
            ^ (((data_a2_is_one  >> 14) & 1) ? 0xc2 : 0x0)      /* data[14] */
            ^ (((data_a2_is_one  >> 13) & 1) ? 0x2c : 0x0)      /* data[13] */
            ^ (((data_a2_is_one  >> 12) & 1) ? 0x51 : 0x0)      /* data[12] */
            ^ (((data_a2_is_one  >> 11) & 1) ? 0xe0 : 0x0)      /* data[11] */
            ^ (((data_a2_is_one  >> 10) & 1) ? 0xa2 : 0x0)      /* data[10] */
            ^ (((data_a2_is_one  >>  9) & 1) ? 0x1c : 0x0)      /* data[ 9] */
            ^ (((data_a2_is_one  >>  8) & 1) ? 0x31 : 0x0)      /* data[ 8] */

            ^ (((data_a2_is_one  >>  7) & 1) ? 0x8c : 0x0)      /* data[ 7] */
            ^ (((data_a2_is_one  >>  6) & 1) ? 0x4a : 0x0)      /* data[ 6] */
            ^ (((data_a2_is_one  >>  5) & 1) ? 0x4c : 0x0)      /* data[ 5] */
            ^ (((data_a2_is_one  >>  4) & 1) ? 0x15 : 0x0)      /* data[ 4] */
            ^ (((data_a2_is_one  >>  3) & 1) ? 0x83 : 0x0)      /* data[ 3] */
            ^ (((data_a2_is_one  >>  2) & 1) ? 0x9e : 0x0)      /* data[ 2] */
            ^ (((data_a2_is_one  >>  1) & 1) ? 0x43 : 0x0)      /* data[ 1] */
            ^  ((data_a2_is_one         & 1) ? 0xc1 : 0x0);     /* data[ 0] */

    return(ecc);
    }
```

Figure 2-4. C language encodeECC32_upper function description (Part 4 of 4)

As the ECC syndrome is calculated on a read operation by applying the H-matrix to the data plus the checkbits, an all zero syndrome indicates an error free operation. If the generated syndrome value is non-zero and matches one of the H-matrix values associated with the data or checkbits, it represents a single-bit error correction case and the specific bit is complemented to produce the correct data value. If the syndrome value matches one of the H-matrix values associated with the address bits, the opposite data word, or is an even weight value, or represents an unused odd weight value, a non-correctable ECC event has been detected and the appropriate error termination response is initiated.

Chapter 3
Buddy Device—System Integration Unit Lite (BD_SIUL2)

3.1 Introduction

The Buddy Device—System Integration Unit Lite2 (BD_SIUL2) provides control over the electrical pads of the buddy device (BD), as well as implementing various SoC functions that are not included in other modules.

The main functions supported by BD_SIUL2 are:

- Device identification
- Device status reporting
- Pad control and signal muxing

It should be noted that some other system level features are located in the BD DCI block including:

- Device identification available via the JTAG accessible Nexus Device ID (DID) register
- Reporting of reset status via the JTAG accessible reset status register
- Control of production device (PD) and BD reset via JTAG accessible reset control register

3.2 Device identification

The BD_SIUL2 includes a device ID register (DIDR) that identifies the BD type and version, allowing a connected tool or development/calibration software running on a connected PD to determine what debug or calibration resources are available. The DIDR uses similar formatting to the MCU ID registers (MIDR1, MIDR2) included in the SIUL2 module.

This information is made available via the BD SIU rather than via the PD because it is either not available via the PD (for example, information on BD-specific resources, such as extended overlay RAM size) or because it is useful for the tool to be able to access the information before the PD is powered up.

Information available includes:

- An ID code that can be used to uniquely identify the BD type
- A version field that is changed for every major mask revision of the BD
- A parameterized field that indicates the amount of extended overlay RAM included
    - The field coding allows indication of up to 4 MB of RAM, with resolution of 256 KB steps

It should be noted that in addition to information available in the DIDR, a connected tool can also retrieve identification information from other sources including:

- The JTAG ID returned by the buddy DCI/JTAGC module, which is different than the JTAG ID returned by a standard PD DCI/JTAGC
- The MIDR register included in the PD SIUL2 module, which includes information on the PD device type, version and package type (including identification of emulation and debug device package variants)

3.3 Device status information

This section describes the devices status information.

3.3.1 Initialization status

The BD SIU includes a flag (BD_INIT) that can be used by the connected PD or any connected debug or calibration tools to track the status of the BD.

The main expected usage mode for this bit is to allow the Boot Assist Flash (BAF) or user calibration software on the PD to determine whether the BD has lost power since it was last initialized. A typical usage sequence for BD_INIT is shown below.

- At BD power on reset (POR), BD_INIT is cleared automatically
  - BD_INIT is not cleared by external BD reset input
- A calibration tool or calibration software initializes the BD (for example, writing BD overlay RAM with calibration data and remap configuration)
- The calibration tool or calibration software sets BD_INIT
- After PD POR, the BAF executes and reads BD_INIT
  - If BD_INIT is clear, the BD overlay is assumed to be uninitialized and is not used for calibration remap
  - If BD_INIT is set, the BD overlay is assumed to be initialized and can be used for calibration remap. (Other tests to verify data integrity may be used.)

BD_INIT is cleared by BD POR, and can be set or cleared via the debug interface or writes from the PD. BD_INIT is only used to detect full power down events resulting in a subsequent POR. It does not detect brown out drops in the supply voltage.

3.3.2 PD reset status

The IPS accessible BD_SIU_RSR register shows what stage of reset processing the connected PD is in, and whether the PD is powered down or not. This is functionally different from the software accessible registers in the PD reset generation module that indicate which reset source has caused the last PD reset.

3.4 Memory mapped registers

This table gives an overview of the BD_SIUL2 registers.

BD_SIUL2 memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Device Identification Register 1 (BD_SIUL2_DIDR1)	32	R	57BD_0000h	3.4.1/28
4	Device Identification Register 2 (BD_SIUL2_DIDR2)	32	R	0003_0100h	3.4.2/29
100	Initialization Status Register (BD_SIUL2_ISR)	32	R/W	0000_0000h	3.4.3/30
104	Reset Status Register (BD_SIUL2_RSR)	32	R	0000_0000h	3.4.4/31

3.4.1 Device Identification Register 1 (BD_SIUL2_DIDR1)

The BD_SIUL2_DIDR1 contains the part number and mask number.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PARTNUM																Reserved								MASKNUM_ MAJOR				MASKNUM_ MINOR			
W																																
Reset	0	1	0	1	0	1	1	1	1	0	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BD_SIUL2_DIDR1 field descriptions

Field	Description
0–15 PARTNUM	Device Part Number. Read-only, mask programmed part number of the device. Reads 0x57BD.
16–23 Reserved	This field is reserved.
24–27 MASKNUM_ MAJOR	Major Mask Revision Number. Read-only, mask programmed mask number of the device. Reads 0x0 for the initial mask set of the device, and changes sequentially for each mask set.
28–31 MASKNUM_ MINOR	Minor Mask Revision Number. Read-only, mask programmed mask number of the MCU. Reads 0x0 for the initial mask set of the device, and changes sequentially for each mask set.

3.4.2 Device Identification Register 2 (BD_SIUL2_DIDR2)

The BD_SIUL2_DIDR2 contains a complement to the part number and amount of RAM available.

Address: 0h base + 4h offset = 4h

Bit	0–11	12–15	16–23	24–31
R	Reserved	RAM_SIZE	PARTNUM	Reserved
W				
Reset	0 0 0 0 0 0 0 0 0 0 0 0	0 0 1 1	0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0

BD_SIUL2_DIDR2 field descriptions

Field	Description
0–11 Reserved	This field is reserved.
12–15 RAM_SIZE	RAM Size. Read-only, indicates the amount of overlay RAM available in the device, with resolution of 256K byte steps 0000 0.25M byte 0001 0.5M byte 0010 0.75M byte 0011 1M byte 0100 1.25M byte 0101 1.5M byte 0110 1.75M byte 0111 2M byte 1000 2.25M byte 1001 2.5M byte 1010 2.75M byte 1011 3M byte 1100 3.25M byte 1101 3.5M byte 1110 3.75M byte 1111 4M byte
16–23 PARTNUM	Device Part Number. Read-only, mask programmed complement to the part number of the device. For the BD 1M byte part with part number 57BD1, BD_SIUL2_DIDR1.PARTNUM=57BD, and BD_SIUL2_DIDR2.PARTNUM=1. 01 1 (BD1M) 02 2 (BD2M)
24–31 Reserved	This field is reserved.

3.4.3 Initialization Status Register (BD_SIUL2_ISR)

The BD_SIUL2_ISR contains a flag that can be used by the PD or any connected debug or calibration tool to track the status of the BD. This register is reset by POR.

The main expected use case for this flag is to allow software on the PD to determine whether the BD has lost power since it was last initialized. A typical sequence is:

- At power on reset (POR), the flag is cleared automatically (not affected by software reset)
- A calibration tool or PD calibration software initializes the BD (for example, writing BD overlay RAM with calibration data and remap configuration)
- The calibration tool or PD calibration software sets the BD_INIT flag
- After PD POR, the PD boot software executes and reads BD_INIT
- If BD_INIT is clear, BD overlay is assumed to be uninitialized and is not used for calibration remap
- If BD_INIT is set, BD overlay is assumed to be initialized and can be used for calibration remap (other tests to verify data integrity may be used)

BD_INIT is cleared by BD POR, and can be set or cleared via writes from the PD or from the Nexus RWA.

Address: 0h base + 100h offset = 100h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved															BD_ INIT
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BD_SIUL2_ISR field descriptions

Field	Description
0–30 Reserved	This field is reserved.
31 BD_INIT	Initialization Flag. This is a read/write bit that can be used to indicate that the BD was initialized. Software reset does not affect the flag. Power-on reset clear the flag. 0 BD not initialized 1 BD initialized

3.4.4 Reset Status Register (BD_SIUL2_RSR)

The BD_SIUL2_RSR contains reset status information about the PD.

Address: 0h base + 104h offset = 104h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved													PD_ RST	PD_ TRIM	PD_ POR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BD_SIUL2_RSR field descriptions

Field	Description
0–28 Reserved	This field is reserved.
29 PD_RST	PD Reset Flag. Read-only, indicates if PD has finished the reset sequence. 0 PD reset sequence not completed 1 PD reset sequence completed
30 PD_TRIM	PD Trim Flag. Read-only, indicates if PD has finished trimming phase during the reset sequence. 0 PD trimming phase not completed 1 PD trimming phase completed
31 PD_POR	PD POR Flag. Read-only, indicates whether PD is powered down or not. 0 PD is powered down or power on reset has not completed 1 PD is powered and power on reset has completed

Chapter 4
Buddy Device—Debug and Calibration Interface (DCI)

4.1 Introduction

The Debug and Calibration Interface (DCI) module provides debug and calibration features for the MCU. It includes a standard IEEE 1149.1/1149.7 compatible JTAG interface which is used to connect with an external JTAG tool using a low frequency clock interface.

It also features a software based debug mode which can be controlled by the MCU without using an external tool. Additionally, the DCI provides features for debug control using break-points and synchronous restart of the cores. This chapter describes the DCI features in the Emulation and Debug Device (ED) that includes features in the Production device (PD) as well as features in the Buddy device (BD). There are two DCIs on the ED which are designated PD DCI and BD DCI.

4.1.1 Features

The DCI features are the following:

- Debug mode enable control for connected tool or software
- Dual DCI system for multi-device clients
- IEEE 1149.1 and IEEE 1149.7 controllers
- Debug break and cross-triggering control
- Synchronous restart control for CPU(s) when exiting debug mode
- Tool hot plug capability

- Security access control
- Debug reset control

4.1.2 Overview

The DCI can take the following different inputs for its JTAG signals:

- P.JTAG—JTAG signals coming from the Buddy device JTAG pads. This is the standard 5-pin JTAG interface and is the default operating mode.
- M.JTAG—JTAG signals generated by JTAGM module using software debug mode.
    - In software debug mode, there is no tool, and debug software running on the MCU generates the JTAG packets and writes them to JTAGM using special debug registers in the JTAGM module.
- I.JTAG— JTAG signals interconnecting the two DCI implementations in the ED coming from the intra-die pads.

By default the DCI operates in standard 5-pin JTAG mode (IEEE 1149.1). It can be configured in 3-pin reduced pin JTAG mode (IEEE 1149.7) after starting from standard 5-pin JTAG mode.

When reduced pin JTAG mode (IEEE 1149.7) is configured TMSC pin is configured in bidirectional mode during operation.

The DCI also provides a centralized break control for system IPs and cores, which can be controlled by an external tool as well as the internal debug peripherals such as timers/ SPU, etc. The DCI also provides the Event Out ($\overline{\text{EVTO}}$) signals to the debug tool in case of the occurrence of any internal debug event.

This figure shows the DCI block diagram.

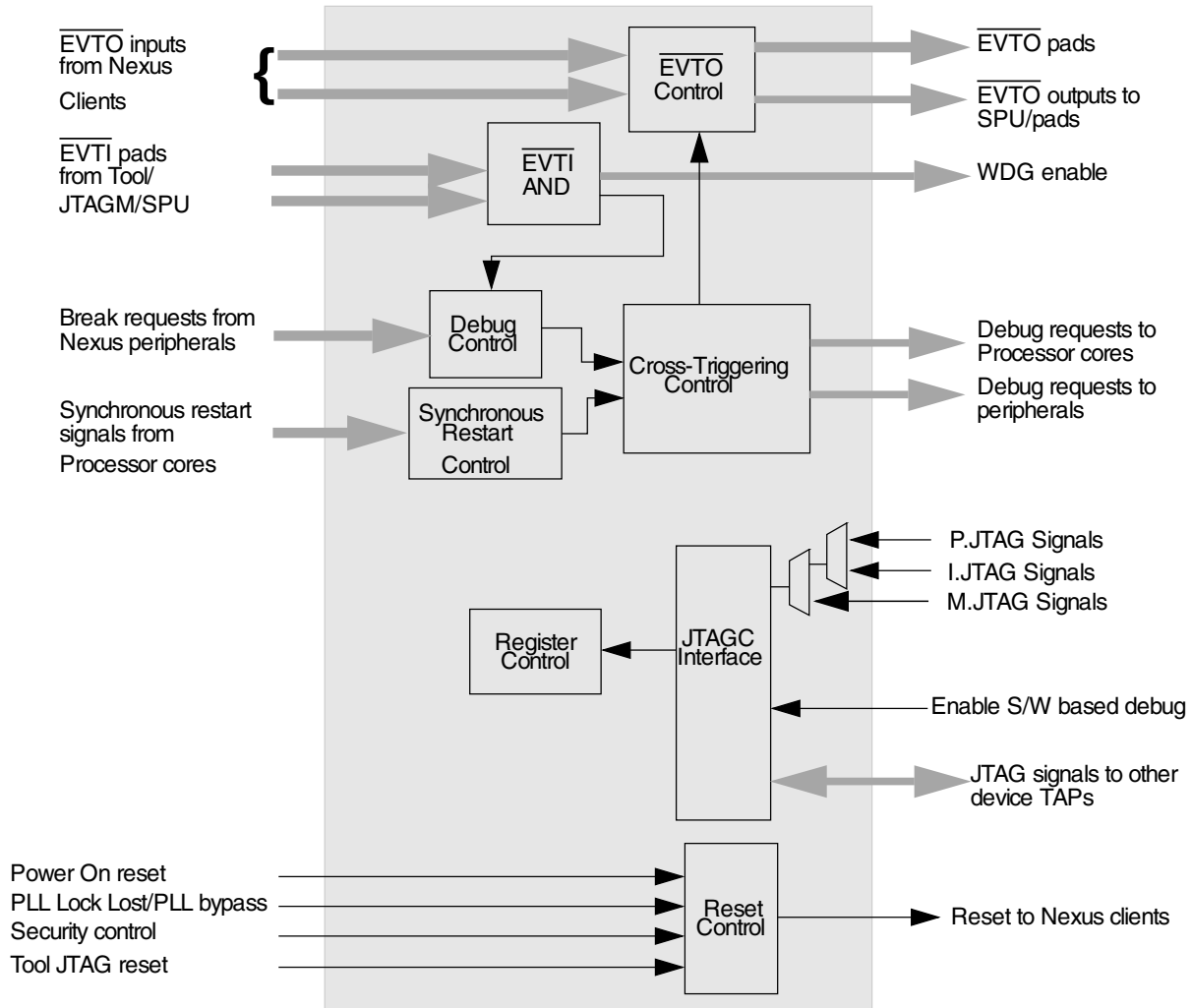


Figure 4-1. DCI block diagram

When the DCI is used in an ED package (PD and the BD together) its configuration is shown as in the following figure. In the ED package, a debug tool is connected to the BD pins and connects to the PD with the I.JTAG interface which are routed using inter-die pads.

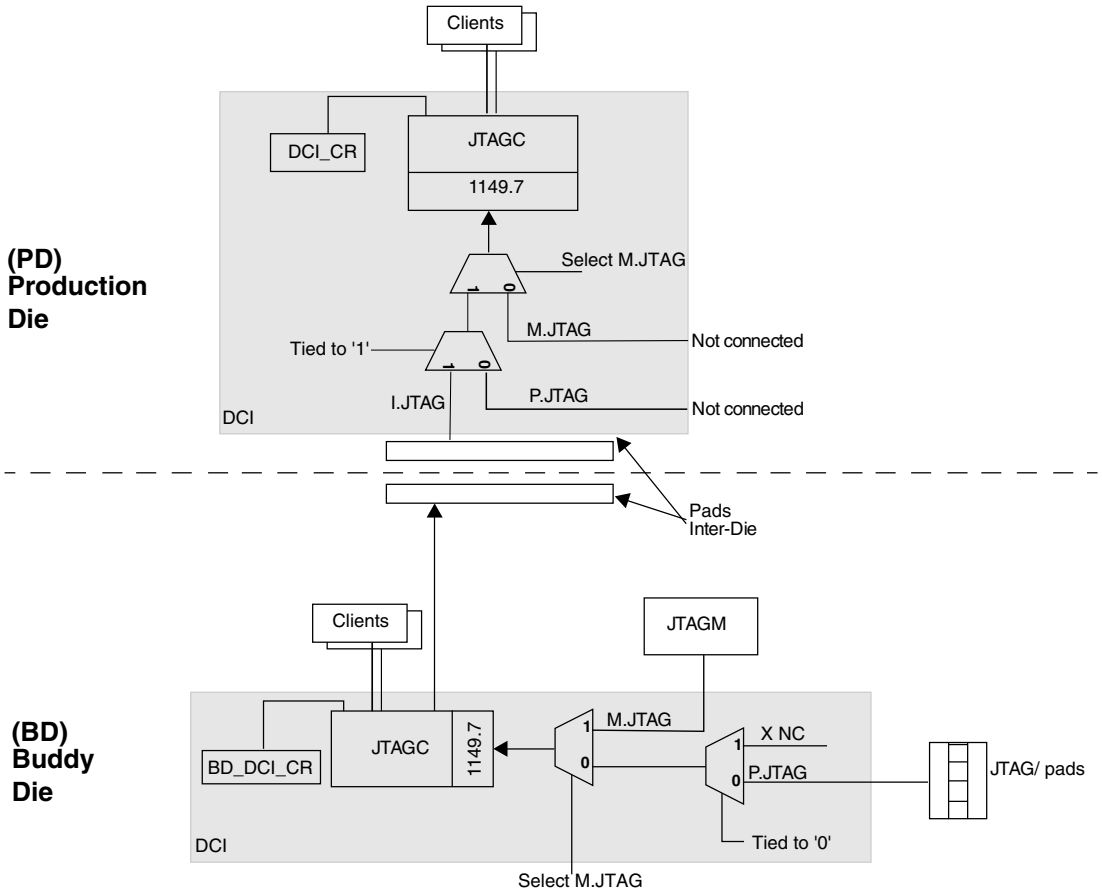


Figure 4-2. DCI in an emulation configuration

4.1.3 Operating modes

Selection between various JTAG sources is done using a JTAGM register bit, JTAGM_MCR[DTM]. The JTAGM bit, JTAGM_MCR[DTM], is programmable via memory-mapped registers. JTAGM is selected as a JTAG source when JTAGM_MCR[DTM] is set.

This table shows a summary of the configuration options.

Table 4-1. DCI JTAG source selection

JTAGM_MCR[DTM]	DCI Source	JCOMP
0	P.JTAG (Standard JTAG)	Asserted
1	M.JTAG (Software)	Negated

The reset signal to the DCI block consists of the following:

- Power-on reset, Low/High voltage detection on IO/Core supplies
- JCOMP reset

The DCI puts the JTAGC TAP in reset when either power-on/low-voltage-detect reset is asserted or JCOMP is negated. In the ED, the BD DCI is reset with BD POR/Low voltage detection on BD_VDD.

The M.JTAG provides an alternate interface to the debug logic.

When no debug tool is connected, device power-on reset puts the DCI in standard 5-pin JTAG mode. This configures the TDO pin in output mode with the reset value (1'b0). There is no change in port values if no debug tool is connected.

This table contains the operating mode descriptions.

Table 4-2. DCI operating modes

Mode	Description
DCI_JTAG	JTAG mode. This is the default state after power-on reset. In this mode the DCI is in JTAG mode (controlled with P.JTAG or I.JTAG) and is controlled by an external JTAG tool.
DCI_SW_DBG	Software Debug State. This state is achieved when no tool is present and software sets the JTAGM_MCR[DTM] bit. The DCI is then controlled using M.JTAG signals.

4.1.3.1 Software enabled debug and calibration mode

Apart from the JTAG sources, such as a JTAG-based tool, it is possible to generate JTAG sequences through software programming.

To support a software debug session via a CAN or SPI link, the software can enable debug and calibration mode via an memory-mapped register. It is done first by setting a control bit in the JTAGM (JTAGM_MCR[DTM]). For more details, refer to the JTAGM documentation. When enabled, the DCI is configured to receive M.JTAG signals which are driven by JTAGM registers accessible through software. Software can control all DCI features like any other JTAG source.

After a system reset, the JTAGM_MCR[DTM] bit is reset and the JCOMP pad is pulled-low, thus keeping the DCI in the reset state. Therefore, software is required to first change to a non-secured mode. Then it can program the JTAGM_MCR[DTM] bit, to select M.JTAG as the source inside the DCI, and then set the JTAGM_MCR[jtagm_JCOMP] bit present in the JTAGM, which takes all debug modules out of reset.

When MCR[jtagm_JCOMP] is programmed to 0, the JTAGC block reset is asserted, thus putting the debug and calibration logic in reset state. This bit is used during software based debug to initialize the debug and calibration logic.

Software based debug is not possible when a tool is connected. The presence of the tool is determined using the rising edge of the JCOMP pad to set an internal flag which, when set, disables the selection of JTAGM as a JTAG source.

NOTE

It is recommended to utilize either hardware based debug (JTAG) or software based debug. If only one of the debug method is used, all three connection methods (connect and reset target, connect and break application, connect and keep application running) are supported.

4.1.3.2 Nexus reset control

The JCOMP input that is used as the primary reset signal for the DCI is also used by the DCI to generate a single-bit reset signal for other Nexus blocks. It is used to reset all debug functions without affecting any system level functions. The DCI provides the reset signal for all Nexus blocks, with the PD DCI providing the reset signal to all PD debug blocks, and the BD DCI providing the reset signal for all BD debug blocks. This reset is controlled using the following signals:

- Power-on reset/Low voltage detect (applicable to both PD and BD)
- JCOMP (applicable to both PD and BD)
- SoC Debug_Enable signal (applicable to PD only)
- System PLL is out of lock state and not bypassed (applicable to PD only)

Asserting power-on reset, negating JCOMP, or negating the Debug_Enable signal, results in asynchronous entry into reset state. Following negation of power-on reset, the BD Nexus reset signal remains asserted until the JCOMP signal is asserted. Any subsequent loss of PLL lock does not generate a Nexus reset. This reset signal is unaffected by other sources of reset.

If Debug_Enable is de-asserted by the SoC security modules (PASS), then the DCI holds the debug and trace logic in reset. The Debug_Enable signal is generated by security logic after functional or destructive reset based on the device censored mode, the PASS_LOCK3[DBL] control, and the device life cycle. If the Debug_Enable signal is asserted by the security logic, it is possible to trace through functional resets

The following table shows how the Nexus reset is asserted/negated based on various signals.

Table 4-3. Nexus Reset Truth Table

POR	JCOMP	Debug_Enable	PLL lock	Nexus reset
Assert	X	X	X	Assert
X	Negate	X	X	Assert
X	X	Disable	X	Assert
X	X	X	Unlocked	Assert
Negate	Assert	Enable	Locked	Negate

4.1.3.3 System watchdog control

The BD DCI does not include the Software Watchdog Timer (SWT) disable feature.

4.1.3.4 Reset control

The BD DCI includes control bits to allow a connected tool to force a reset of the connected device. The description is as follows:

- Force BD Reset—This reset is generated by programming DCI_CR[FBD_RESET] bit. When set it forces reset. This bit has to be reset by the tool to negate the reset signal.

These reset signals are directly connected to the respective DCI_CR bits. The DCI does not reset the DCI_CR bits after generation of the respective reset signals. The tool must specifically reset the appropriate DCI_CR bits to remove the reset signals. These reset control bits can be set/reset through JTAG programming, and only reset through JCOMP assertion or Power-on-reset.

4.2 External signal description

The JTAGC module used inside DCI module defines the interface with external signals. It consists of five signals (TMS/TDI/TDO/TCK/JCOMP) that connect to off chip development tools and allow access to test support functions. For more details for JTAG signals, refer to JTAGC documentation.

4.3 Register description

The PD DCI module contains two registers, DCI_CR and DCI_PINCR, which are present inside the JTAGC module. The BD DCI only contains the DCI_CR and does not contain the DCI_PINCR. These registers are only controlled using the JTAGC TAP and no memory-mapped registers are present.

4.3.1 DCI control register (BD_DCI_CR)

The BD_DCI_CR is a 32-bit data register. This register is programmed using JTAG shift with the corresponding enable instruction described in the JTAGC chapter. The following table shows the format of the BD_DCI_CR.

NOTE

The BD version of this register is refered to as DCI_CR in the other sections of this chapter.

In an ED which has both PD and BD instances of the DCI, all bits must be programmed on the BD DCI instance. The only bits of this register that do not follow this rule are EVTI1_PAD_EN and EVTI0_PAD_EN, which must be programmed in both DCI instances.

	31	30	29	28	27	26	25	24
R W	Reserved							
Reset	1	1	1	1	1	1	1	1

	23	22	21	20	19	18	17	16
R W	Reserved							
Reset	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8
R W	Reserved					Reserved	Reserved	
Reset	1	1	1	1	1	0	0	0

	7	6	5	4	3	2	1	0
R W	FBD_RESET	Reserved	EVTI_1_PAD_EN	EVTI_0_PAD_EN	Reserved		Reserved	Reserved
Reset	0	0	0	0	0	0	0	0

The BD_DCI_CR is described in this table.

Table 4-4. BD_DCI_CR field descriptions

Field	Description	
7 FBD_RESET	Force Reset	
	0	Do not force Reset
	1	Force Reset
5 EVTI1_PAD_EN[1]	$\overline{\text{EVTI}}$[1] feature enable	
	0	$\overline{\text{EVTI}}$[1] disabled (default)
	1	$\overline{\text{EVTI}}$[1] enabled
4 EVTI0_PAD_EN[1]	$\overline{\text{EVTI}}$[0] feature enable	
	0	$\overline{\text{EVTI}}$[0] disabled (default)
	1	$\overline{\text{EVTI}}$[0] enabled

1. In an ED that has both PD and BD, EVTI1_PAD_EN and EVTI0_PAD_EN must be programmed in both BD and PD instances of the DCI in order to allow events generated on the BD to pass through the BD DCI and reach the PD DCI.

4.3.2 DCI EVTx pin multiplexing control register (DCI_PINCR)

The DCI_PINCR is a 32-bit data register. This register is programmed using JTAG shift with the corresponding enable instruction described in the JTAGC chapter. For a description of the DCI_PINCR functionality, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Chapter 5
Buddy Device—JTAG Controller (JTAGC)

5.1 Introduction

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format.

5.1.1 Block diagram

The following is a simplified block diagram of the JTAG Controller (JTAGC) block. Refer to the chip-specific configuration information as well as Register description for more information about the JTAGC registers.

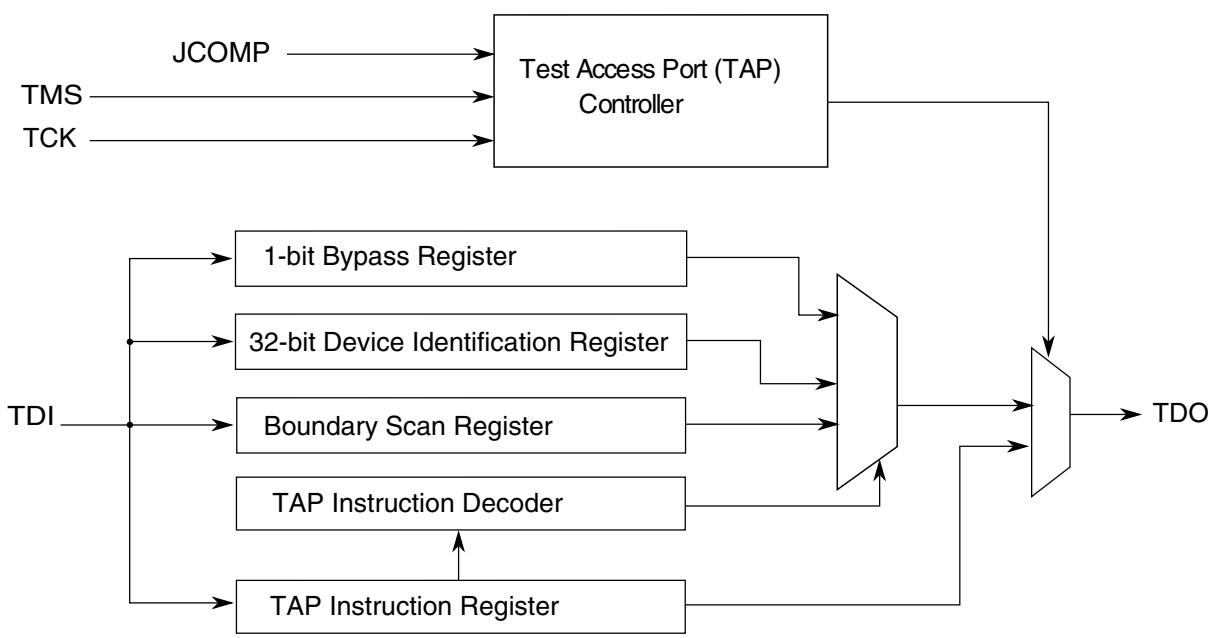


Figure 5-1. JTAG (IEEE 1149.1) block diagram

5.1.2 Features

The JTAGC block is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface
  - 4 pins (TDI, TMS, TCK, and TDO)
- JCOMP input that provides reset control
- Instruction register that supports several IEEE 1149.1-2001 defined instructions as well as several public and private device-specific instructions. Refer to Table 5-4 for a list of supported instructions.
- Sharing of the TAP with other TAP controllers via ACCESS_AUX_x instructions
- JTAG_PASSWORD register
- Bypass register, boundary scan register, data registers, and device identification register.
- TAP controller state machine that controls the operation of the data registers, instruction register and associated circuitry.

5.1.3 Modes of operation

The JTAGC block uses JCOMP and a power-on reset indication as its primary reset signals. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

5.1.3.1 Reset

The JTAGC block is placed in reset when either power-on reset is asserted, JCOMP is negated, or the TMS input is held high for enough consecutive rising edges of TCK to sequence the TAP controller state machine into the Test-Logic-Reset state. Holding TMS high for five consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Asserting power-on reset or setting JCOMP to a value other than the value required to enable the JTAGC block results in asynchronous entry into the reset state. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered
- The instruction register is loaded with the IDCODE instruction

5.1.3.2 IEEE 1149.1-2001 defined test modes

The JTAGC block supports several IEEE 1149.1-2001 defined test modes. A test mode is selected by loading the appropriate instruction into the instruction register while the JTAGC is enabled. Supported test instructions include EXTEST, HIGHZ, CLAMP, SAMPLE and SAMPLE/PRELOAD. Each instruction defines the set of data register(s) that may operate and interact with the on-chip system logic while the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is enabled for serial access between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. The single-bit bypass register shift stage is enabled for serial access between TDI and TDO when the BYPASS, HIGHZ, CLAMP or reserved instructions are active. The functionality of each test mode is explained in more detail in JTAGC block instructions.

5.1.3.3 Bypass mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAGC block into bypass mode. While in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

5.2 External signal description

The JTAGC consists of a set of signals that connect to off chip development tools and allow access to test support functions. The JTAGC signals are outlined in the following table and described in the following sections.

Table 5-1. JTAG signal properties

Name	I/O	Function	Reset State	Pull
TCK	Input	Test Clock	—	Down

Table continues on the next page...

Table 5-1. JTAG signal properties (continued)

Name	I/O	Function	Reset State	Pull
TDI	Input	Test Data In	—	Up
TDO	Output	Test Data Out	High Z[1]	—
TMS	Input	Test Mode Select	—	Up
JCOMP	Input	JTAG Compliancy	—	Down

1. TDO output buffer enable is negated when the JTAGC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented at the TDO pad for use when JTAGC is inactive.

5.2.1 TCK—Test clock input

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the TAP.

5.2.2 TDI—Test data input

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

5.2.3 TDO—Test data output

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is three-stateable and is actively driven only in the Shift-IR and Shift-DR states of the TAP controller state machine, which is described in TAP controller state machine.

5.2.4 TMS—Test mode select

Test Mode Select (TMS) is an input pin used to sequence the IEEE 1149.1-2001 test control state machine. TMS is sampled on the rising edge of TCK.

5.2.5 JCOMP—JTAG compliancy

The JCOMP signal provides IEEE 1149.1-2001 compatibility and provides the ability to share the TAP. The JTAGC TAP controller is enabled when JCOMP is set to the JTAGC enable encoding, otherwise the JTAGC TAP controller remains in reset.

5.3 Register description

This section provides a detailed description of the JTAGC block registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

5.3.1 Instruction register

The JTAGC block uses a 6-bit instruction register as shown in the following figure. The instruction register allows instructions to be loaded into the block to select the test to be performed or the test data register to be accessed or both. Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the IDCODE instruction. During the Capture-IR TAP controller state, the instruction shift register is loaded with the value 000001b, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

	5	4	3	2	1	0
R	0	0	0	0	0	1
W	Instruction Code					
Reset:	0	0	0	0	0	1

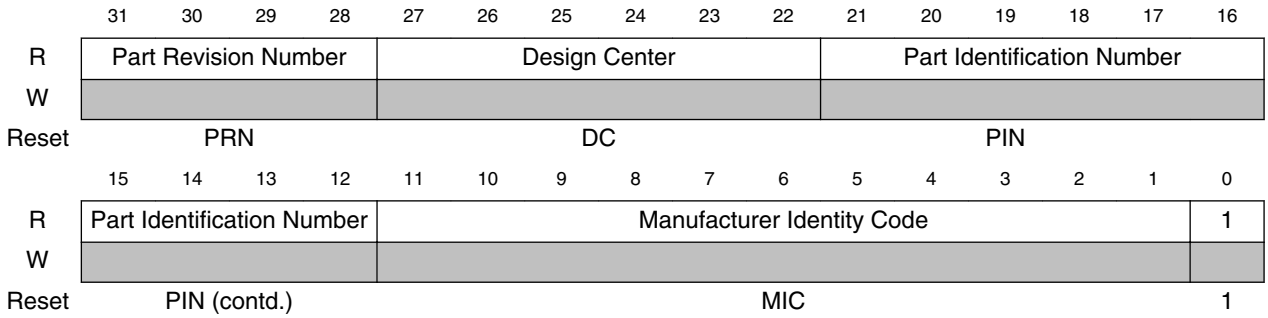
Figure 5-2. Instruction register

5.3.2 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS, CLAMP, HIGHZ or reserve instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

5.3.3 Device identification register

The device identification (JTAG ID) register, shown in the following figure, allows the revision number, part number, manufacturer, and design center responsible for the design of the part to be determined through the TAP. The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is active. Entry into the Capture-DR state while the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the Update-DR state.



The following table describes the device identification register functions.

Table 5-2. Device identification register field descriptions

Field	Description
PRN	Part Revision Number. Contains the revision number of the part. Value is 0x0.
DC	Design Center. Indicates the design center. Value is 0x2B.
PIN	Part Identification Number. Contains the part number of the device. 0x381.
MIC	Manufacturer Identity Code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID. Value is 0x00E.
IDCODE ID	IDCODE Register ID. Identifies this register as the device identification register and not the bypass register. Always set to 1.

5.3.4 JTAG_PASSWORD register

The JTAG_PASSWORD register is a 256-bit shift register path from TDI to TDO selected when the ENABLE_JTAG_PASSWORD instruction is active. The default reset value of the JTAG_PASSWORD register is 256'b0. The JTAG_PASSWORD register transfers its value to a parallel hold register on the rising edge of TCK when the TAP controller state machine is in the Update-DR state. Once the ENABLE_JTAG_PASSWORD instruction is executed, the register value remains valid until a JTAG reset occurs. The operation of this register is described in the security documentation.

	255	...	2	1	0
R	JTAG_PASSWORD				
W					
Reset:	*[1]	*	*	*	*

1. The reset value of JTAG_PASSWORD is 256 'b0.

The following table describes the JTAG_PASSWORD register functions.

Table 5-3. JTAG_PASSWORD register field descriptions

Field	Description
JTAG_PASSWORD	JTAG Password. The JTAG_PASSWORD bits are used to provide the JTAG password for security.

5.3.5 Boundary scan register

The boundary scan register is connected between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. It is used to capture input pin data, force fixed values on output pins, and select a logic value and direction for bidirectional pins. Each bit of the boundary scan register represents a separate boundary scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in Boundary scan. The size of the boundary scan register and bit ordering is device-dependent and can be found in the device BSDL file.

The BD boundary scan chain includes registers connected to all PD/BD interface signals at the BD side. These registers are controlled by the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions applied to the BD JTAG controller. Similarly, the PD boundary scan chain also contains registers connected to the interface signals at the PD side, which are controlled by instructions applied to the PD JTAG controller.

5.4 Functional description

This section explains the JTAGC functional description.

5.4.1 JTAGC reset configuration

While in reset, the TAP controller is forced into the Test-Logic-Reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

5.4.2 IEEE 1149.1-2001 (JTAG) Test Access Port

The JTAGC block uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. Ownership of the port is determined by the value of the currently loaded instruction. For more detail on TAP sharing via JTAGC instructions refer to ACCESS_AUX_x instructions.

Data is shifted between TDI and TDO though the selected register starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register, test data registers, and the bypass register.

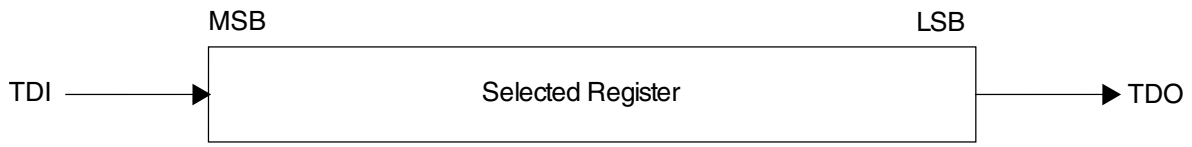
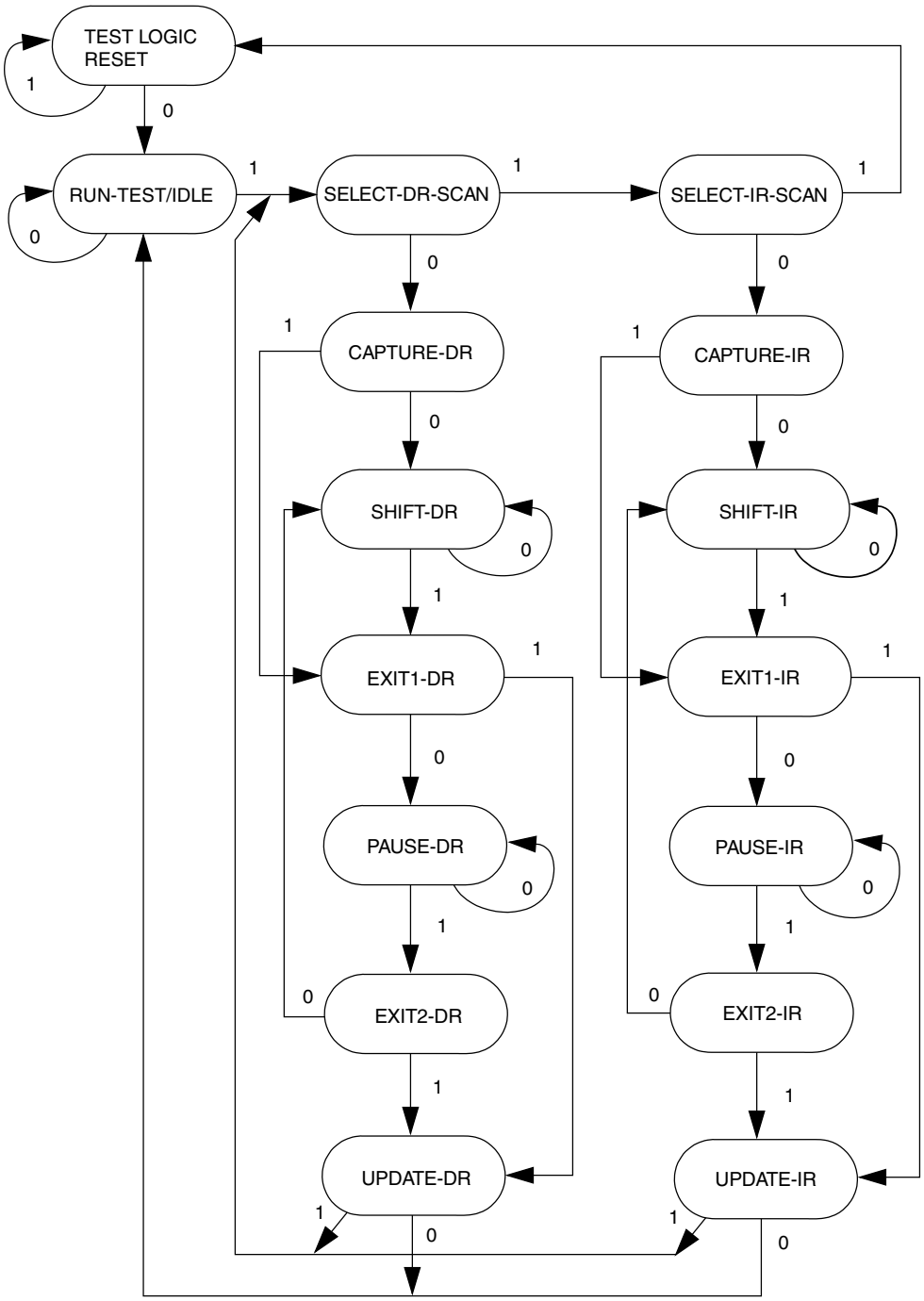


Figure 5-3. Shifting data through a register

5.4.3 TAP controller state machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin. The following figure shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCK signal. As the following figure shows, holding TMS at logic 1 while clocking TCK through a sufficient number of rising edges also causes the state machine to enter the Test-Logic-Reset state.



The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

Figure 5-4. IEEE 1149.1-2001 TAP controller finite state machine

5.4.3.1 Enabling the TAP controller

The JTAGC TAP controller is enabled by setting JCOMP to a logic 1 value.

5.4.3.2 Selecting an IEEE 1149.1-2001 register

Access to the JTAGC data registers is achieved by loading the instruction register with any of the JTAGC block instructions while the JTAGC is enabled. Instructions are shifted in via the Select-IR-Scan path and loaded in the Update-IR state. At this point, all data register access is performed via the Select-DR-Scan path.

The Select-DR-Scan path is used to read or write the register data by shifting in the data (LSB first) during the Shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the Capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the Update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

5.4.4 JTAGC block instructions

The JTAGC block implements the IEEE 1149.1-2001 defined instructions listed in the following table. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 5-4. General 6-bit JTAG instructions

Instruction	Code[5:0]	Instruction Summary
IDCODE	000001	Selects device identification register for shift
SAMPLE/PRELOAD	000010	Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation
SAMPLE	000011	Selects boundary scan register for shifting and sampling without disturbing functional operation
EXTEST	000100	Selects boundary scan register and applies preloaded values to output pins. **NOTE:** Execution of this instruction asserts functional reset.
Factory debug reserved	000101	Intended for factory debug only
Factory debug reserved	000110	Intended for factory debug only
ENABLE_JTAG_PASSWORD	000111	Selects JTAG_PASSWORD register
HIGHZ	001001	Selects bypass register and three-states all output pins. **NOTE:** Execution of this instruction asserts functional reset.
Factory debug reserved	001010	Intended for factory debug only
CLAMP	001100	Selects bypass register and applies preloaded values to output pins. **NOTE:** Execution of this instruction asserts functional reset.

Table continues on the next page...

Table 5-4. General 6-bit JTAG instructions (continued)

Instruction	Code[5:0]	Instruction Summary
ENABLE_DCI_CR	001110	Enables access to DCI CR register
Reserved	001111	Reserved
BYPASS	111111	Selects bypass register for data operations
Reserved[1]	All other opcodes	Decoded to select bypass register
ACCESS_AUX_x	100000-111110	Grants one of the auxiliary TAP controllers ownership of the TAP

1. The manufacturer reserves the right to change the decoding of reserved instruction codes in the future

The ACCESS_AUX instructions are described in the chip configuration debug information.

5.4.4.1 IDCODE instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAGC block is reset.

5.4.4.2 SAMPLE/PRELOAD instruction

The SAMPLE/PRELOAD instruction has two functions:

- The SAMPLE portion of the instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE/PRELOAD instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. Both the data capture and the shift operation are transparent to system operation.
- The PRELOAD portion of the instruction initializes the boundary scan register cells before selecting the EXTEST or CLAMP instructions to perform boundary scan tests. This is achieved by shifting in initialization data to the boundary scan register during the Shift-DR state. The initialization data is transferred to the parallel outputs of the boundary scan register cells on the falling edge of TCK in the Update-DR state. The data is applied to the external output pins by the EXTEST or CLAMP instruction. System operation is not affected.

5.4.4.3 SAMPLE instruction

The SAMPLE instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. There is no defined action in the Update-DR state. Both the data capture and the shift operation are transparent to system operation.

5.4.4.4 EXTEST External test instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins. Typically, the preloaded data is loaded into the boundary scan register using the SAMPLE/PRELOAD instruction before the selection of EXTEST. EXTEST asserts the internal system reset for the MCU to force a predictable internal state while performing external boundary scan operations.

5.4.4.5 ENABLE_JTAG_PASSWORD instruction

The ENABLE_JTAG_PASSWORD instruction selects the JTAG_PASSWORD register for connection as the shift path between TDI and TDO.

5.4.4.6 HIGHZ instruction

HIGHZ selects the bypass register as the shift path between TDI and TDO. While HIGHZ is active all output drivers are placed in an inactive drive state (e.g., high impedance). HIGHZ also asserts the internal system reset for the MCU to force a predictable internal state.

5.4.4.7 CLAMP instruction

CLAMP allows the state of signals driven from MCU pins to be determined from the boundary scan register while the bypass register is selected as the serial path between TDI and TDO. CLAMP enhances test efficiency by reducing the overall shift path to a

single bit (the bypass register) while conducting an EXTEST type of instruction through the boundary scan register. CLAMP also asserts the internal system reset for the MCU to force a predictable internal state.

5.4.4.8 ACCESS_AUX_x instructions

The JTAGC is configurable to allow other TAP controllers on the device to share the port with it. This is done by providing ACCESS_AUX_x instructions for each of these TAP controllers. When this instruction is loaded, control of the JTAG pins are transferred to the selected TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The JTAGC regains control of the JTAG port during the UPDATE-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive. Instructions not used to access an auxiliary TAP controller on a device are treated like the BYPASS instruction.

5.4.4.9 BYPASS instruction

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions. While the BYPASS instruction is active the system logic operates normally.

5.4.5 Boundary scan

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST, SAMPLE, or SAMPLE/PRELOAD instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

PD and the BD have their own separate boundary scan chains, each one controlled by the respective JTAG controller. Besides the package I/O pins, these chains also encompasses the interface signals between BD and PD. Testing of the interface can be done by applying EXTEST, SAMPLE or SAMPLE/PRELOAD instructions to both PD and BD JTAG.

5.5 Initialization/Application information

The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

To initialize the JTAGC block and enable access to registers, the following sequence is required:

1. Set the JCOMP signal to the JTAGC enable value, thereby enabling the JTAGC TAP controller.
2. Load the appropriate instruction for the test or action to be performed.

Chapter 6
Buddy Device—Nexus Aurora Router (NAR)

6.1 Introduction

On the device, there are multiple blocks that require development interface support. The blocks that are Nexus-compliant (based on the IEEE-ISTO 5001 standard) are expected to interface with the development tool through a dedicated high-bandwidth debug port. The Nexus Aurora Router (NAR) controls the usage of the output port in a manner that allows all the individual development interface blocks to share the port, and appear to the development tool to be a single block.

Top-level architecture is shown in the following figure. Multiple clients send messages by sharing the high-bandwidth port through the NAR. For a list of the specific clients that are connected to the NAR, see the Calibration and Debug configuration chapter which describes how the modules are configured. The Nexus Aurora Link (NAL) takes the 32-bit Nexus message beats from the NAR and processes them in Aurora-ready data octets for the Nexus Aurora Phy (NAP) block, which interfaces directly with the high-bandwidth port. The Aurora output port of NAR can be fed as a client to another NAR.

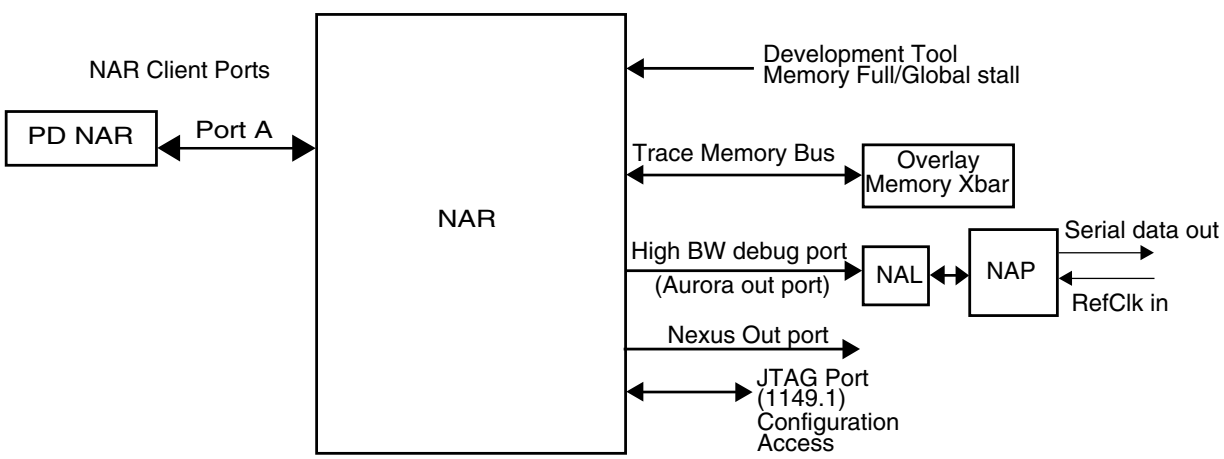


Figure 6-1. NAR top level block diagram

The trace memory bus port provides added flexibility to the channel trace data to memory on the Nexus Aurora crossbar. The Nexus Out port provides additional flexibility to the channel trace data to memory on the Nexus recipient. Setup and configuration of the NAR is performed by accessing the configuration registers via a JTAG (1149.1) port.

Each Nexus client has a dedicated receive queue to allow seamless switching between sources at Nexus message boundaries. Incoming Nexus data is parsed into the receive queues so that end-of-message can be determined. Messages are then transferred into the main queue. An arbitration block determines the order in which clients are serviced, using an optional client request level bus to prioritize the data (lacking request level data, a simple round-robin algorithm is used). The NAR has two output ports: the high bandwidth debug port (Aurora out) and the trace memory bus. If both output ports are enabled at the same time, the main queue is logically partitioned inside NAR. These two partitions of the main queue are supported for better throughput each having 50% size. The two NAR output ports can be in operation simultaneously, with the output arbitration block splitting out messages by type or by source. Data going through the Aurora out port or the Trace port goes through an additional translation block to put the data in the correct format. The actual Aurora formatting is done in the NAL. Additional blocks for message generation and timestamp generation are described in more detail in the following sections.

The Buddy Device (BD) NAR supports the following client interface:

- Aurora type interface with clock domain transfer at the same frequency

6.2 Overview

The NAR serves as an arbiter for the high-bandwidth debug port (HBDP). The NAR pulls Nexus messages from multiple clients, queues the data, and forwards it (based on the user-controlled configuration) to the high-speed Aurora Out port and/or the Trace port. The trace memory bus controller allows the NAR to transmit data via the normal system interface. NAR control and configuration registers are accessed via JTAG (IEEE 1149.1) port.

6.3 Features

The NAR supports the following features:

- Connects 1 client(s) with dedicated receive queue(s). Clients can transmit data as long as there is space in its associated receive queue, but only the active port receive queue empties into the main queue.
- Aurora output data format for dedicated high-bandwidth serial transmission
- Aurora output format support at client interface
- Trace Memory port (64-bit address and data) can be used to write Nexus messages directly to external overlay memory. Block transfer mode provides highest-bandwidth option.
- Nexus out port for direct connection to pins (low-speed applications) or to another Nexus port controller
- Configurable Start Address and Range Selection for Trace Memory port write operation provides flexibility to choose Overlay memory Section and range to store Trace data.
- Both output ports may operate simultaneously. Filtering logic sorts messages to determine destination port.
- Partitionable main queue makes it possible to simultaneously service multiple output ports. It can also be used to devote more bandwidth to high-priority message-types or split trace stream between on-chip memory and trace port.
- Re-allocatable receive queues allow queue-space of an inactive client to be reassigned to an active client
- Stall detection and handling allows NAR to continue when a client stops sending data in the middle of a message
- Programmable Global stall signal on debug tool indication of memory overflow
- Event generation on External Trace Memory (overlay) programmable full
- Suppress mode can be used to shut down message delivery and relieve back-pressure to clients without turning off Nexus messaging in the clients.

6.4 Register definition

The NAR configuration registers provide configuration and control for all NAR operations. They are accessible through the JTAG (IEEE 1149.1) port. Addressing (within the local NAR address space) for the NAR registers is shown in the following table. If the NAR clock is turned off, the NAR registers cannot be accessed.

Table 6-1. NAR registers

Register name	Read/Write	JTAG index (NAR relative)
NAR_CR—NAR Control Register	R/W	0
NAR_ST—NAR Status Register	R	1
NAR_SFR—Source Filter Register	R/W	2
NAR_TFR—Type Filter Register	R/W	3
NAR_TBALO—Trace memory port base address, low	R/W	4
NAR_TBAHI—Trace memory port base address, high	R/W	5
NAR_TCR—Trace Memory bus control register	R/W	6
NAR_STCR—Suppress Trigger Control Registers 1–4	R/W	7 (Trigger 1) 8 (Trigger 2) 9 (Trigger 3) A (Trigger 4)
NAR_CSSR–Client Suppress Status Register	R	B
NAR_CDR—Client Disable Register	R/W	C
NAR_NAPCR—Nexus Aurora Phy Control Register	R/W	D
NAR_AHFPAR—Trace memory fill level and partition configuration	R/W	E

6.4.1 NAR control register (NAR_CR)

The NAR control register provides all the main control and enable bits for the NAR and its output ports. The following table shows the format of the NAR_CR.

Register index: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NEN	NCD	TEN	TRS	TSG		SED	SRD	TBW	AQP		0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WPE			0	0	MSP		0	ALT		0	AFA	0	FSD	PUS	NSR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_CR is described in the following table.

Table 6-2. NAR_CR field descriptions

Field	Description	
31	NAR Enable.	
NEN	0	NAR is disabled unless HBDP is independently enabled. See Disabled for more information.
	1	NAR is enabled.
30	NAR Communication Disable.	
NCD	0	NAR monitors clients for activity
	1	NAR communication with clients is disabled
29	Timestamp Enable.	
TEN	0	Timestamp packet is not included in messages
	1	Timestamp packet is included in messages from all client including NAR generated messages
28	Timestamp Reset.	
TRS	0 to 1 transition: reset timestamp counter in NAR	
27–26	Timestamp Granularity (Mode).	
TSG	**Note:** Some clients may only support timestamps on every message.	
	00	Add timestamp to every message
	01	Add timestamp to every 4th message
	10	Add timestamp to every 16th message
	11	Add timestamp to every 32nd message
25	Stall Error Disable.	
SED	0	NAR injects dummy complete message when active client stalls
	1	NAR does not inject dummy complete message
24	Stall Recovery Disable.	
SRD	0	Stalled receive queue resumes operation when its client comes back on line
	1	Stall error recovery is disabled
23	Tracebuffer Wrap.	
TBW	0	NAR does not capture new data when main queue is full
	1	NAR begins overwriting main queue from the beginning upon queue overflow
22–21	Alternate Queue Partition.	
AQP	00	Queue partitioning is disabled
	01	Reserved
	10	1/2 of queue is dedicated to alternate partition
	11	Reserved
20	Reserved and must be written with 0	
19–16	Reserved and must be written with 0s	
15–13	Watchpoint Enable	
WPE	000	NAR watchpoint messaging is disabled
	xx1	NAR generates a watchpoint message upon a sync event
	x1x	NAR generates a watchpoint message upon a trace memory bus or tracebuffer block full/wrap event
	1xx	NAR generates a watchpoint message upon a suppress-mode event
12	Reserved	

Table continues on the next page...

Table 6-2. NAR_CR field descriptions (continued)

Field	Description	
11	Reserved. This bit must be written with 0.	
10–9 MSP	Message Select Port. Defines which port is the target for message type/source filtering. Only active in two-port applications.	
	00	No message select port. 0–1 port is active.
	01	HBDP is the MSP
	10	Trace memory bus is the MSP
	11	Nexus is the MSP
8	Reserved	
7–6 ALT	Alternate Port. Defines which port is the target for non-filtered messages; main port in single-port applications.	
	00	No alternate port. Running in trace-buffer mode.
	01	HBDP is the ALT
	10	Trace memory bus is the ALT
	11	Nexus is the ALT
5	Reserved	
4 AFA	Automatic Flush Accumulators.	
	0	Output queue accumulators do not flush stranded beats
	1	Accumulators automatically flush stranded beats
3	Reserved. This bit must be written with 0.	
2 FSD	Frame Serial Data.	
	0	Framing disabled. Data is streamed through the Aurora out port.
	1	Framing enabled. Start/End frame symbol inserted between every 256 Aurora words.
1 PUS	Power-Up Suppressed.	
	0	NAR powers up active
	1	NAR powers up in global suppressed mode
0 NSR	NAR Soft Reset.	
	1	Resets all configuration registers and sets queue address pointers to the top of their respective queues. Also resets itself after a single-cycle delay. (This does not affect other blocks)

6.4.2 NAR status register (NAR_ST)

The NAR_ST register provides information on NAR activity and queue status, as well as HBDP status and errors. It is a read-only register. The following table shows the format of the NAR_ST.

Register index: 1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NEN	NUM			NCE				MPF	MLV		APF	ALV		0	0
W																

Table continues on the next page...

Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HPR	HPE	MPE	MOV	0	TOV	QWA									
W		w1c	w1c	w1c		w1c										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

The NAR_ST register is described in the following table.

Table 6-3. NAR_ST field descriptions

Field	Description	
31 NEN	NAR Enabled.	
	0	NAR is disabled and inactive
	1	NAR is enabled with clocks on
30–28 NUM	NAR Mode.	
	000	NAR is in disabled mode
	001	NAR is in global suppressed mode
	010	NAR is in communication-disabled mode
	011	NAR is in tracebuffer mode
	100	NAR is in active transmit mode
	101	Reserved
	110	Reserved
	111	NAR is in run-control mode
27–24 NCE	NAR Config Error. See Table 6-18 for coding	
23 MPF	MSP Partition Full.	
	0	MSP partition not full
	1	MSP partition full
22–21 MLV	MSP Partition Queue Level. This field is available only when both output ports are enabled i.e. the main queue is partitioned.	
	00	0% < queue level < 25%
	01	25% < queue level < 50%
	10	50% < queue level < 75%
	11	75% < queue level <100%
20 APF	ALT Partition Full.	
	0	ALT partition not full
	1	ALT partition full
19–18 ALV	ALT Partition Queue Level.	
	00	0% < queue level < 25%
	01	25% < queue level < 50%
	10	50% < queue level < 75%
	11	75% < queue level <100%

Table continues on the next page...

Table 6-3. NAR_ST field descriptions (continued)

Field	Description	
17–16	Reserved	
15 HPR	HBDP Ready.	
	0	HBDP (Aurora Out port) is not ready
	1	HBDP is ready and accepts data for transmission
14 HPE	HBDP Error.	
	0	No HBDP error
	1	Error condition exists on HBDP
13 MPE	Trace memory bus Error.	
	0	No Trace memory bus error
	1	Error condition detected on trace transaction
12 MOV	Trace memory bus block overflow.	
	0	No Trace memory bus overflow
	1	Targeted external memory block is full. Trace memory bus writes are halted (or wrapped).
11	Reserved	
10 TOV	Tracebuffer Overflow.	
	0	No tracebuffer overflow
	1	The tracebuffer has overflowed/wrapped at least once
9–0 QWA	Queue Write Address. Current value of the output queue write pointer of ALT partition (last valid write + 1)	

6.4.3 Message filtering registers

The two output ports (HBDP and trace memory bus) are independently controllable. In the BD NAR, two ports can be active simultaneously. When multiple ports are active, user-programmable filters are determine which port receives a message, based on the message source or type. Filters apply to the designated Message Select Port (MSP), and are used to select messages for that port. All other messages go to the other (alternate) port. If multiple ports are used, an MSP must be designated; otherwise a configuration error results. See Output arbitration, for details on how the NAR arbitrates between multiple output ports. These registers should only be set in the BD NAR because partitioning is not supported in the PD NAR.

6.4.3.1 Source filter register (NAR_SFR)

The MSP can be configured to accept or exclude messages from up to four sources. The sources which are to be filtered are programmed in the source filter register (NAR_SFR) as shown in the following table. This register should only be set in the BD NAR because partitioning is not supported in the PD NAR.

Register index: 2

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFI	FE0	SF0				0	0	0	FE1	SF1				0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	FE2	SF2				0	0	0	FE3	SF3				0	ACC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_SFR is described in the following table. The source trace IDs, that are compared to the source filter fields (SFn), are device specific. For a list of the clients that are connected to the NAR and their corresponding Nexus source trace IDs, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 6-4. NAR_SFR field descriptions

Field	Description
31 SFI	Source Filter Invert. 0 Match if message source equals any of the enabled filters 1 Match if message source equals none of the enabled filters
30 FE0	Enable Filter 0. 0 Filter 0 disabled. SF0 ignored 1 Filter 0 enabled and contained in SF0
29–26 SF0	Source Filter 0. The message source trace ID is compared with this field. See Output arbitration for an example.
25–23	Reserved.
22 FE1	Enable Filter 1. 0 Filter 1 disabled. SF1 ignored 1 Filter 1 enabled and contained in SF1
21–18 SF1	Source Filter 1. The message source trace ID is compared with this field. See Output arbitration for an example.
17–15	Reserved

Table continues on the next page...

Table 6-4. NAR_SFR field descriptions (continued)

Field	Description
14 FE2	Enable Filter 2. 0 Filter 2 disabled. SF2 ignored 1 Filter 2 enabled and contained in SF2
13–10 SF2	Source Filter 2. The message source trace ID is compared with this field. See Output arbitration for an example.
9–7	Reserved
6 FE3	Enable Filter 3. 0 Filter 3 disabled. SF3 ignored 1 Filter 3 enabled and contained in SF3
5–2 SF3	Source Filter 3. The message source trace ID is compared with this field. See Output arbitration for an example.
1	Reserved
0 ACC	OR source and type filters. 0 Match if source AND type filters match 1 Match if source OR type filters match

6.4.3.2 Message type filter register (NAR_TFR)

The MSP can be configured to accept or exclude messages of up to four types, as encoded in the message TCODE field. The TCODEs which are to be filtered are programmed in the message type filter register (NAR_TFR) as shown in the following table. This register should only be set in the BD NAR because partitioning is not supported in the PD NAR.

Register index: 3

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TFI	FE0	TF0						0	FE1	TF1					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	FE2	TF2						0	FE3	TF3					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_TFR is described in the following table.

Table 6-5. NAR_TFR field descriptions

Field	Description
31 TFI	Type Filter Invert. 0 Match if message TCODE equals any of the enabled filters 1 Match if message TCODE equals none of the enabled filters
30 FE0	Enable Filter 0. 0 Filter 0 disabled. TF0 ignored 1 Filter 0 enabled and contained in TF0
29–24 TF0	Type Filter 0. The message TCODE is compared with this field. See Output arbitration for an example.
23	Reserved
22 FE1	Enable Filter 1. 0 Filter 1 disabled. TF1 ignored 1 Filter 1 enabled and contained in TF1
21–16 TF1	Type Filter 1. The message TCODE is compared with this field. See Output arbitration for an example.
15	Reserved
16 FE2	Enable Filter 2. 0 Filter 2 disabled. TF2 ignored 1 Filter 2 enabled and contained in TF2
13–8 TF2	Type Filter 2. The message TCODE is compared with this field. See Output arbitration for an example.
7	Reserved
6 FE3	Enable Filter 3. 0 Filter 3 disabled. TF3 ignored 1 Filter 3 enabled and contained in TF3
5–0 SF3	Type Filter 3. The message TCODE is compared with this field. See Output arbitration for an example.

6.4.4 Trace memory bus configuration registers

The trace memory bus can be used to transfer debug messages out to a dedicated block of external memory space. The trace memory bus configuration registers are used to specify the location and size of the external debug space.

6.4.4.1 Trace memory bus base address registers (NAR_TBALO and NAR_TBAHI)

The trace memory bus base address registers determine where Nexus data sent out through the trace memory bus goes. The NAR sends out blocks of data on the trace memory bus, writing to an address block defined by the specified base address. Data can transmitted either on an as-available, word-wide basis, or in larger blocks. In block transfer mode, the size of the data block is determined by the BTM field of the Control Register; the default is 64 bytes. Even if block transfer mode is not used, the 5, 6, 7, or 8 (depending on the BTS field in NAR_TCR register) LSBs of the base address are not accessible. This forces the debug memory space to be aligned to a block boundary.

When read, the address registers return the current address, allowing the user to monitor the progress of a trace memory transfer. The following tables show the format of the trace memory base address registers (NAR_TBALO and NAR_TBAHI). For a description of the allowed address for these registers, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Register index: 5

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBAHI																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Register index: 4

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TBALO																						ADRLO[9]	ADRLO[8]	ADRLO[7]	ADRLO[6]	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_TBALO and NAR_TBAHI registers are described in the following tables.

Table 6-6. NAR_TBAHI field descriptions

Field	Description
31–0 TBAHI	Trace Debug Memory Base Address on AHB-EW port, upper part. For 32-bit address applications this field is all zeros.

Table 6-7. NAR_TBALO field descriptions

Field	Description
31–10 TBALO	Trace Debug Memory Base Address on AHB-EW port, lower part
9 ADRLO[9]	Base Address LSB for BTS = 11 (384-byte block)
8 ADRLO[8]	Base Address LSB for BTS = 10 (256-byte block)
7 ADRLO[7]	Base Address LSB for BTS = 01 (128-byte block)
6 ADRLO[6]	Base Address LSB for BTS = 00 (64-byte block)
5–0	Reserved

6.4.4.2 Trace memory bus control register (NAR_TCR)

The trace memory bus control register is used to control the attributes of the trace memory bus port, such as block transfer size and message priority. Since the trace memory bus transfers are expected to go to a static block of external memory, a maximum transfer size must be specified to prevent the NAR from overwriting memory it is not authorized to access. This is done via the MXFR max transfer size register field. The memory allocation is specified in units of the trace memory bus block transfer size (64, 128, 256, or 384 bytes), so the actual size of the reserved memory block depends on the BTS setting. The following table shows the format of the NAR_TCR.

Register index: 6

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BTM	BTS		BTH		0	0	0	TBW	MXFR						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MXFR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_TCR is described in the following table.

Table 6-8. NAR_TCR field description

Field	Description	
31	Block Transfer Mode.	
BTM	0	Data for the Trace Memory bus port is transmitted one word at a time, upon first availability
	1	Data to the Trace Memory bus port is bundled into blocks before transmitting
30–29	Block Transfer Size.	
BTS	00	Block transfer size = 64 bytes
	01	Block transfer size = 128 bytes
	10	Block transfer size = 256 bytes
	11	Block transfer size = 384 bytes
28–27	Block Transfer Threshold.	
BTH	00	Full threshold. Number of entries in partition must equal block transfer size before block transfer starts
	01	Begin transfer when block is 25% full
	10	Begin transfer when block is 50% full
	11	Begin transfer when block is 75% full
26–24	Reserved. These bits must be written with 0.	
23	Trace Memory Bus Block Wrap.	
TBW	0	NAR stops writing to the Trace Memory bus debug memory space once Max Size has been reached
	1	NAR overwrites Trace Memory bus debug memory from the beginning upon Trace Memory bus block overflow.
22–0	Trace Memory Bus Max Transfer Size.	
MXFR	Total size of the dedicated debug memory space, in Trace Memory bus data transfer blocks.	

6.4.5 Suppress mode triggers registers (NAR_STCR)

Four suppress mode triggers are provided for precise control of the NAR suppress mode feature described in Suppress mode. The suppress mode triggers are only available on the PD NAR and do not apply to the BD NAR. The registers are present in the BD NAR, but input suppress triggers are not connected. Each trigger can be programmed to suppress or wake clients individually or the entire NAR. All receive queues can be suppressed on the occurrence of a selected trigger, or the full NAR can be brought out of suppress mode upon the occurrence of the appropriate trigger. A trigger can also be programmed to toggle one or more clients between the suppressed and active states. Each Suppress Trigger Control Register, shown in the following table, is associated with one of the triggers, and are used to determine the action taken, if any, upon the assertion of that trigger. The suppress triggers are connected to the Sequence Processing Unit (SPU) action unit.

Register index: 7-A

Table continues on the next page...

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LST															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_STCR are described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the local suppress trigger, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 6-9. NAR_STCR field descriptions

Field	Description	
31 GST	Global Suppress Trigger.	
	0	The associated trigger cannot cause NAR to enter global suppress mode
	1	The associated trigger is enabled for global suppress mode
29–16	Reserved	
15–0 LST	Local Suppress Trigger.	
	0000000000000000	The associated trigger is disabled for local suppress mode for all clients
	xxxxxxxxxxxxxxx1	Trigger is enabled for Client 1 local suppress
	xxxxxxxxxxxxxx1x	Trigger is enabled for Client 2 local suppress
	xxxxxxxxxxxxx1xx	Trigger is enabled for Client 3 local suppress
	xxxxxxxxxxxx1xxx	Trigger is enabled for Client 4 local suppress
	xxxxxxxxxxx1xxxx	Trigger is enabled for Client 5 local suppress
	xxxxxxxxxx1xxxxx	Trigger is enabled for Client 6 local suppress
	xxxxxxxxx1xxxxxx	Trigger is enabled for Client 7 local suppress
	xxxxxxxx1xxxxxxx	Trigger is enabled for Client 8 local suppress
	xxxxxxx1xxxxxxxx	Trigger is enabled for Client 9 local suppress
	xxxxxx1xxxxxxxxx	Trigger is enabled for Client 10 local suppress
	xxxxx1xxxxxxxxxx	Trigger is enabled for Client 11 local suppress
	xxxx1xxxxxxxxxxx	Trigger is enabled for Client 12 local suppress
	xxx1xxxxxxxxxxxx	Trigger is enabled for Client 13 local suppress
	xx1xxxxxxxxxxxxx	Trigger is enabled for Client 14 local suppress
	x1xxxxxxxxxxxxxx	Trigger is enabled for Client 15 local suppress
	1xxxxxxxxxxxxxxx	Trigger is enabled for Client 16 local suppress
	All other options	Reserved

6.4.6 Client suppress status register (NAR_CSSR)

Clients can be individually suppressed, leaving the rest of the NAR unaffected. When enabled for local suppress, assertion of one of the suppress triggers causes one or more client receive queues to stop queueing messages. A subsequent pulse of the same trigger causes the client(s) to awaken. Local suppress mode is enabled by setting the appropriate bit in one of the NAR_STCRs. An additional status register, Client Suppress Status (NAR_CSSR), shown in the following table, can then be read to determine which clients are suppressed, and which are active.

Register index: B

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LSS																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_CSSR is described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the local suppress status, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 6-10. NAR_CSSR field descriptions

Field	Description	
31–16	Local Suppress Status.	
LSS	0000000000000000	No client is in local suppress mode
	xxxxxxxxxxxxxxx1	Client 1 is suppressed
	xxxxxxxxxxxxxx1x	Client 2 is suppressed
	xxxxxxxxxxxxx1xx	Client 3 is suppressed
	xxxxxxxxxxxx1xxx	Client 4 is suppressed
	xxxxxxxxxxx1xxxx	Client 5 is suppressed
	xxxxxxxxxx1xxxxx	Client 6 is suppressed
	xxxxxxxxx1xxxxxx	Client 7 is suppressed
	xxxxxxxx1xxxxxxx	Client 8 is suppressed
	xxxxxxx1xxxxxxxx	Client 9 is suppressed
	xxxxxx1xxxxxxxxx	Client 10 is suppressed
	xxxxx1xxxxxxxxxx	Client 11 is suppressed
	xxxx1xxxxxxxxxxx	Client 12 is suppressed
	xxx1xxxxxxxxxxxx	Client 13 is suppressed
	xx1xxxxxxxxxxxxx	Client 14 is suppressed
	x1xxxxxxxxxxxxxx	Client 15 is suppressed
	1xxxxxxxxxxxxxxx	Client 16 is suppressed
	All other options	Reserved
15–0	Reserved	

6.4.7 Receive queue client disable register (NAR_CDR)

Disabling a client reallocates its receive queue to the next active client, allowing that memory space to be put to use. The Client Disable Register, shown in the following table, allows the user to individually disable or enable clients as needed. The status bits also inform the user when the client has new data available.

Note

Even though the BD NAR is described as only having one client, there is actually a second dummy client, client 2. To re-allocate the dummy RX queue to client 1 (the PD NAR), set the BD NAR client 2 RQD bit.

Register index: C

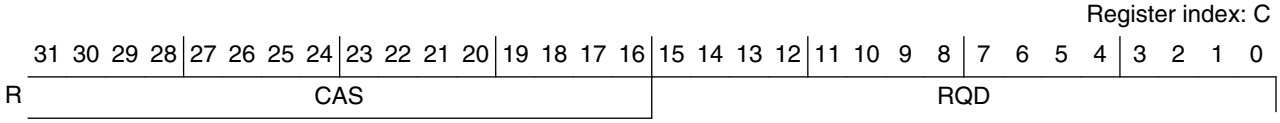
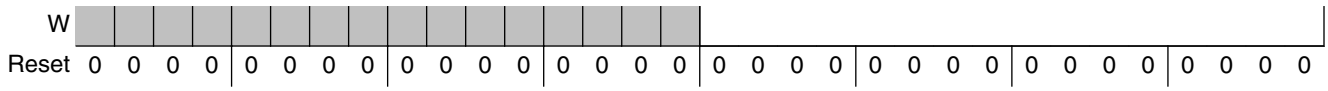


Table continues on the next page...



The NAR_CDR is described in the following table. The set of clients connected to the NAR is device specific. For a list of the clients that are connected to the NAR and their corresponding NAR client index for the client activity status and receive queue disable, see the Calibration and Debug configuration chapter which describes how the modules are configured.

Table 6-11. NAR_CDR field descriptions

Field	Description	
31–16	Client Activity Status.	
CAS	xxxxxxxxxxxxxxx1	Client 1 is active with new data
	xxxxxxxxxxxxxx1x	Client 2 is active with new data
	xxxxxxxxxxxxx1xx	Client 3 is active with new data
	xxxxxxxxxxxx1xxx	Client 4 is active with new data
	xxxxxxxxxxx1xxxx	Client 5 is active with new data
	xxxxxxxxxx1xxxxx	Client 6 is active with new data
	xxxxxxxxx1xxxxxx	Client 7 is active with new data
	xxxxxxxx1xxxxxxx	Client 8 is active with new data
	xxxxxxx1xxxxxxxx	Client 9 is active with new data
	xxxxxx1xxxxxxxxx	Client 10 is active with new data
	xxxxx1xxxxxxxxxx	Client 11 is active with new data
	xxxx1xxxxxxxxxxx	Client 12 is active with new data
	xxx1xxxxxxxxxxxx	Client 13 is active with new data
	xx1xxxxxxxxxxxxx	Client 14 is active with new data
	x1xxxxxxxxxxxxxx	Client 15 is active with new data
	1xxxxxxxxxxxxxxx	Client 16 is active with new data
	All other options	Reserved

Table continues on the next page...

Table 6-11. NAR_CDR field descriptions (continued)

Field	Description	
15–0	Receive Queue Disable.	
RQD	xxxxxxxxxxxxxxx1	Disable receive queue for Client 1
	xxxxxxxxxxxxxx1x	Disable receive queue for Client 2
	xxxxxxxxxxxxx1xx	Disable receive queue for Client 3
	xxxxxxxxxxxx1xxx	Disable receive queue for Client 4
	xxxxxxxxxxx1xxxx	Disable receive queue for Client 5
	xxxxxxxxxx1xxxxx	Disable receive queue for Client 6
	xxxxxxxxx1xxxxxx	Disable receive queue for Client 7
	xxxxxxxx1xxxxxxx	Disable receive queue for Client 8
	xxxxxxx1xxxxxxxx	Disable receive queue for Client 9
	xxxxxx1xxxxxxxxx	Disable receive queue for Client 10
	xxxxx1xxxxxxxxxx	Disable receive queue for Client 11
	xxxx1xxxxxxxxxxx	Disable receive queue for Client 12
	xxx1xxxxxxxxxxxx	Disable receive queue for Client 13
	xx1xxxxxxxxxxxxx	Disable receive queue for Client 14
	x1xxxxxxxxxxxxxx	Disable receive queue for Client 15
	1xxxxxxxxxxxxxxx	Disable receive queue for Client 16
	All other options	Reserved

6.4.8 Nexus Aurora Phy control register (NAR_NAPCR)

The Nexus Aurora Phy control register provides configuration control to number of HBDP lanes of Nexus Aurora Phy and LVDS pad control signals. The following table shows the format of the NAR_NAPCR.

Register index: D

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	NUM_LANE			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0		0	LVDS_TXOPT				LVDS_RX_TERM_EN	LVDS_BIAS_PRG	LVDS_RXOPT					
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

The NAR_NAPCR is described in the following table.

Table 6-12. NAR_NAPCR field descriptions

Field	Description
19–16 NUM_LANE	TX Lane Configuration. Number of enabled TX lanes. This field is device specific. For a description of this field see the Calibration and Debug configuration chapter which describes how the modules are configured.
11–8 LVDS_TXOPT	This is an internal characterization field.
7 LVDS_RX_TERM_EN	This is an internal characterization field.
6 LVDS_BIAS_PRG	This is an internal characterization field.
5–0 LVDS_RXOPT	This is an internal characterization field.

6.4.9 Trace Memory fill level and partition configuration register (NAR_AHFPAR)

This register provides the configuration to set the fill level marker for debug trace memory, connected at the trace memory bus, to generate the partial full event indication and the partition related configuration for main queue (internal central queue). The following table shows the format of the NAR_AHFPAR.

Register index: E

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	GSE	0	0	0	GLOBAL_STALL_COUNT				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0				

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	AHB6 5kEN	AHB0 1kEN	PART_CFG		AHB_65K_Marker				AHB_01K_Marker			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NAR_AHFPAR is described in the following table.

Table 6-13. NAR_AHFPAR field description

FIELD	Description
31-25	Reserved and must be written with 0s

Table continues on the next page...

Table 6-13. NAR_AHFPAR field description (continued)

FIELD	Description
24 GSE	Global Stall Enable. Setting this bit enables the NAR to act on dci_evti[1] event to generate a global stall
23-21	Reserved and must be written with 0s
20-16 GLOBAL_STALL_COUNT	Global Stall Count. Value at this field determines how many cycles, in the NAR's operating clock, the global stall remains asserted on a dci_evti event. This is a five bit field so the maximum value is 31 cycles. See Global (Development tool client) stall control
15-12	Reserved and must be written with 0s
11 AHB65kEN	Enable 65K Marker Event
10 AHB01kEN	Enable 1K Marker Event
9-8 PART_CFG	Partition Configuration. 00 No partition 01 Reserved 10 50% partition (in two partition two configuration case) 11 Reserved
7–4 AHB_65K_Marker	Value at this field is used to compare a 4-bit slice of internal Trace Memory address bus which works at boundary of 65K. A match with this value generates an event to SPU to indicate the fill status of External Overlay Trace memory. See Overlay and internal trace memory full status generation **Note:** This is used in BD as it has memory ranges from 65K to 983K
3–0 AHB_01K_Marker	Value at this field is used to compare a 4-bit slice of internal Trace Memory address bus which works at boundary of 1K. A match with this value generates an event to SPU to indicate the fill status of External Overlay Trace memory. See Overlay and internal trace memory full status generation **Note:** This is used in PD as it has memory ranges from 1K to 15K

6.5 Functional description

High-level descriptions of the operation of the NAR and its main functional blocks are given in the following sections.

6.5.1 Reset/Startup

Assertion of global reset causes the NAR operation to immediately cease. All queues and configuration registers are reset. Upon exiting reset, the Nexus Port Controller (NPC) block inside the NAR is disabled. Holding TMS high for five consecutive rising edges of TCK puts TAP controller into the Test-Logic-Reset state regardless of the current state.

This applies reset to only TAP controller and JTAG related logic. Applying soft reset through the NAR_CR resets the entire NAR including the TAP controller and configuration registers.

The typical startup procedure for running a trace through the NPC is as follows:

1. Reset the block
2. Upon exiting reset, the Aurora Phy configuration is determined by sampling the configuration pin values from the Nexus Aurora Phy (NAP). If the NAP is enabled, then the NAR is enabled. The Aurora communication is started only when the NAP is ready to accept data. (If any debug lanes are enabled, the Aurora Link begins training at this time.)
3. Debugger/external user sets the NAR enable (NEN) bit in the NAR_CR and configures the NAR. The configuration includes setting up message selection/ filtering when two output ports are enabled, enabling timestamp and so on.
4. User programs additional device level debug facilities, such as event-processing, performance-monitoring, and so on.
5. User sets up trace filtering.
6. Enable Nexus transmission in clients.

To enable NAR operation, either the Aurora Link must be enabled at reset, or the user must set the NEN bit in the NAR control register.

Note

The HBDP may already be active by the time the NAR comes up. The NAR only tries to initialize the HBDP if it is currently inactive.

The NAR can be enabled to begin immediate operation, or to start in suppress mode, see Suppress mode. Once the NAR and desired ports are enabled, the user should wait until the NAR-ready and HBDP port-ready status bits are set in the NAR status register before beginning Nexus messaging in the clients.

6.5.2 Operation modes

The NAR operating mode is determined by the state of the main control signals: reset and Phy enable. In its enabled state, operation is controlled by its configuration registers. A truth table for all the NAR states is given in the following table.

Table 6-14. NAR operational modes truth table

Reset	NAR enable	NAR comm disable	Global suppress	HBDP enable	ALT partition	MSP partition	Mode
0	x	x	x	x	x	x	RESET
1	0	x	x	0	x	x	Disabled
1	1	1	x	0	x	x	Trace-Disable
1	x	1	x	1	x	x	Trace-Disable
1	0	x	x	1	x	x	Trace-Disable
1	1	0	1	x	x	x	Suppressed
1	1	0	0	x	Trace Memory bus	x	NAR Transmit
1	1	0	0	1	AUR	x	NAR Transmit
1	1	0	0	1	Trace Memory bus	AUR	NAR Transmit
1	1	0	0	1	AUR	Trace Memory bus	NAR Transmit

There can be overlap between modes as modes are not mutually exclusive.

6.5.2.1 Reset

During reset all internal resources and queues are reset and all output ports are disabled. All inputs are ignored. Upon exiting reset mode, the NAR is in a fully disabled state.

6.5.2.2 Disabled

In the disabled state, all message traffic is ignored, and all output ports are disabled. The NAR is disabled if and only if the NAR enable bit of the NAR control register is not set and the Aurora Link is inactive.

6.5.2.3 Suppressed

The suppressed state differs from the disabled state in that the NAR is on. The NAR pulls out any messages that come up at the clients, keeping the Message Start-End Out (MSEO) bits (so that it can keep track of message boundaries) and discarding the rest.

In the suppressed state, the NAR is waiting for a trigger to begin normal operation. Once that trigger is received, the NAR begins queueing messages that started after the trigger arrived. Incomplete messages that were in transit to the NAR when the trigger arrived are discarded. The NAR can be put into suppressed mode by enabling one of the global suppress states.

6.5.2.4 Trace disable

In trace disable mode, the NAR is powered up but all client-trace is disabled, and no messages are pulled. This mode is also useful for port initialization (to prevent any activity from occurring before the outputs are ready) or for debug/trace-buffer mode, to ensure that, during data retrieval, the NARs queues are stable.

The NAR goes into trace disable (communication-disable) mode when the NAR-enable and NAR communication-disable bits are set or when the HBDP channel is active while the NAR-enable bit is not set.

6.5.2.5 NAR transmit

This is the standard operational mode of the NAR when it is transferring Nexus messages from multiple clients to one or two enabled output port(s). The NAR performs four main operations as described in the following table on the incoming messages before transmitting to one of the output ports.

Table 6-15. NAR transmit operations

Operation	Description
Data reception	NAR accepts data from each client based on the status of the client receive queue status. The maximum data rate is 8 × 32-bit data words (where 8 is the number of Nexus message beats in a full entry and 32 is the number of bits in the Nexus word) per cycle per client. MSEO bits of incoming data are monitored to identify end-of-message boundaries.
Input Arbitration	Active receive queue unloads into main queue(s), 8 × 32 bits per cycle (where 8 is the number of Nexus message beats in a full entry and 32 is the number of bits in the Nexus word). If end-of-message detected, new active receive queue is determined based on client request level or round-robin algorithm.
Output Arbitration	If multiple output ports are enabled, route messages from top of queue to appropriate output port based on message type or source. If split queue is used, outputs can run simultaneously, with each sub-queue communicating directly with a port.
Translation	Data going out the Aurora or trace memory bus ports must be translated into the correct format prior to transmission.

Enabling the NAR and one or more output ports without setting the communication-disable bit puts the NAR into transmit mode.

6.5.3 Data reception and arbitration

The BD NAR supports 1 client (although a second dummy client is used to increase the RX buffer size for client 1). Each client has a dedicated input port on the NAR and communicates with its associated input port via a dedicated data bus. A data transmission must consist of an integer number of 32-bit Nexus words (32-bit word is called a beat). The number of beats varies from client to client. The number of beats (Bn) for each client are described in the Calibration and Debug configuration chapter which describes how the modules are configured.

There is one type of client supported at NAR client interface. One Nexus word consists of 30 Message Data Out (MDO) bits plus two Message Start/End Out (MSEO) control bits.

When new data is available in the core, it is transferred over to the asynchronous gasket in the internal NAR clock domain. The gasket then asserts enough data valids to fill internal receive queue, using a receive queue available signal to track available space in the associated queue, under the assumption that the receive queue available signal is initialized with the size of the queue. If the internal arbiter consumes any of its queued entries, it responds with a data accept signal, which increments the receive queue available signal and allows the gasket to send the next piece of data. Under certain circumstances, like receive queue reallocation, it is possible for the internal arbiter to send out surplus data accept signals such that the receive queue available signal can become larger than the size of the queue. If the receive queues are sized correctly, when NAR is actively servicing a particular client, then asserted data accept signals allow the client to continually stream data until the NAR switches to a new client. The data tracking at the gasket assures that it never sends more data than the receive queue can accept and inactive queues are always full and ready for when they are switched to.

The client gaskets must always be in sync with the internal logic of NAR in terms of the number of available entries in each receive queue. For this reason, the internal NPC and its client gaskets need to be reset by the same signal. Any soft-reset of the NAR (via the SRESET configuration bit) is communicated to all gaskets and used to reset every thing inside it.

Each input port has a dedicated receive queue with a minimum of two entries, where an entry is wide enough to store an entire Bn × 32 data word (where 32 is the number of bits in the Nexus word or the beat). The receive queue can be made larger to allow the NAR to continue transmitting during switches from one client to another in a pipelined environment. Every cycle, if there is space in a particular receive queue, it latches in one full AUX word from its client. Because the client may be running at a much higher frequency than the NAR, several cycles worth of Nexus messages are contained in a single transaction, as shown in the following figure for a 3-beat per word transaction.

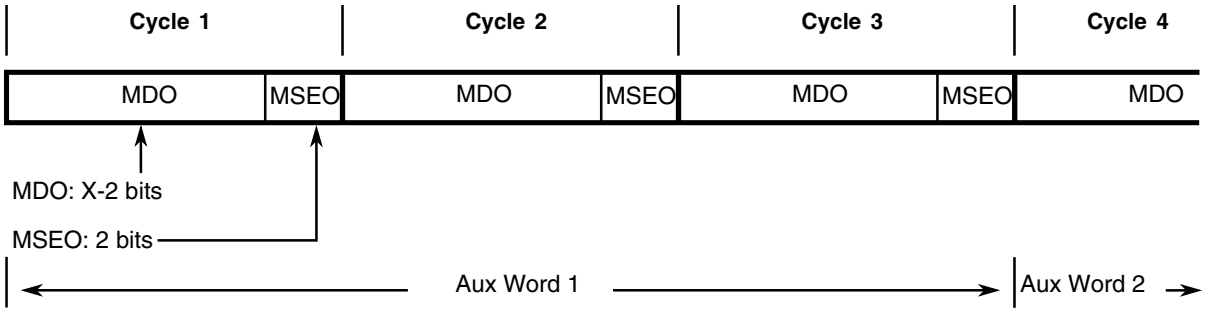


Figure 6-2. AUX bus parsing

Each AUX word consists of Bn cycles of Nexus data, along with an optional request-level sideband signal consisting of a priority field (bit 0) and a weight field (bits 1 and 2). Request level priority indicates to the NAR the clients queue status, allowing it to give preferential treatment to clients that are closer to queue overflow. Clients with a set priority bit are always chosen ahead of clients without the bit. For this reason, priority should be used sparingly, only when a client is in danger of overflowing. Request-level weighting also gives a client preferential treatment, but without entirely shutting out the other clients. Normally, the NAR attempts to switch to a new client every time it sees an end-of-message boundary.

As the AUX word is pulled into the receive queue, the MSEO bits are monitored to determine the current Nexus state of the data flow; since Nexus is a variable-length protocol, messages can end in any cycle. If the start message or end message state is detected on any data beat, the beat is marked as containing an end-of-message boundary.

Any received beat which is in the idle state, indicating null data content, is discarded (see the IEEE-ISTO 5001-2003 standard for details of the Nexus protocol). At the same time, if two outputs are enabled, then the source and TCODE fields are extracted from each new message and compared with the user-programmed filters. Messages that generate a match are tagged with an MSP bit, which allows the NAR to route it to the correct port.

Every cycle, the NAR transfers 8 beats of data from the receive queues into the main output queue. Only one port (the active port) is allowed to empty into the main queue at a time: the active port unloads its receive queue into the main queue until the end of the current Nexus message is detected. At that point, a new active port is selected based on the latest request level received at each port. In the case where multiple ports have the same request level (or when request level is not used), then the active port is rotated amongst those ports. If the end of message occurs in the middle of an AUX word, then all the beats up to the end of message are transferred from the old active port, while the rest of the beats come from the new active port. For example, if the message ended at beat two of a three-beat AUX word, then the first two beats into the main queue would come from the original active port, while the last beat would come from the new active port.

6.5.4 Suppress mode

The suppress mode feature enables the user to quickly turn Nexus messaging through the NAR on or off, based on the four dedicated suppress-mode triggers. Suppress mode is only supported in the PD NAR, it is not supported in the BD NAR. A suppress can either be global, meaning messaging is turned off for all clients, or local, which only applies to selected clients. In suppress mode, the NAR pulls messages from the suppressed client(s), but does not queue or deliver them. After extracting the MSEO bits to keep track of message boundaries, the rest of the message is discarded. This allows the NAR to better keep up with fast-running Nexus clients without having to turn off messaging in the client.

Typically, a client would be suppressed during less interesting periods, and would be awakened when some preset condition is detected. This is communicated to the NAR via the one of the suppress triggers, which can be sourced by a client or an external Event Processing block. When an enter suppress trigger is received, the NAR output queue finishes servicing the current active ports until the end of the message. If the end of the message occurs in a middle beat of a Nexus word, then the remainder of the word is padded with null messages. Once the last message has been loaded into the output queue, the receive queues are all emptied. New messages are pulled from clients as they appear, go through their respective receive queues and then are discarded. In this way, the NAR can naturally monitor the MSEO bits and keep track of each client's Nexus state. In suppress mode, the output ports remain active until the output queue is emptied, meaning there is a delay between the assertion of the suppress trigger and the end of messaging.

When the NAR is in suppress mode and receives a wake-up trigger, it continues discarding any partial messages still in the receive queues. Any message that begins after the trigger is queued normally, insuring that only complete messages are sent to the output queue.

Global suppress mode applies to all clients. Any of the suppress triggers can be programmed to cause the NAR to toggle between suppressed and awake by setting the GST bit of the associated Suppress Trigger Control Register (NAR_STCR). Additionally, the NAR starts up in global suppress mode if the PUS bit of the Control Register is set when the NAR is enabled. All clients are then suppressed until a wake-up trigger is received. If the PUS bit is set without enabling any of the triggers for global suppress, a configuration error results and the NAR powers up in active mode.

Clients can go also into local suppress mode without putting the entire NAR to sleep. Each NAR_STCR has a 16-bit LST field. Setting one or more of these bits causes the trigger assigned to that NAR_STCR to toggle the associated clients between suppressed and awake. All other clients are unaffected. Thus, suppress trigger one can be assigned to

one set of clients, while suppress trigger two can be assigned to another set of clients, and different parts of the device can be ignored at different times, all based on different, programmable events. Global suppress always has higher priority than local suppress. That is, if the NAR is in global-suppressed mode, any local suppress wake-up triggers are ignored. Also, if some clients are in local suppress when a global suppress trigger hits, then all go into suppress mode; however, upon the subsequent global wake-up, the clients that were previously in local suppress remain so.

6.5.5 Stall detection

Since the NAR can only switch between clients on an end-of-message boundary, every message that goes into the NAR must complete. If a client happens to die or stall in the middle of transmitting a message, this uncompleted message could cause the NAR to stall, at a time when other clients might still be able to deliver useful information. To prevent this from occurring, the NAR monitors each client for a stall condition, which is defined as occurring when data is not valid in the middle of a Nexus message for a set number of cycles.

When a stall condition is detected, the NAR can do one thing. If the client stalls, then the NAR must complete the current stalled message to continue operating. This scenario is termed a stall error. On a stall error, the NAR injects a dummy message beat into the message stream after the last valid beat of data received from the stalled client. The dummy message is a single beat injected at the output of the receive queue into the main queue. The MSEO bits of the data payload are set to 2b11 regardless of the previous Nexus state. This forces the dummy message into the End Message state. When the NAR injects a dummy message, it also generates an error message to notify the user what has happened.

Injecting a dummy message is an intrusive act, which alters the data stream from the client. If a stalled client is thought likely to recover cleanly, it might be advisable to suppress stall errors to prevent the data stream from being corrupted. This can be done by setting the SED bit of the configuration register, which carries the risk of a truly dead client causing the NAR to hang. Otherwise, the default operation is for the NAR to inject the dummy message and then monitor the client for signs of recovery. If the client does come back online, the NAR attempts to wait until it sees an end-of-message boundary before capturing new data. At best, if the client recovers smoothly, only the interrupted message is lost; if the client does not recover smoothly, it might not be possible for the client and NAR to get back in sync in terms of their Nexus states, which would lead to nothing but garbled data coming out. For this reason, it is also possible to prevent the

NAR from trying to recover from a stall error, by setting the SRD bit of the NAR_CR. If the SRD bit is set, a stalled client is not allowed to recover until the SRD bit is unset or the NAR is reset.

6.5.6 Output arbitration

The NAR allows two output ports to be active at the same time. This feature is only available on the BD NAR. Typically, this feature is only useful if one of the ports is used to service messages of a specific type and/or messages from a particular client, while the other port supports all the rest of the message traffic. For example, the NAR could be set up to send in-circuit trace (ICT) traffic out the HBDP and to reserve the trace memory bus port for all other Nexus messages.

If two output ports are enabled, one of them must be designated as the Message Select Port (MSP). The MSP is the destination for those messages that satisfy a set of user-defined requirements as to the source and type of message. All other traffic is sent to the other enabled port. Even though the NAR supports three outputs (HBDP, Nexus, and trace memory bus) only two can be used at the same time. Ports are enabled by setting the port enable bits in the BD NAR_CR. The MSP is designated by setting bits [10–9] and ALT is designated by setting bits [7–6]. Once the MSP has been specified, other bits in the NAR_CR, along with the Source Filter Register (NAR_SFR) and Type Filter Register (NAR_TFR), are used to determine which messages go to the MSP and which do not. These facilities can be used to build a logical expression that can filter messages based on up to four types and/or four sources.

The NAR_SFR can be programmed with up to four source codes, while the NAR_TFR can hold up to four message-type codes. Each filter also includes an enable bit to turn it on or off. A source match occurs if a messages source code matches one of the source filters or (if the SFI bit is set) if it does not match any of the source filters. Similarly, a type match occurs if a messages type code matches one of the type filters or (if the TFI bit is set) if it does not match any of the type filters. Finally, the ACC bit of the NAR_SFR determines when both source and type filters are used, whether the two are ANDed together (ACC = 0) or ORed together (ACC = 1). For example, to send ICT messages out the HBDP and all other traffic to the trace memory bus port:

- Set ALT = 10 in the NAR_CR—sets trace memory bus port as the alternate port
- Set MSP = 01 in the NAR_CR—sets HBDP as the message select port
- Set FE0 = 1 & TF0 = 100010 (0x22) in the NAR_TFR—enables type filter 0 to match ICT messages

To send everything but data trace from source 13 to the trace memory bus port, while all other traffic goes to the HBDP:

- Set ALT = 01 in the NAR_CR—sets HBDP as the alternate port
- Set MSP = 10 in the NAR_CR—sets trace memory bus port as the message select port
- Set TFI = 1—inverts type-filter logic
- Set FE0 = 1 & TF0 = 000101 (0x05) in the NAR_TFR to select data trace of size 8 bits
- Set FE1 = 1 & TF1 = 000110 (0x06) in the NAR_TFR to select data trace of size 16 bits
- Set FE2 = 1 & TF2 = 001101 (0x0d) in the NAR_TFR to select data trace of size 32 bits
- Set FE3 = 1 & TF3 = 001110 (0x0e) in the NAR_TFR to select data trace size of 64 bits
- Set FE0 = 1 & SF0 = 001101 (0x0d) in the NAR_SFR to select source trace ID = 13
- Set ACC = 0 in NAR_SFR so the final filter is ~(data_trace) AND (source = 13)HBDP

Data is written into the output queue at a maximum rate of one entry per cycle. If the current messages source and type pass the filters, then the message is marked for the MSP. When the entry is later pulled from the queue, it is then sent to the MSP. Unmarked entries are sent to the alternate port. Message destination must be recalculated on every end-of-message boundary, using the fixed-width SRC and TYPE fields at the beginning of each Nexus message.

6.5.7 Queue partitioning

It is possible to partition the queue into two sections, one that supports the MSP, and the other that supports the alternate port as shown in Figure 6-3. Messages are routed, based on the MSP bit field, to either the MSP queue partition or the alternate queue partition. This allows both ports to be active simultaneously, since messages can be taken directly from both partitions. Each partition uses an accumulator to store a full central queue width beats of data before writing into memory, since on any given cycle the full Nexus word out of receiver arbitration may be split between the two partitions. Queue partitioning is enabled by setting the AQP bits of the NAR_CR.

The main advantage of using a partitioned queue is that it can improve data throughput and performance when multiple ports are active. It allows the trace memory bus port to utilize block transfer mode, which also improves performance. The only drawback is that queue utilization efficiency can be negatively impacted.

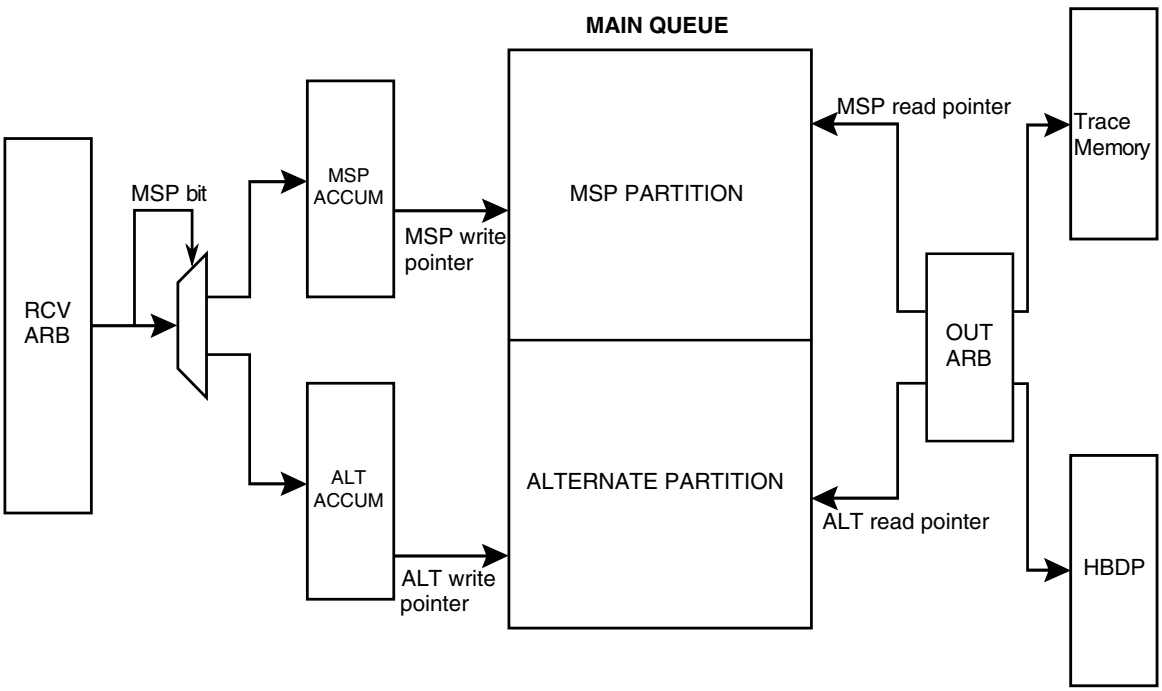


Figure 6-3. Queue partitioning

6.5.8 Message translation and formatting

The NAR performs the formatting on Nexus messages before sending them to an output port. The exact translation performed depends on the destination port.

6.5.8.1 Aurora format

Nexus messages are pre-translated before sending to the HBDP using the Aurora protocol. The NAR output is a stream of 16 × 4-bit words (where 4 is the width of the NAR Aurora out port) that are further encoded by an external Aurora block before being sent for transmission on the Aurora (HBDP) port.

Entries from the output queue are chopped into chunks of 4 × 16-bit words. The amount of data that can be transmitted per Aurora cycle depends on the number of lanes available in the HBDP (transmitted HBDP word = 16 × number of lanes).

6.5.8.2 Trace memory bus format

The trace memory bus works on AHB-EW protocol. Data destined for the trace memory bus port is split back out into 32-bit beats (Nexus word). Depending on the relative sizes of the beat and the width of the trace memory bus port, one or more Nexus words are packed into a trace memory bus data word, with any remaining bits padded to 0.

For example, if the size of the beat = 32 and the width of the trace memory bus = 64, then each trace memory bus data packet contains two beats of Nexus data. Since the data is stored in a regular format (MSEO bits in same place at every location), this makes data interpretation straightforward, regardless of where the trace memory bus port is sending the data. The trace memory bus translation block performs the necessary accumulation and resizing in the likely case where the trace memory bus width does not equal the queue entry size.

6.5.8.3 Nexus format

If the Nexus port is used, entries are sent directly out of the queue with no translation or resizing. The Nexus output, Nexus input, and queue are all the same width.

The Nexus port also includes an EVTO output. This allows EVTO signaling functionality to be maintained in systems with multiple NARs. EVTO can also be used as a trigger by an external event processing block, to allow events such as client halt to occur based on NAR events such as tracebuffer overflow. Currently, the only event that can cause an EVTO assertion is an HBDP hard error/reset, since data is almost certainly lost when the HBDP link goes down, the NAR needs to communicate that back to the clients so that they can generate sync messages.

Many Nexus message types use address/data compression, wherein the compressed data is generated by XORing the current data with the previous data sent. This type of compression can only work if there is no data lost; if data is lost, the client must start over by sending a new sync message which contains the full uncompressed address or data.

6.5.9 Aurora interface

The external Nexus Aurora Link (NAL) contains all of the link-layer, PCS, and control logic for the HBDP channel. The Aurora data is transmitted from the NAR to the NAL for sending to the HBDP. Typically, the NAL powers up coming out of reset, but it can

also be enabled by the NAR by turning on the NAR with the HBDP enabled. The reverse is true as well. If the NAL is active while the NAR is disabled, then the NAR powers up to communication-disabled mode.

Data flow between the NAR and the NAL uses a simple, synchronous interface. If the NAL is active, it indicates this by a ready signal from the NAL. If Aurora is configured as output it is indicated by a signal from the NAR.

The Aurora interface can either be operated in framing or streaming mode. Streaming mode provides the highest performance and lowest communication overhead through the link. Whereas framing mode provides greater robustness and improved error-handling at the receive side of the link. When framing mode is enabled, the NAR encapsulates the Aurora data in frames consisting of 256 Aurora chunks; that is, after every 256 Aurora transmissions, the NAR asserts end of frame and start of frame to mark the end of the last frame and the beginning of the next frame. In either mode, the NAR always asserts a start-of-frame when beginning transmission, and an end-of-frame when its output queue empties, to notify the channel partner of the end of valid data.

6.5.10 Trace memory bus output port usage

The NAR can communicate directly to external memory by sending Nexus data out onto the trace memory bus port, using the AHB-EW (Early Write) protocol. This provides a convenient, secondary area for data in applications or in situations where the HBDP is not available. The trace memory bus access are generated internally by trace memory bus controller.

Because in many systems it is desirable to have the trace memory bus interface run at a divided clock rate relative to the main platform clock, the trace memory bus port operates off of a separate AHB clock. This clock is assumed to be free-running (i.e. unaffected by NAR-enable and the status of the Aurora Link), synchronous to the platform clock used by the rest of the NAR, and either running at the same rate as the platform clock or an integer divide of that rate.

The trace memory bus interface protocol is a slightly modified version of AMBA AHB specification. The write data comes in the address phase instead of data phase and it is termed as Early Write (EW). When NAR uses the trace memory bus port as a destination for Nexus data, it performs a series of writes to a predefined block of memory space.

This memory space is specified through the trace memory bus base address and trace memory bus max transfer configuration registers. The size of the block is set by the max transaction count (MXFR field in NAR_TCR) times the size of a transaction (trace memory bus-width, 64-byte, 128-byte, 256-byte, or 384-byte, depending on block transfer mode and block transfer size). For example, if:

- Base address = 0x0D00_0000 (i.e. TBAHI = 0x0, TBALO = 0x0D00_0000)
- NAR_TCR[BTM] = 1 (Block-transfer mode enabled)
- NAR_TCR[BTS] = 00 (Block-transfer size = 64-byte)
- NAR_TCR[MXFR] = 0x100 (max transaction count)

then the total block size would be 0x100 × 0x40 = 0x4000 bytes. The dedicated debug address range would then be 0x0D00_0000 – 0x0D00_3FFF.

The NAR can either write to the debug space one (trace memory bus-width) word at a time, or for improved performance, it can use a block transfer. If the BTM bit of the trace memory bus Control Register (NAR_TCR) is set, then the NAR uses block transactions to send data through the trace memory bus port, where the size of the block is determined by the setting of the BTS field of the NAR_TCR.

In block transfer mode, the NAR waits until the main queue partition dedicated to the trace memory bus port reaches a specified threshold point, determined by the BTH field of the NAR_TCR, before initiating a block transfer. The default threshold is the size of the block transfer. That is, a block transfer does not start until there is enough data in the associated partition to fill the block. But it might be desirable, for bandwidth reasons or because the associated partition is smaller than the block transfer size, to begin the transfer sooner. Hence, based on BTH, a transfer can begin when the partition is at 25%, 50%, or 75% of the block transfer size. This capability should be used with caution. If the Nexus client does not generate enough new data during the transfer to fill the block, then a block transfer error results.

The NAR generates Nexus trace write transactions for the trace memory bus port using the address from the base address register for the first transaction and then incrementing through the memory space. Once the programmed max number of transactions have been sent, the NAR sets the trace memory bus overflow status bit and, if desired, asserts the NAR event-out. At this point, the NAR can either shut down the trace memory bus port for trace, or, if the TBW bit of the NAR_TCR is set, reset the address to the beginning of the debug space and continue sending trace data. This allows the debugger to view the most recent data, in cases where the buffer overflows.

When TBW bit of NAR_TCR is not set then NAR shuts down the trace memory bus port. To restart the trace memory write again from trace memory base address, the user should write either the NPC trace memory bus base address (TBALO/TBAHI) or the transaction count (MXFR). The user does not have to actually change the value, but can write the same target base address again. The NAR sees any write to one of these registers as a reconfiguration of the trace memory bus port, and resets its internal address pointer to the base address along with clearing the target-space-full flag.

6.5.11 Internal message generation

Generally, the NAR simply conveys Nexus messages from client to port, and does not generate messages of its own. There are, however, certain situations when the NAR needs to communicate directly with the debugger. Internally generated messages are added to the queue the same as external messages. If the queue is partitioned, then internal messages are always channeled to whichever partition is dedicated to the HBDP.

The NAR generates the following message types:

- Device_ID—The first message transmitted when the NAR comes up. It consists of a 6-bit TCODE (TCODE = 1) and the 32-bit device ID. The first beat of the message is always an IDLE (MSEO = 2b11, MDO = all 0s) to allow the debugger Nexus state machine to correctly initialize (by moving to end message or idle state). Device_ID is also generated if the queue is repartitioned (by changing either the ALT or MSP setting in the NAR_CR) while it is not empty. This serves the dual purpose of insuring that the user always gets at least one device_ID message and providing a message boundary in the case where there might have been corruption due to the lost data.
- Watchpoint—The NAR generates its own watchpoint match message under the circumstances shown in Table 6-19.
- NAR_Error (TCODE = 8)—If an error condition is detected inside the NAR, it generates its own error message. An NAR_Error message consists of the 6-bit TCODE, the 6-bit NAR source ID, and a 32-bit data field which corresponds to the NAR status register. An NAR error can be triggered by:
  - HBDP error
  - Illegal configuration
  - Stall Error
  - Trace memory bus read/write access error

This table describes the NAR message formats.

Table 6-16. NAR message formats

Message name	Min packet size (bits)	Max packet size (bits)	Packet type	Packet name	Packet description
Device ID	6	6	fixed	TCODE	TCODE value = 1
	32	32	fixed	DID	DID register value
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp
Error Message[1]	6	6	fixed	TCODE	TCODE value = 8
	6	6	fixed	SRC [2]	NAR source ID
	4	4	fixed	ETYPE	See Table 6-17
	12	12	fixed	ECODE	See Table 6-18
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp
Watchpoint Hit Message[3]	6	6	fixed	TCODE	TCODE value = 15
	6	6	fixed	SRC [2]	NAR source ID
	6	6	fixed	WPHIT	See Table 6-19
	0	24	variable	TSTAMP	Optional, globally synchronized timestamp

1. The Error message conforms to the IEEE-ISTO 5001-2012 standard.
2. Tools should treat the NAR SRC field as a 4-bit SRC field and a 2-bit unused field. The extra 2-bit field (the two most significant bits of the 6-bit TCODE) is always 0b00. This operation is not compliant to the IEEE-ISTO 5001 Nexus standard.
3. The Watchpoint Hit message does not conform to the IEEE-ISTO 5001-2012 standard, it conforms to the IEEE-ISTO 5001-2003 standard. The WPHIT should be a variable length packet, but is implemented as a fixed length packet. In most cases, tools will not see a difference, however, a difference will be seen if a timestamp field is appended to the message.

6.5.11.1 Error conditions

NAR errors can occur upon events such as illegal memory accesses, block-transfer overloads, communication errors, or when the NAR is configured incorrectly. If the configuration registers are programmed incorrectly, it is possible to put the NAR into an illegal state. The NARs response varies depending on the error. For a minor error, the NAR reverts to a default mode and continues operation, while major errors result in complete shutdown.

In either case, a Nexus error message is sent to the Aurora out port, if its enabled, otherwise to the trace memory bus port. The Nexus error message contains both ETYPE and ECODE fields:

- ETYPE is used to denote the basic type of error that has occurred
- ECODE carries specific additional information about the error

All NAR error types are defined in the following table.

Table 6-17. NAR error types

ETYPE	Description	Classification	NAR action	Comment
0000 (h'0)	Receive queue overrun	Minor	Dropped message	If a client sends a data word that the NAR cannot accept (due to a full receive queue), the new data word is dropped.
0001 (h'1)	Internal message contention	Minor	Dropped message	Internal errors/watchpoints occurred faster than NAR could generate messages for them, causing lower-priority message(s) to be dropped. The type of message dropped is conveyed in the ECODE field, see Table 6-18.
0011 (h'3)	Trace memory bus port read/ write access error	Minor	Access fails	Unable to send an Aurora-to-trace memory bus request. Typical causes would be writes larger than the data buffer or sending in a new request before the previous one completes.[1]
0101 (h'5)	Invalid NRR	Minor	Access fails	Tried to access an un-implemented NRR.
1001 (h'9)	Config error	Major	See Table 6-18	The specific configuration error is conveyed in bits 0–3 of the ECODE field (Table 6-18), as well as the NCE field of Status Register.
1100 (h'C)	HBDP error	Minor	N/A	A hard error is automatically reset the Aurora Link and causes it to re-initialize. Although the Link is down, the error message is queued and released as soon as the Link comes back up.
1101 (h'D)	Receive Queue Overrun	Minor	Drop message	If a client sends a data word that the NPC cannot accept (due to a full receive queue), the new data word will be dropped2.
1110 (h'E)	Block transfer error	Minor	Dummy fill	If the Trace memory bus port becomes data-starved during a block transfer, it fills the rest of the block with IDLE messages.
1111 (h'F)	Stall error	Minor	—	—

1. In the case where a request was successfully translated to a Trace memory bus transaction which itself returned an error, that is conveyed in the STATUS field of the Nexus Target Response message subsequently returned to the debugger.

Typically, the ECODE field is used to communicate which type of message was dropped in queue overflow or message contention situation, but for the NAR it is also used to denote the type of configuration error that occurred. If a configuration error is detected, the NAR generates a Config Error Nexus message and transmits it, with the actual type of error encoded in the four MSBs of ECODE, as well as the NCE field of the Status Register. The ECODE field encodings are listed in the following table.

Table 6-18. Configuration error codes

ECODE	ETYPE	Description	NAR action	Comment
0000xxxxxxx1	0001	Watchpoint message dropped	N/A	Occurs with internal message overflow error.
0000xxxxxx1x	0001	Data trace message dropped	N/A	Occurs with internal message overflow error; MMA response message was dropped.

Table continues on the next page...

Table 6-18. Configuration error codes (continued)

ECODE	ETYPE	Description	NAR action	Comment
00001xxxxxxx	0001	Response message dropped	N/A	Occurs with internal message overflow error; target response (memory access or NRR) message was dropped.
0000xxx1xxxx	0001	Status message dropped	N/A	Occurs with internal message overflow error.
000100000000	1001	Config error: multiply defined port	Communication disable mode	Both MSP and ALT were assigned to the same port. NPC waits until the configuration is fixed before proceeding.
001000000000	1001	Config error: no ALT defined	Tracebuffer	An MSP was defined without defining the ALT. The NPC reverts to tracebuffer mode.
001100000000	1001	Config error: cannot enable port	Communication disable mode	User has attempted to enable a port that does not exist in this implementation of the NPC.
010000000000	1001	Config error: cannot support message filtering	Message filtering and/or queue partitioning disabled	User has attempted to enable message filtering or queue partitioning with only one output enabled. Filtering/partitioning is turned off.
100100000000	1001	Config error: no wakeup trigger	Power up in active mode	Occurs if none of the defined clients in the system can cause a wakeup (PUS is set in NAR_CR, but no NAR_STCRs have WUT set).

6.5.11.2 Watchpoint messages

The NAR generates its own watchpoint match message under the circumstances shown in the following table. The value of the Watchpoint Hit field of the Watchpoint Message is also shown in this table.

Table 6-19. Watchpoint hit values

WPHIT	Description	Comment
b010000	Entering Suppress Mode	If (local or global) suppress mode is enabled and the appropriate trigger is received, the NAR generates a watchpoint message. The type of event (in this case, a suppress event) is encoded in the WPHIT field of the watchpoint message (see the Nexus standard, sec. 5.3.21).
b100000	Client Sync	If sync messaging is enabled, the NAR generates a sync watchpoint message whenever a client sync event occurs.
b000100	Trace Memory bus Block Overflow (TBO)	If the trace memory bus block is filled, causing it to either disable trace memory bus data trace or wrap back to the beginning of the dedicated debug memory block and if trace memory bus overflow messaging is enabled, then an TBO message is generated.
b001000	Tracebuffer Wrap	If the NAR is being used in tracebuffer mode with the wrap-around feature, then a watchpoint message is generated every time the tracebuffer wraps back to address 0. This would be useful in informing the user that the buffer had wrapped and that older data had been overwritten.

6.5.12 Timestamp function

The NAR and its clients may transmit timestamps to allow precise ordering of events. Synchronous or asynchronous clients use the NAR timer value broadcasted over a bus to ensure that all timestamps are in sync as shown in the following figure. The NAR uses enable signal to centrally control the enabling of time stamping, which informs the clients to start adding the timestamp field to their message traffic.

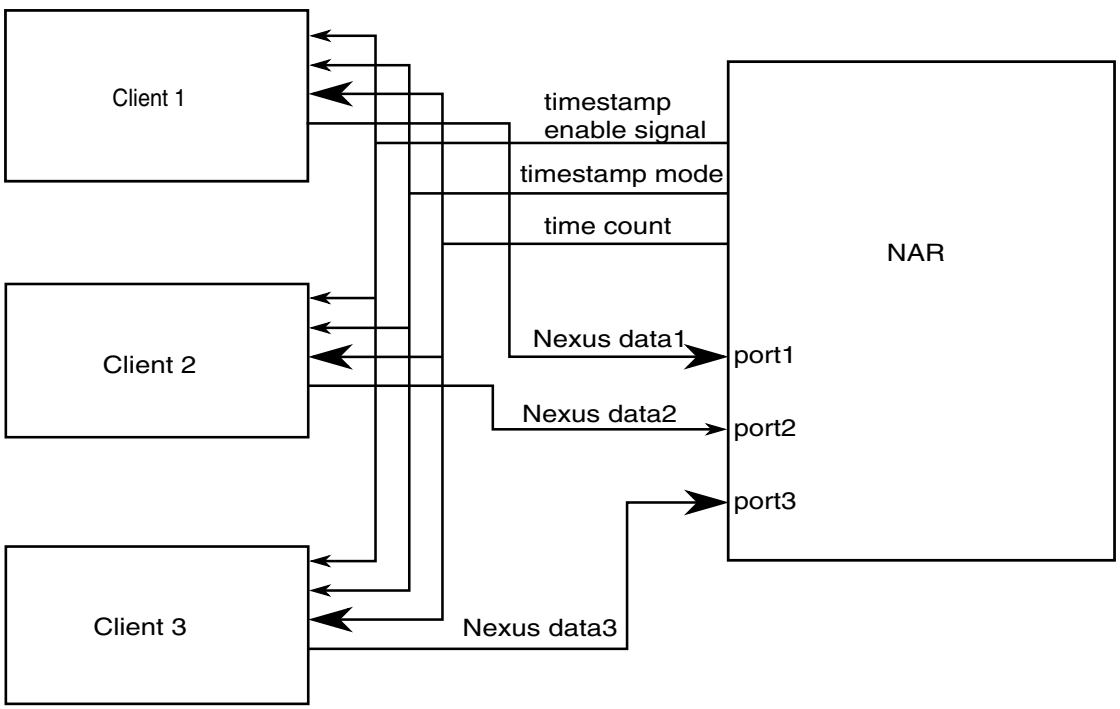


Figure 6-4. Timestamp architecture

Timestamping is enabled by setting the TEN bit of the NAR_CR. This causes the NAR to assert the time stamp enable signal to clients. Typically, the user would also set the TRS bit at this time to reset the NAR Time stamp counter. Internally the NAR has a 30-bit counter external to the NPC block. This 30-bit counter is broadcast to all clients after encoding its value in Gray code. This minimizes the error in synchronization by the clients. The NPC uses an additional internal 24-bit counter to send the timestamp inside its messages. The client converts the gray coded timestamp to binary to append to the Nexus messages.

The timestamp mode, a 2-bit field, is broadcasted to all client which tells the granularity of sending time stamps into message. This is controlled by the TSG bits of the NAR_CR.

Asynchronous clients, those on a separate clock domain from the NAR, synchronize the enable, mode, and timestamp value in their domain. The error introduced by this is assumed to be within tolerable range by application.

A timestamp counter reset can be forced by causing a 0 to 1 transition in the TRS bits of the NAR_CR. The NAR deasserts the enable signal to clients while the reset occurs, and then the counter is restarted (assuming the TEN bit is set). Data throughput can be increased (at the loss of timestamp granularity) by controlling the TSG bits of the NAR_CR. Rather than adding a timestamp to every message, clients can be told to only stamp every 4th, 16th, or 32nd message, easing the bandwidth requirements on the output ports.

The timestamp block maintains the timestamp counter for NAR. It is the central time source for all synchronous/asynchronous clients.

6.5.13 Global (Development tool client) stall control

The external development tool has finite memory. To prevent data loss when this memory is full, the tool gives an event to the device. The DCI block processes this and forwards to NAR block. The dci_evti[1] carries this information. NAR detects the falling edge of dci_evti[1] and loads a counter with a configurable value programmed in GLOBAL_STALL_COUNT field of NAR_AHFPAR. This counter starts decrementing immediately, reaches to zero and stops. During the running period of the counter the Global Stall signal remains asserted thus indicating to stall the message interface to the NAL by that many cycles. The GLOBAL_STALL_COUNT field is five bits so client can be stalled up to 31 cycles.

6.5.14 Overlay and internal trace memory full status generation

In certain events, such as main memory (internal trace memory) full or external overlay trace memory full, it is required to control the clients to stop sending data, to avoid any loss of messages from the clients.

The NAR provides three outputs (triggers) to indicate three different situations as described below:

- Internal trace memory (the main queue) is full
- External debug memory on trace memory bus is full
- External debug memory on trace memory bus is full up to a programmable watermark set by the configuration register

The last two triggers are used by SPU to generate stall and suppress triggers. Typically the stall request should be generated whenever space in the external memory connected to trace memory bus has been exhausted. The NAR starts writing from the first location (wrap) as specified in trace memory bus address register in this condition if overwrite trace memory bus debug memory (TBW bit) is set in the NAR_TCR.

The user should configure the partial full watermark register such that it can give around 90% full indication of external debug memory on the trace memory bus. The emulator tool memory full indication is typically generated from the DCI block which actually communicates with the tool.

For example, the following steps can be used to generate event at around 90% fill level for trace memory connected in AHB-EW and to set the watermark in the configuration register.

- From the configuration, the programmer knows the following:
  - AHB start (base) address—Base address = 0x0D00_0000 (i.e.TBAHI = 0x0, TBALO = 0x0D00_0000
  - Block transfer size—NAR_TCR[BTM] = 1 (Block-transfer mode enabled)
  - Max transfer count—NAR_TCR[MXFR] = 0xC0 (maximum transaction count)
- Thus, the total block size is (0xC0 × 0x40) = 0x3000 bytes. The dedicated debug address range is then 0x0D00_0000 – 0x0D00_2FFF.
- To generate an event at nearly 90% fill level of the trace memory at 1 KB boundary, the value to be programmed in the register NAR_AHFPAR[3:0] is calculated as:
  - Number of 1K blocks in 0x3000 bytes = (3000 right shift by 10 bits) = C = 12
  - So the value to be programmed should be 11 as 90% of 12 is almost 11.
- NAR_AHFPAR[3:0] = (NAR_TBALO[13:10] + 0xB) = (0x7+0xB) 0x2 (upper bits truncated due to width) and enable this with NAR_AHFPAR[10] = 1.

This assumes that NAR starts writing from base address and stops at maximum address and no reading is done in overlay memory trace data area during this operation. Whenever the address bus bits [13:10] match with this value, the partial event is generated (toggled).

Similarly for 65 KB boundary matching NAR_AHFPAR[7:4] needs to be programmed and the corresponding enable bit needs to be set.

6.5.15 Configuration/debug interface JTAG

The configuration and status registers of the NAR are accessed through the standard IEEE 1149.1 JTAG interface. The JTAG frequency (TCK) must be 1/2 of the NAR operating clock. The access mechanism through JTAG is described in this section.

The NAR block uses the IEEE 1149.1-2001 TAP for accessing registers. TAP signals include TCK, TDI, TMS, and TDO. The NAR implements a TAP controller state machine that transitions based on the state of the IEEE 1149.1-2001 16-state state machine. It also implements Nexus controller state machine as defined by the IEEE-ISTO 5001-2001 standard.

The instructions implemented by the NAR TAP controller are listed in Table 6-20. Each unimplemented instruction acts like the BYPASS instruction. The size of the NAR instruction register is 4 bits.

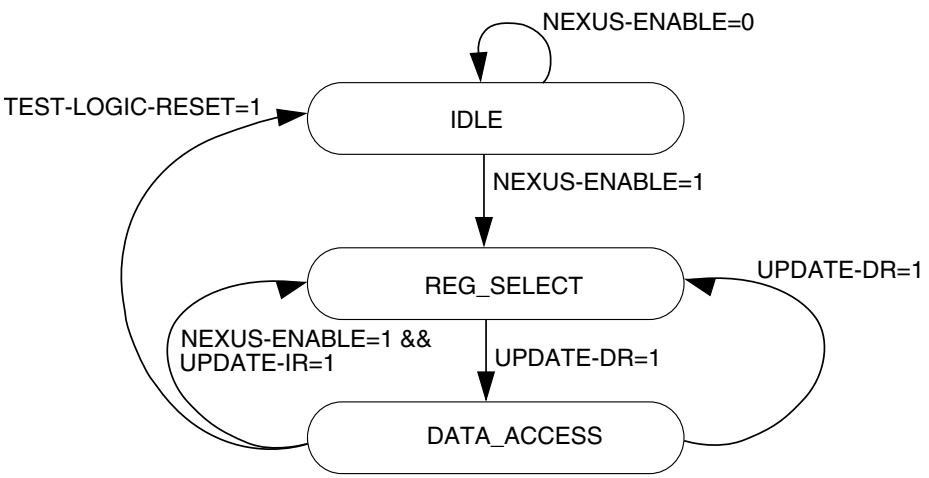


Figure 6-5. Nexus controller state machine

Table 6-20. Implemented instructions

Instruction Name	Private/ Public	Opcode	Description
NEXUS_ENABLE	Public	0x0	Activate Nexus controller state machine to read and write NAR registers.
BYPASS	Private	0x0F	NAR BYPASS instruction. Also the value loaded into the NAR IR upon exit of reset.

Data is shifted between TDI and TDO starting with the least significant bit as illustrated in the following figure. This applies for the instruction register and all Nexus tool-mapped registers.

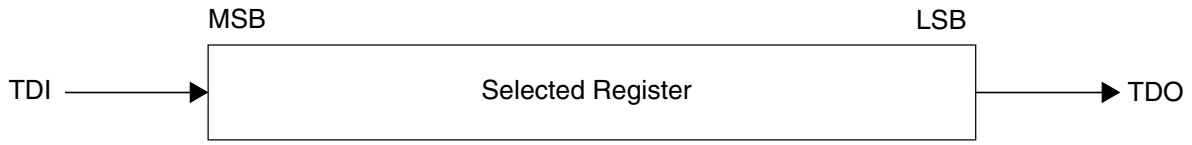


Figure 6-6. Shifting data into register

6.5.15.1 Enabling the NAR TAP controller

Assertion of the power-on reset signal resets the NAR TAP controller. When not in power-on reset the NAR TAP controller is enabled by driving JCOMP with the NAR enable value and exiting the Test-Logic-Reset state. Loading the NEXUS-ENABLE instruction then grants access to Nexus debug.

6.5.15.2 Loading NEXUS-ENABLE instruction

Access to the NAR registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. This instruction is shifted in via the SELECT-IR-SCAN path and loaded in the UPDATE-IR state. At this point, the Nexus controller state machine, shown in Figure 6-5, transitions to the REG_SELECT state. The Nexus controller has three states: idle, register select, and data access. The following table illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

Table 6-21. Loading NEXUS-ENABLE instruction

Clock	TMS	IEEE 1149.1 state	Nexus state	Description
0	0	RUN-TEST/IDLE	IDLE	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	IDLE	Transitional state
2	1	SELECT-IR-SCAN	IDLE	Transitional state
3	0	CAPTURE-IR	IDLE	Internal shifter loaded with current instruction
4	0	SHIFT-IR	IDLE	TDO becomes active, and the IEEE 1149.1-2001 shifter is ready. Shift in all but the last bit of the NEXUS-ENABLE instruction.
5 TCKS				
12	1	EXIT1-IR	IDLE	Last bit of instruction shifted in
13	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
14	0	RUN-TEST/IDLE	REG_SELECT	Ready to be read/write Nexus registers

6.5.15.3 Selecting a JTAG register to create a bus (memory or peripheral) access

When the NEXUS-ENABLE instruction is decoded by the TAP controller, the input port allows the development tool access to all NAR registers. Each register has a 7-bit address index.

All register access is performed via the SELECT-DR-SCAN path. The Nexus controller defaults to the REG_SELECT state when enabled. Accessing a register requires two passes through the SELECT-DR-SCAN path: one pass to select the register and the second pass to read/write the register.

The first pass through the SELECT-DR-SCAN path is used to enter an 8-bit Nexus command consisting of a read/write control bit in the LSB followed by a 7-bit register address index, as illustrated in the following figure. The read/write control bit is set to 1 for writes and 0 for reads. The index for registers are shown in Table 6-1.

Table 6-22. Nexus command format for register selection

MSB	LSB
7-bit register index	R/W

The second pass through the SELECT-DR-SCAN path is used to read or write the register data by shifting in the data (LSB first) during the SHIFT-DR state.When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the CAPTURE-DR state.When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the UPDATE-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

The following table illustrates a sequence which writes a 32-bit value to a register.

Table 6-23. Writing a 32-bit value to a register

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	REG_SELECT	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	REG_SELECT	First pass through SELECT-DR-SCAN path
2	0	CAPTURE-DR	REG_SELECT	Internal shifter loaded with current value of controller command input.
3	0	SHIFT-DR	REG_SELECT	TDO becomes active, and write bit and 4 bits of register index shifted in.
7 TCKs				
11	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI
12	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register
13	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through SELECT-DR-SCAN path
14	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register

Table continues on the next page...

Table 6-23. Writing a 32-bit value to a register (continued)

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
15	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				
47	1	EXIT1-DR	DATA_ACCESS	Last bit of current value shifted out TDO. Last bit of new value shifted in TDI.
48	1	UPDATE-DR	DATA_ACCESS	Value written to register
49	0	RUN-TEST/IDLE	REG_SELECT	Controller returned to idle state. It could also return to SELECT-DR-SCAN to write another register.

6.5.16 Trace memory bus controller

The purpose of the trace memory bus is to transfer trace data from main queue to external overlay memory. This port works on a modified AHB protocol that supports early write. It has AHB interface signals as defined in AMBA specification (Rev 2.0). In addition, there are side-band signals for added functionality.

Key features of the module include:

- AHB-EW Interface
  - Early Write (for higher BW in special module)
  - 32-bit address bus width
  - 64-bit data bus width
  - Transfer Types
  - IDLE, NONSEQ, SEQ
  - Single Transfers
  - Burst Transfers
  - Configurable support for INCR
  - Configurable support for INCR4, INCR8, INCR16, WRAP4
  - Response Type
  - OKAY, ERROR
  - Transfer Size
  - Byte, Half Word, Word, Double-Word

- Protection Control
- Cacheable/ Non Cacheable Access Control
- User/Privilege Access

- Endianess Support
  - Configurable Little Endian/Big Endian support on AMBA-AHB interface
- Pipeline Support
  - Total of two transactions (One plus one ongoing)

6.5.16.1 Operation

The trace memory bus controller allows all read/write operations to be cut through. The controller has a pipeline depth of two. While the data phase of one transaction is ongoing, it can accept one more read/write request from the internal bus master and initiate the address phase on the AHB interface during the last data cycle of the ongoing transaction.

6.5.16.2 Write operation

When the trace memory bus initiator asserts a request for a write operation, it can find the acknowledge signal parked asserted if the controller is ready to accept the transaction. If the pipeline is full or in case of an ongoing read transaction, the controller inserts wait states by keeping the acknowledge low. The trace memory bus controller initiates the address phase on AHB interface once it accepts the transaction by asserting the acknowledge on the trace memory bus interface. The write data from the trace memory bus initiator is accepted by the gasket by asserting the acknowledge only on completion of address phase at AHB interface, which is signified by receiving the ready signal asserted. Simultaneously, the data is sent to the data bus. Once the requisite number of bytes have been accepted from the trace memory bus initiator, the gasket generates end of data and transfer according to the Trace memory bus protocol.

The AHB protocol supports data transactions of byte, half-word, word and double-word (when both trace memory bus and AHB data bus width is 64 bits). A write transaction from the trace memory bus initiator take place as single/multiple transactions on the AHB side.

6.5.16.3 Read operation

When the trace memory bus initiator asserts a request signal to request for a read operation, it can find the acknowledge signal parked asserted if the gasket is ready to accept the transaction. If the pipeline is full, the controller inserts wait states by keeping the acknowledge low. The trace memory bus controller initiates the address phase on AHB-EW interface once it accepts the transaction by asserting the acknowledge on trace memory bus interface. The controller reads the data from the bus when the ready signal is sensed asserted. The data is transferred on the trace memory bus data bus with the data acknowledge asserted. The gasket asserts end of data and transfer at the end of transaction as per trace memory bus protocol.

The AHB protocol supports data transactions of byte, word, half-word and double-word (when both trace memory bus and AHB data bus width is 64-bits). A read transaction from trace memory bus initiator takes place as single/multiple transaction on the AHB interface before the read data is merged and sent on the trace memory bus read data bus.

In case of variable data bus width on trace memory bus and AHB interface (32-bit data bus on one interface and 64-bit data bus on the other interface), a maximum of 64 bits is read on each beat.

6.5.17 Aurora client controller

This block converts the Aurora client protocol from the PD NAR into an internal client protocol that is accepted by the NPC block. The Nexus beat size must be same at both interfaces. The relation between the Aurora client data width and the NPC client interface data width must be maintained as described here.

The Aurora data width is defined as the Aurora width × 16 bits, where 16 bits comprise one Aurora word. The Nexus client data width is defined as Bn × 32, where 32 = MDO bits + 2 MSEO bits and Bn is the number of beats.

Thus:

Aurora data width = the Nexus client data

Aurora width × 16 = Bn × 32

Aurora width = 2 × Bn

This block works in a similar way to the legacy client controller block except for the interface level control. The Aurora client has the information about the FIFO size. It sends as many data as the FIFO size then waits for the data accept signal from this block.

The controller gives the data accept only when a read occurs and sends data to the NPC interface. The FIFO act as a storage and serves the purpose of clock domain transfer as well.

6.6 Application information

The base application for the NAR is collecting and collating Nexus trace data. Programming examples for how to set up the NAR for trace are given in Output arbitration.

Example flow of NAR configuration for different cases.

Configuration with Two output port enabled

1. For Two port enabled one port must be Message Select (MSP) and other Alternate (ALT); Trace Memory and Aurora ports with multiple clients

   Selection of MSP port depends on user choice. One option could be based on the requirement of Filtering. If Messages from some known sources with some specific type (if any) is required to be sent to a particular port then select user can select that port as MSP. All other messages will go to the Alternate port. The filter can be used as inverted manner where unmatched messages will go to MSP. (Please see section “Output arbitration” for details)
   - Configure the Source Filter register (NAR_SFR), To enable proper filtering of messages based on Message source going to MSP or ALT port
   - Configure the Type Filter register (NAR_TFR), To enable proper filtering of messages based on Message Type going to MSP or ALT port
   - Configure Trace Memory port attributes : (This step is not required if Trace memory port is not used)
     - Start Address (base address) in NAR_TBALO & NAR_TBAHI register
     - Transfer mode, Block size, Threshold in %, Wrapping enable and Total size of memory
   - Configure Client disable register NAR_CDR (disable clients which are not required)
   - Configure NPC suppress trigger registers NAR_STCR[x] (x=0-3) (suppress the clients for which data is not to be forwarded at output) (Difference between suppressing and disabling is that suppressing can be done dynamically during NAR operation but disabling can’t be done this way which may leads to system hang)
   - Configure the Event generation register (NAR_AHFPAR) for programmable fill level event generation
   - Configure the NAR control register for other controls and Enable NAR

- Set the MSP as Nexus Out and ALT as Trace memory
- Set AQP for partitioning the Main Queue for ALT and MSP ports
- Set time stamp controls
- Enable NAR to start operation
- Configuration for Single Output port with single partition in NAR central Message Queue

2. Only Trace Memory port is enabled with multiple client
   - Configure Trace Memory port attributes : (This step is not required if Trace memory port is not used)
     - Start Address (base address) in NAR_TBALO & NAR_TBAHI register
     - Transfer mode, Block size, Threshold in %, Wrapping enable and Total size of memory
   - Configure Client disable register NAR_CDR (disable clients which are not required)
   - Configure the Event generation register (NAR_AHFPAR) for programmable fill level event generation
   - Configure the NAR control register for other controls and Enable NAR
     - Set the Trace Memory port as ALT port (Alternate port is only the main port in single out port application)
     - Set AQP for disabling partitioning
     - Set time stamp controls
     - Set watch point controls
     - Enable NAR to start operation

Chapter 7
Buddy Device—Nexus Aurora Link (NAL)

7.1 Introduction

The Aurora Protocol is used to send debug information over a high-speed serial link out of the device. The Nexus Aurora Link (NAL) consists of all the logic on the MCU needed to implement the connection up to the physical-layer SerDes and transceivers; it includes Aurora control and data multiplexing, data-encoding and striping, and the Physical Coding Sublayer (PCS) portion of the protocol. The following figure details the NAL block diagram. The number of lanes, n, is 4.

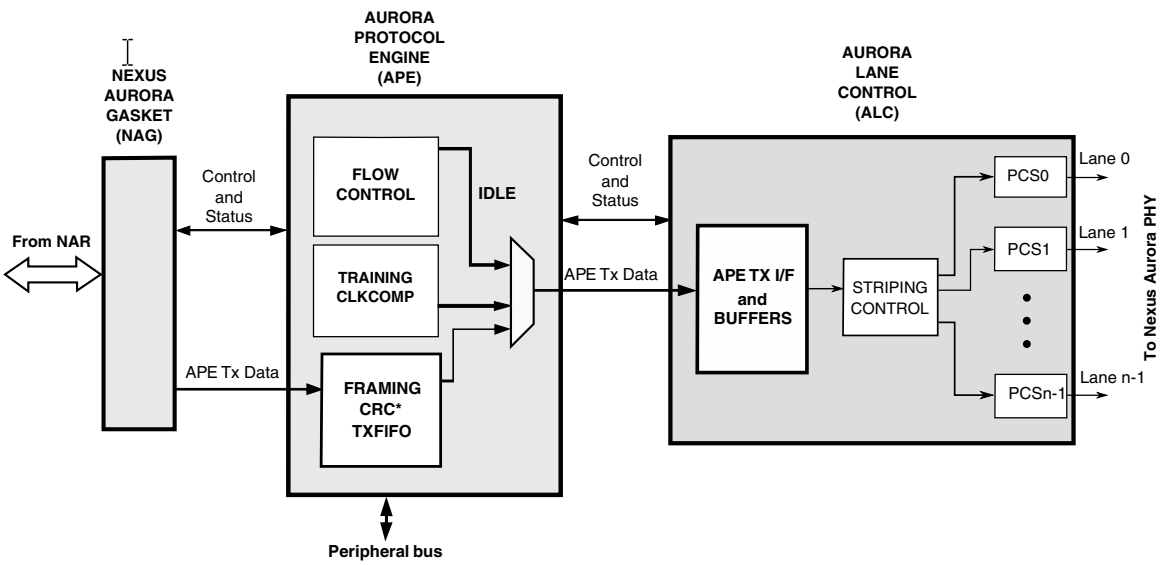


Figure 7-1. NAL block diagram

Debug data in the Nexus format from multiple clients is collected by the Nexus Aurora Router (NAR) ; messages are then packetized and sent across an asynchronous clock boundary-crossing gasket to the Aurora Protocol Engine, which performs all the logical layer functions and flow control; data from the protocol engine is sent on to the Aurora Lane Control (ALC) block, which performs data-striping and sizing, based on the number of lanes selected for debug; it also performs the PCS functions of the link. The PCS

drives the Nexus Aurora Physical (NAP) interface. The NAP, in conjunction with the NAL, supports one-way simplex high-speed serial communication with an external debugger.

The NAL is composed of three main blocks: the NAR/Aurora Gasket, the Aurora Protocol Engine, and Aurora Lane Control. The table below lists NAL functionality by block and lists the high level features for the Nexus Aurora PHY.

Table 7-1. NAL and NAP feature summary

Sub-Component	Feature
NAR Aurora Gasket (NAG)	• Transmit FIFO to buffer data as it crosses the NAR/NAL clock-domain boundary and Transmit FIFO space available indication to NAR. The NAR uses Transmit FIFO space available indication(s) to determine if data can be sent to the NAL. • Platform to Domain clock domain crossing. The NAR is on the core clock domain while the NAP is on a domain sourced from the external PHY clock input signals.
Aurora Protocol Engine (APE)	• Link training • Per the Aurora Protocol specification, clock-compensation, and data framing • Cyclic Redundancy Check (CRC) generator
Aurora Link Control (ALC)	• Striping of data stream from the Aurora core across all available lanes in two-byte chunks • PCS functionality for NAL
Nexus Aurora PHY (NAP)	• Physical transmission of serial data to device pins • External clock source for SerDes clock domain

7.2 Transmit operation

During transmit operation, the NAR collects debug data from Nexus clients over a 32-bit interface: 30 bits are used for MDO and 2 bits are used for MSEO information. The NAR places MDO data into a payload (maximum payload size depends on the number of lanes, each lane has a maximum of 7424 symbols before terminating the payload) and adds the following:

- 16-bit start symbol
- 16-bit end symbol
- 16-bit CRC if enabled before end of previous symbol

Each payload is transferred from the NAR to the NAR/NAL gasket, through the Aurora Protocol Engine, the Aurora Lane Control and finally out to the device SerDes/LVDS pins through the NAP.

After each end symbol packet transfer, the NAL adds a clock compensation sequence needed for downstream SerDes devices to remain synchronized to the MCU.

NOTE

Symbol counting for clock compensation starts just prior to the verification symbols; therefore, training symbols are excluded when counting.

7.3 Nexus Aurora formatting

The Aurora physical interface is the transport mechanism for Nexus messages. The Nexus messages are formatted by the Aurora formatter and striped into Aurora lanes. The Nexus messages themselves encapsulated into a virtual Nexus port that consists of a 30-bit Message Data Output (MDO) port and a 2-bit Message Start/End Output (MSEO) signal. The MSEO portion of the port is the two least significant bits (LSB) of the 32-bit Aurora message. The rest of the 32-bit Aurora frame consists of 30 MDO bits that are added LSB first.

7.4 Static training

The NAL supports static training. The NAL performs Static training by using programmable timers to determine how long to stay in each stage of the training process. In static training, there is no communication between the channel partners to exchange training status; rather the NAL sends the align/bond/verify sequence to its partner for a set amount of time sufficient for the receive partner to complete the training stage.

Note

Prior to initialization, the NAL must be reset to its POR state; to do this, assert the block reset. See the RST bit field description in the NAL General Control Register (NAL_GCR).

The following steps enable static training:

1. Program the counters in the NAL_TCR and set the STE bit
2. Set the RST bit

This resets the channel and start the training routine. The time spent in each stage can range from 16 to 3072 cycles; once a counter times out, a counter time-out signal is asserted to the Aurora Protocol Engine (APE) to prompt it to move to the next stage. Additionally, it is possible to hold the training procedure in any of the stages indefinitely, by setting one of the three hold bits (AHD/BHD/VHD). When set, these bits cause the

channel to sit in Align/Bond/Verify until unset, allowing a user to interactively step through the training procedure by monitoring the status at the debugger side and then stepping the device side to the next stage.

During the Aurora training sequence, the NAL generates a specific set of character sequences per the Aurora protocol specification which are to be transmitted simultaneously on all lanes.

7.5 Clock compensation considerations

The Aurora Protocol Specification describes clock compensation as a means for the link to compensate for clock rate differences between the transmitter and receiver. The spec requires that a clock compensation be inserted at a minimum every 10000 symbol pairs; this should allow the link to accomodate up to a 200 ppm difference in clock rates.

Due to the fact that the NAL is restricted to only inserting clock-compensation sequences after an end-of-frame, it may not be possible to always meet the Aurora spec requirement. However, there are ways to control the framing of data through the link and insure that satisfactory performance is maintained. There are two types of scenarios that can prevent an adequate number of clock compensation sequences from being generated: the sparse data case and the unframed data case. In the sparse data case, there are long periods of time when no data is flowing through the link; no data means no frames. In the unframed case, there is plenty of data, but this data is not framed, and so again, clock-compensation never gets a chance to make it out.

The unframed case is simplest to address. The NAR can either operate in framed or unframed mode. In unframed mode, an end-of-frame is only injected when the NAR runs out of data (as soon as new data is collected, it is sent out accompanied with a start-of-frame; if the NAR is configured in this mode, the clock-compensation spec can easily be violated. However, in framed mode, an end-of-frame/start-of-frame pair is injected after every 256 data transmissions. Here, each frame consists of 256 x AW symbol pairs. Inside the APE, a clock-compensation counter keeps track of how much data has been transmitted to the ALC since the last clock-compensation sequence; once the counter hits 7424, clock-compensation will be sent on the next end-of-frame.

On the other hand, even if the NAR is in framing mode, there is little that can be done if the NAR has no data to send. In the sparse data scenario, the NAL could end up transmitting idles for an undetermined number of cycles while waiting for the NAR to provide new data. However, a user can get around this problem by insuring that the NAR always has at least some minimal amount of data to send. There are several ways to do this:

- If full-duplex infrastructure is available, then the simplest thing to do is for the debugger to monitor the TX data stream from the NAL for long idle periods. If the link has been idle for longer than some minimal amount of time (say 8000 symbol pairs), then the debugger could send in a read request through the aurora-in RX port -- a quick NRR read of the dev_id register would be ideal for this purpose. The NAR would then return the requested data, accompanied by a start-of-frame/end-of-frame pair, which the NAL could then tag with clock-compensation. See the NAR block guide for more details on aurora-in NRR access.
- If full-duplex is not available, then the debugger could still force the NAR to send the dev_id message by going through the JTAG/SAP port and setting the NAR SRESET bit. Reinitializing the NAR puts it back into its startup state, wherein it will invariably send device ID before doing anything else. The only drawbacks to this approach are its long latency (going through JTAG) and the fact that NAR must be correctly reconfigured after every SRESET.
- Another approach is to use periodic watchpoints to insure that the data stream never falls completely silent for more than some predefined count. Most nexus clients, including the NAR itself, can generate a watchpoint sync message based on an external or internal trigger. By hooking one of these triggers up to a counter (a performance-monitor, event-processing unit debug counter can be used for this purpose), we can insure that, even if there is no other trace data traffic, there will at least be periodic watchpoint messages which can be used to trigger clock compensation. The drawback here is that it requires devoting one of the debug counters for clock compensation, but this may not always be a problem: one of the proposed methods of correlating nexus timestamps from different asynchronous domains is to use the same sort of periodic sync trigger to force all asynchornous clients to send a stamped watchpoint message at (roughly) the same time. This asynchronous sync trigger can also be used for clock compensation.

7.6 Programmable registers

The Nexus Aurora Link (NAL) module provides a Nexus parallel to Aurora formatting for the MCU trace information. These registers are not memory-mapped and can only be accessed via the JTAG interface.

Table 7-2. NAL memory map

Offset	Register name	Width (in bits)	Access	Reset value
0h	NAL General Status Register (NAL_GSR)	32	R	0008_0000h
8h	NAL General Control Register (NAL_GCR)	32	R/W	0000_0000h
Ch	NAL Training Control Register (NAL_TCR)	32	R/W	0000_0000h

7.6.1 Nexus Aurora Link General Status Register (NAL_GSR)

The NAL_GSR is a read only register that describes the status of the NAL.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												TXONLY	0		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			TXCFG			0									AS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 7-3. NAL_GSR field descriptions

Field	Description
31–20 Reserved	Reserved This bitfield is reserved.
19 TXONLY	Transmit Only mode. Indicates that Aurora is configured in a transmit-only mode. 0 Reserved 1 Aurora is configured for transmit-only simplex communication. Training method is configured to use Static Training.
18–13 Reserved	Reserved This bitfield is reserved.
12–10 TXCFG	TX Lane Configuration. Number of enabled TX lanes. These bits are read-only and are controlled external to the NAL. 000 Reserved 001 2 TX lanes 010 Reserved 011 4 TX lanes 100 Reserved 101 Reserved 110 Reserved 111 Reserved
9–1 Reserved	Reserved This bitfield is reserved.
0 AS	Aurora Status. 0 Aurora is not enabled. 1 Aurora is enabled.

7.6.2 NAL General Control Register (NAL_GCR)

The NAL_GCR configures the behavior of the APE. It can be programmed to reset the Aurora channel, or to configure simplex/duplex initialization and to enable debug features.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RST	ALCPD	0												FCM	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	PCRST	0	SLD			0	BOOE	BOOD				CRCEB	CCOEN	0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 7-4. NAL_GCR field descriptions

Field	Description
31 RST	Reset the Aurora Channel. 0 to 1 transition resets the channel and begins the training sequence. 0 normal operation 1 transition from 0 to 1 resets channel and starts training sequence
30 ALCPD	ALC/NAL Power Down. Gate off clocks to the NAL blocks (ALC and APE). 0 NAL is enabled with all clocks running provided other mechanisms of powerdown (such as alc_num_lanes) are not in effect. 1 NAL is powered down and inactive.
29–18 Reserved	Reserved This read-only bitfield is reserved and always has the value zero.
17 FCM	Flow Control Mode. 0 Completion Mode 1 Immediate Mode
16-15 Reserved	Reserved and must be written with zero.
14 PCRST	Protocol Converter Reset. Resets the ALC when asserted.
13 Reserved	Reserved This read-only bit is reserved and always has the value zero.

Table continues on the next page...

Table 7-4. NAL_GCR field descriptions (continued)

Field	Description
12–10 SLD	Symbol Lock Delay. 000 Allow 3 characters to accumulate in elastic buffer before starting to read 001 Allow 4 characters to accumulate in elastic buffer before starting to read 010 Allow 5 characters to accumulate in elastic buffer before starting to read 011 Allow 6 characters to accumulate in elastic buffer before starting to read 100 Allow 7 characters to accumulate in elastic buffer before starting to read 101 Allow 8 characters to accumulate in elastic buffer before starting to read 110 Allow 9 characters to accumulate in elastic buffer before starting to read 111 Allow 10 characters to accumulate in elastic buffer before starting to read
9 Reserved	Reserved This read-only bit is reserved and always has the value zero.
8 BOOE	Bond Offset Override Enable. 0 ALC channel bond attempts to determine channel bond offset. 1 ALC uses user override setting for channel bond offset.
7–4 BOOD	Bond offset override data. Only valid when BOOE is set. 0000/1000 use 0-cycle offset 0001 use -7 cycle offset 0010 use -6 cycle offset 0011 use -5 cycle offset 0100 use -4 cycle offset 0101 use -3 cycle offset 0110 use -2 cycle offset 0111 use -1 cycle offset 1001 use +1 cycle offset 1010 use +2 cycle offset 1011 use +3 cycle offset 1100 use +4 cycle offset 1101 use +5 cycle offset 1110 use +6 cycle offset 1111 use +7 cycle offset
3 CRCEB	CRC Enable. 0 CRC generation and checking is disabled for TX and RX data. 1 CRC is inserted for TX frames and CRC is checked for RX paths.
2 CCOEN	Clock Compensation Override Enable. This is a debug feature which allows the duration between clock compensation sequences to be reduced to 1280 cycles. 0 Clock compensation counter times out at 7424 cycles 1 Clock compensation counter times out at 1280 cycles

Table continues on the next page...

Table 7-4. NAL_GCR field descriptions (continued)

Field	Description
1–0 Reserved	Reserved This read-only bitfield is reserved and always has the value zero.

7.6.3 NAL Training Control Register (NAL_TCR)

The NAL_TCR supports static training as described in Static training.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	STE	AHD	BHD	VHD	Reserved					ATC				Reserved					BTC				Reserved						VTC			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 7-5. NAL_TCR field descriptions

Field	Description
31 STE	Static Training Enable. 0 Reserved 1 Timer-based training method is used, ie. RX data is ignored and training is established based on ATC/BTC/VTC timers.
30 AHD	Hold in Align. 0 Follow Aurora training sequence 1 Remain in align until this bit is cleared (applies even during duplex-mode training)
29 BHD	Hold in Bond. 0 Follow Aurora training sequence 1 Remain in bond until this bit is cleared. (applies even during duplex-mode training)
28 VHD	Hold in Verify. 0 Follow Aurora training sequence 1 Remain in verify until this bit is cleared. (applies even during duplex-mode training)
27–23 Reserved	Reserved This bitfield is reserved and must be written with zero.

Table continues on the next page...

Table 7-5. NAL_TCR field descriptions (continued)

Field	Description
22–19 ATC	Align Timer Count 0000 Remain in align for 16 cycles 0001 Remain in align for 24 cycles 0010 Remain in align for 32 cycles 0011 Remain in align for 48 cycles 0100 Remain in align for 64 cycles 0101 Remain in align for 96 cycles 0110 Remain in align for 128 cycles 0111 Remain in align for 192 cycles 1000 Remain in align for 256 cycles 1001 Remain in align for 384 cycles 1010 Remain in align for 512 cycles 1011 Remain in align for 768 cycles 1100 Remain in align for 1024 cycles 1101 Remain in align for 1536 cycles 1110 Remain in align for 2048 cycles 1111 Remain in align for 3072 cycles
18–14 Reserved	Reserved This bitfield is reserved.
13–10 BTC	Bond Timer Count 0000 Remain in bond for 16 cycles 0001 Remain in bond for 24 cycles 0010 Remain in bond for 32 cycles 0011 Remain in bond for 48 cycles 0100 Remain in bond for 64 cycles 0101 Remain in bond for 96 cycles 0110 Remain in bond for 128 cycles 0111 Remain in bond for 192 cycles 1000 Remain in bond for 256 cycles 1001 Remain in bond for 384 cycles 1010 Remain in bond for 512 cycles 1011 Remain in bond for 768 cycles 1100 Remain in bond for 1024 cycles 1101 Remain in bond for 1536 cycles 1110 Remain in bond for 2048 cycles 1111 Remain in bond for 3072 cycles

Table continues on the next page...

Table 7-5. NAL_TCR field descriptions (continued)

Field	Description
9–4 Reserved	Reserved This bitfield is reserved.
3–0 VTC	Verify Timer Count 0000 Remain in verify for 16 cycles 0001 Remain in verify for 24 cycles 0010 Remain in verify for 32 cycles 0011 Remain in verify for 48 cycles 0100 Remain in verify for 64 cycles 0101 Remain in verify for 96 cycles 0110 Remain in verify for 128 cycles 0111 Remain in verify for 192 cycles 1000 Remain in verify for 256 cycles 1001 Remain in verify for 384 cycles 1010 Remain in verify for 512 cycles 1011 Remain in verify for 768 cycles 1100 Remain in verify for 1024 cycles 1101 Remain in verify for 1536 cycles 1110 Remain in verify for 2048 cycles 1111 Remain in verify for 3072 cycles

Chapter 8
Buddy Device—Nexus Aurora PHY (NAP)

8.1 Introduction

The Nexus Aurora PHY (NAP), in conjunction with the Nexus Aurora Link (NAL), supports one-way simplex high-speed serial communication with an external debugger. Configuration settings in the Nexus Aurora Router (NAR) Nexus Aurora Phy control register (NAR_NAPCR) specify the NAP and NAL lane configuration. The NAL performs all logical layer functions, flow control, data-striping and sizing based on the number of data lanes available for use, and physical coding sublayer functions on the data being sent to the NAP for transmission. The NAP receives 8b/10b encoded data from the NAL then serializes and transmits this data through a given PHY lane's Low Voltage Differential Signaling (LVDS) buffers to an external debugger.

Each NAP lane consists of a data symbol character accumulator which registers the 10-bit character for that lane from the NAL on the rising edge of the NAL clock. Next, a serializer takes the 10-bit character and serializes it into the individual data bits that are then transmitted at the PHY clock rate via the LVDS output drivers to the external debugger receive channel.

Note

The NAL performs static link training for an attached debugger's PHY. The NAP is not directly aware of the link training operation, since it merely transmits the data presented to it by the NAL.

On the receive side of the external debugger, it is the responsibility of the external debugger to perform elastic buffering, symbol detection, symbol locking (called Lane Alignment in Aurora terminology), 10b/8b decoding, decode and disparity error tracking in the received data stream.

The following figure shows a high level block diagram of the High Speed Nexus Interface and the NAP.

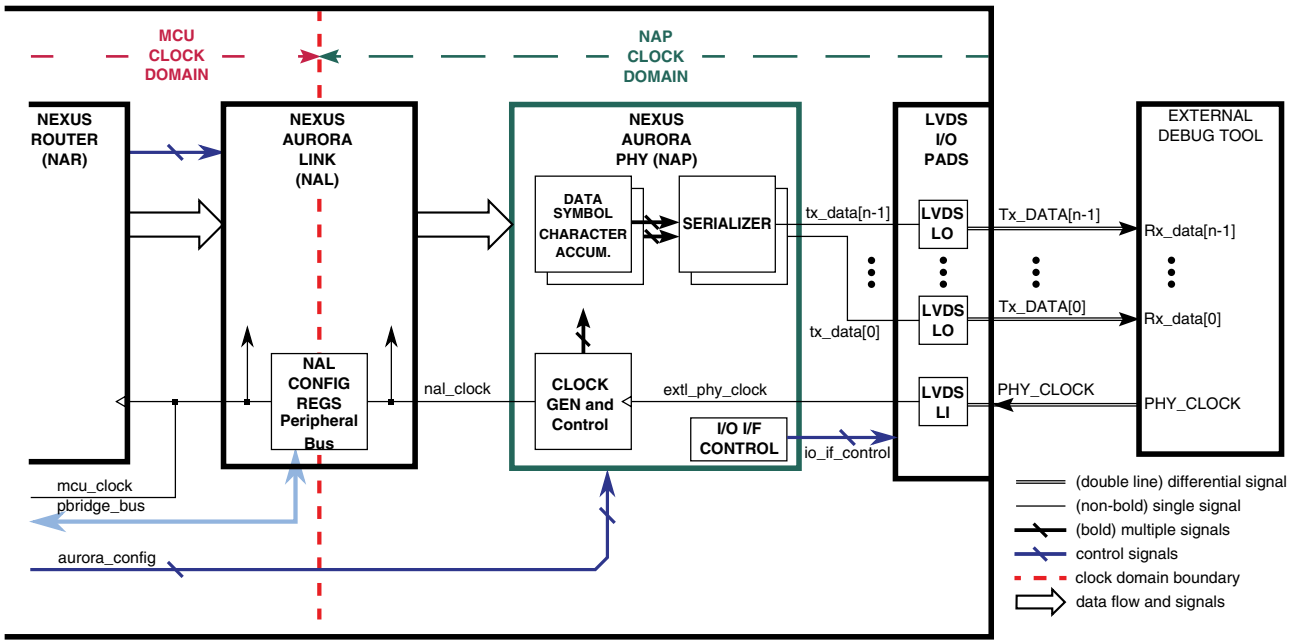


Figure 8-1. High speed Nexus interface block diagram

8.1.1 Features

The features of the NAP are as follows:

- Xilinx Aurora Protocol Specification V2.x compliant
- 1Gb/s per channel trace data payload bandwidth with 1.25 GHz clock
- Simplex mode with static (timer-based) channel training

8.1.2 Modes of operation

The operating mode of the NAP is determined by the device control signals. The functional configuration is determined by the NAR_NAPCR setting; functional operation is enabled by the NAR.

8.1.2.1 Reset

During Nexus Reset all internal NAP resources are reset, all output ports are disabled, and all inputs are ignored. Since there is no device clock domain logic in the NAP itself, the NAP is unaffected by device system reset. Depending on the NAP configuration settings, the NAP may activate upon exiting Nexus Reset if any Aurora lanes are enabled during this reset mode, else it remains disabled and inactive.

8.1.2.2 Link training

Before beginning trace output mode operation, the Aurora Link must go through a training process conducted by the NAL to ensure that an external debugger can correctly reconstruct the transmitted data stream. Training proceeds once the link powers up, and must complete successfully before data transmission can occur.

The NAL initializes and trains the Aurora Link in a simplex configuration using a timer based static-training method. The initialization and training sequence described in Aurora Protocol Specification progresses from one step to the next via the use of programmable timers contained in the NAL. A timer is programmed for each stage of training and as each timer expires, the training sequence proceeds to the next stage until complete. The NAP is not directly aware of the link training operation, since it merely transmits the data presented to it by the NAL.

8.1.2.3 Transmit

During Nexus trace output operation, it is the responsibility of the NAR to push data into the Aurora Link at a sufficiently high rate to ensure that the link does not become data-starved. The NAL and NAP support up to 4 transmit lanes. The Aurora encapsulated data is modified by the NAL to ensure the appropriate number of symbols are transmitted through the NAP.

8.2 External signal description

The external signals of the NAP are listed in the following table. Each of the NAP signals consists of a positive and negative differential pair.

Table 8-1. NAP signals

Signal Name	Direction	Description
PHY_CLOCK (CLKP/CLKN)	Input	NAP clock input from external debug tool

Table continues on the next page...

Table 8-1. NAP signals (continued)

Signal Name	Direction	Description
TX Data0 (TX0P/TX0N)	Output	NAP Tx data for lane 0
TX Data1 (TX1P/TX1N)	Output	NAP Tx data for lane 1
TX Data2 (TX2P/TX2N)	Output	NAP Tx data for lane 2
TX Data3 (TX3P/TX3N)	Output	NAP Tx data for lane 3

8.3 Memory map and register definition

There are no control, configuration or status registers within the NAP. All control, configuration and status options for number of lanes and LVDS pads are handled in the NAR_NAPCR which is described in the NAR Chapter.

8.4 Functional description

High-level descriptions of the operation of the NAP and its main functional blocks are given in the following sections.

8.4.1 Data symbol character accumulator

The data symbol character accumulator portion of the NAP for each lane is made up of a symbol character register whose inputs are connected to the 10-bit NAL character bus. The data symbol character accumulator essentially registers a 10-bit character for that lane simultaneous with the rising edge of the NAL clock and presents the 10-bit quantity to the serializer at the appropriate times.

8.4.2 Serializer

The serializer portion of the NAP for each lane takes the data captured by that lane's Data Symbol Character Accumulator and demuxes the data to provide a single serial stream of symbol data to be output by the LVDS LO buffer for that lane at the PHY clock rate.

The following figure shows a high level timing diagram that demonstrates the operation of the serializer for a NAP lane, where S0–S9 represents a 10-bit symbol, S10–S19 represent the next symbol, and so on.

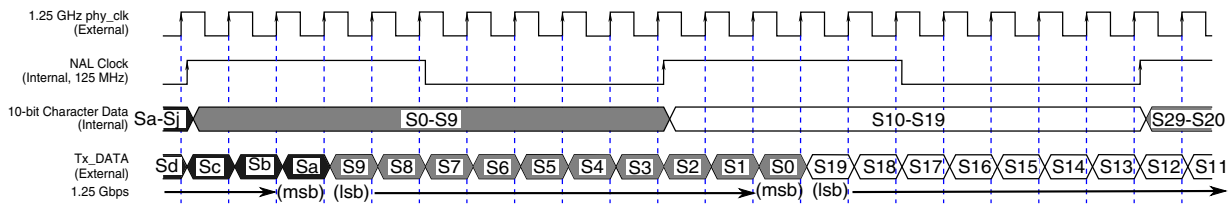


Figure 8-2. Serializer timing diagram

8.4.3 Clock generation and control

The Clock Generation and Control block is responsible for generating the necessary clocks and state signals needed by the rest of the NAP. The Data Symbol Character Accumulators and serializers are the main consumers of the signals generated by the Clock Generation and Control block.

8.5 Initialization information

The NAP does not require initialization other than the Nexus Reset and programming of the NAR_NAPCR to configure the number of lanes and LVDS pads.

Chapter 9
Buddy Device—Nexus Read Write Access (RWA)

9.1 Introduction

The Nexus Read Write Access (RWA) module allows an external tool connected to the Emulation and Debug Device (ED) to read or write the Buddy Die calibration and trace memory via the JTAG interface. All accesses are setup and initiated by the Read/Write Access Control/Status Register (RWCS), as well as the Read/Write Access Address (RWA) and Read/Write Access Data Registers (RWD). The RWA module supports single as well as block reads and writes.

The RWA module is enabled by loading a single instruction (ACCESS_AUX_RWA_BD = 0x23) into the JTAG Instruction Register (IR). After the RWA module has been enabled, access to the RWA registers is enabled by loading an instruction (CORE_ENABLE) into the JTAG IR, as described in Nexus register access via JTAG. The RWA module is disabled when the JTAG state machine initially reaches the Test-Logic-Reset state from any other state. The RWA module is also disabled if a Power-on-Reset (POR) event occurs. When disabled, the RWA module registers are not available for (32-bit or less) reads or writes.

9.2 Register definition

This table details the register map for the RWA module.

Table 9-1. RWA register map

Register	Nexus access opcode	Read/ write	Read address	Write address
Read/Write Access Control/Status (RWCS)	0x7	R/W	0x0E	0x0F
Read/Write Access Address (RWA)	0x9	R/W	0x12	0x13
Read/Write Access Data (RWD)	0xA	R/W	0x14	0x15

9.2.1 Read/write access control/status (RWCS)

The Read/write access control/status register provides control and status for the RWA module as shown in this table.

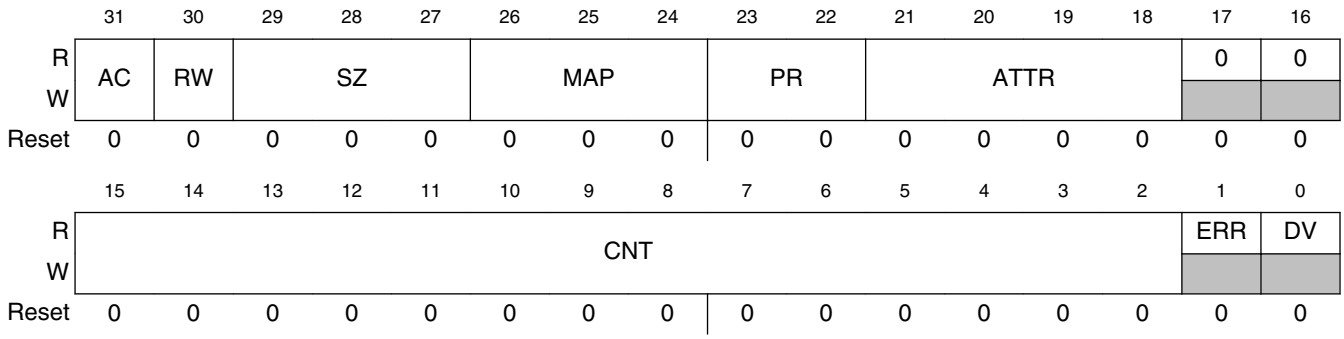


Table 9-2. RWCS fields

Field	Description	
31 AC	Access Control.	
	0	End access/ Access has completed
	1	Start access
30 RW	Read/Write Select.	
	0	Read access
	1	Write access
29:27 SZ	Word Size.	
	000	8-bit (byte)
	001	16-bit (half-word)
	010	32-bit (word)
	011	Reserved (default to word)
	100	Reserved (default to word)
	101	Reserved (default to word)
	110	Reserved (default to word)
	111	Reserved (default to word)
26:24 MAP	MAP Select.	
	000	Primary memory map
	001	Reserved
	010	Reserved
	011	Reserved
	100	Reserved
	101	Reserved
	110	Reserved
	111	Reserved

Table continues on the next page...

Table 9-2. RWCS fields (continued)

Field	Description	
23:22 PR	Read/Write Access Priority.	
	00	Reserved (default to highest priority)
	01	Reserved (default to highest priority)
	10	Reserved (default to highest priority)
	11	Highest access priority
21:18 ATTR	Access Attributes. RWA accesses are run as privileged data non-cacheable accesses by default, and drive the p_d_hprot[5:0] bus access attributes to 6'b000011 accordingly. This field is provided to allow a portion of these default values to be modified when performing read or write accesses using the RWA as shown in Table 9-4. hprot[1] must be set to one.	
	0xxx	Reserved
	1xxx	Reserved
	x0xx	p_d_hprot[4] driven to 0 for accesses
	x1xx	p_d_hprot[4] driven to 1 for accesses
	xx0x	p_d_hprot[3] driven to 0 for accesses
	xx1x	p_d_hprot[3] driven to 1 for accesses
	xxx0	p_d_hprot[2] driven to 0 for accesses
	xxx1	p_d_hprot[2] driven to 1 for accesses
17:16	Reserved for future functionality.	
15:2 CNT	Access Control Count. Number of accesses of word size SZ.	
1 ERR	Read/Write Access Error (see Table 9-3).	
0 DV	Read/Write Access Data Valid (see Table 9-3).	

Table 9-3. Read/Write access status bit encoding

Read action	Write action	ERR	DV
Read access has not completed	Write access completed without error	0	0
Read access error has occurred	Write access error has occurred	1	0
Read access completed without error	Write access has not completed	0	1
Not allowed	Not allowed	1	1

Table 9-4. Protection encoding

p_hprot[4]	p_hprot[3]	p_hprot[2]	Transfer type
0	0	0	Cache-inhibited
0	0	1	Guarded, not cache-inhibited
0	1	0	Reserved
0	1	1	Reserved

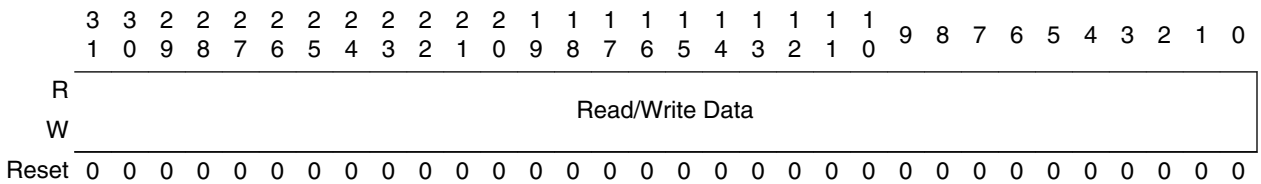
Table continues on the next page...

Table 9-4. Protection encoding (continued)

p_hprot[4]	p_hprot[3]	p_hprot[2]	Transfer type
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Cachable, writethrough
1	1	1	Cachable, writeback

9.2.2 Read/write access data (RWD)

The Read/write access data register (RWD) provides the data to/from memory when initiating a read or a write access.



Read/write accesses to the memory require that the debug firmware properly retrieve/ place the data in the RWD. This table shows the proper placement of data into the RWD.

Table 9-5. RWD data placement for transfers

Transfer size and byte offset	RWA(2:0)	RWCS[SZ]	RWD[1]			
			31:24	23:16	15:8	7:0
Byte	x x x	0 0 0	—	—	—	X
Half	x x 0	0 0 1	—	—	X	X
Word	x 0 0	0 1 0	X	X	X	X

1. X indicates byte lanes with valid data; — indicated bytes lanes with unused data.

This table shows the mapping of RWD bytes to byte lanes of the read and write data buses.

Table 9-6. RWD byte lane mapping

Transfer size and byte offset	RWA(2:0)	RWD[1]			
		31:24	23:16	15:8	7:0
Byte @000	0 0 0	—	—	—	XBAR[7:0]
Byte @001	0 0 1	—	—	—	XBAR[15:8]
Byte @010	0 1 0	—	—	—	XBAR[23:16]

Table continues on the next page...

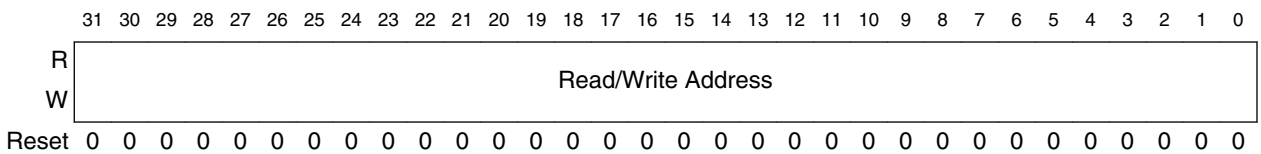
Table 9-6. RWD byte lane mapping (continued)

Transfer size and byte offset	RWA(2:0)	RWD[1]			
		31:24	23:16	15:8	7:0
Byte @011	0 1 1	—	—	—	XBAR[31:24]
Byte @100	1 0 0	—	—	—	XBAR[39:32]
Byte @101	1 0 1	—	—	—	XBAR[47:40]
Byte @110	1 1 0	—	—	—	XBAR[55:48]
Byte @111	1 1 1	—	—	-	XBAR[63:56]
Half @000	0 0 0	—	—	XBAR[15:8]	XBAR[7:0]
Half @010	0 1 0	—-	—	XBAR[31:24]	XBAR[23:16]
Half @100	1 0 0	—-	—	XBAR[47:40]	XBAR[39:32]
Half @110	1 1 0	—	—	XBAR[63:56]	XBAR[55:48]
Word @000	0 0 0	XBAR[31:24]	XBAR[23:16]	XBAR[15:8]	XBAR[7:0]
Word @100	1 0 0	XBAR[63:56]	XBAR[55:48]	XBAR[47:40]	XBAR[39:32]

1. — indicates byte lanes that contain unused data

9.2.3 Read/write access address (RWA)

The Read/write access address register provides the address to be accessed when initiating a read or a write access.



9.3 Nexus register access via JTAG

After the RWA module has been enabled, access to the RWA registers is enabled by loading an instruction (CORE_ENABLE = 0x7C) into the JTAG Instruction Register (IR). Once the CORE_ENABLE instruction has been loaded, the JTAG port allows tool/target communications with all Nexus registers according to the map in Table 9-1.

Reading or writing of a RWA module register then requires two passes through the Data-Scan (DR) path of the JTAG state machine.

1. The first pass through the DR selects the RWA module register to be accessed by providing an index and the direction (read/write) as shown in Table 9-1. This is achieved by loading an 8-bit value into the JTAG data register (DR).

This register has the following format:

(7bits)	(1 bit)
Nexus access opcode	R/W

Table 9-7. JTAG DR field description for Nexus register access

Field	Description	
Nexus Register Index	Selected from values in Table 9-1	
Read/Write (R/W)	0	Read
	1	Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
   a. During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the Capture-DR state.
   b. During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the Update-DR state.

9.4 Memory access examples

The following subsections describe the steps which are required to access memory.

9.4.1 Single write access

1. Initialize the RWA register through the JTAG. Configure as follows:
   - Write Address = 32h'xxxxxxxx (write address)
2. Initialize the RWCS register through the JTAG. Configure the bits as follows:
   - Access Control (AC) = 1b'1 (to indicate start access)
   - Map Select (MAP) = 3b'000 (primary memory map)
   - Access Priority (PR) = 2b'00 (lowest priority)
   - Read/Write (RW) = 1b'1 (write access)

- Word Size (SZ) = 3b'0xx (32-bit, 16-bit, 8-bit)
- Access Count (CNT) = 14h'0000 or 14h'0001(single access)

Note

Access Count (CNT) of 14'h0000 or 14'h0001 performs a single access.

3. Initialize the RWD register through the JTAG. Configure as follows:
   - Write Data = 32h'xxxxxxxx (write data)
4. The RWA module then transfers the data value from the data buffer RWD register to the memory mapped address in the RWA register. When the access has completed without error (ERR = 1'b0), the RWA clears the DV bit in the RWCS register. This indicates that the device is ready for the next access.

9.4.2 Block write access

1. For a block write access, follow Steps 1, 2, and 3 outlined in Single write access to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS register.
2. The RWA module then transfers the first data value from the RWD register to the memory mapped address in the RWA register. When the transfer has completed without error (ERR=1'b0), the address from the RWA register is incremented to the next word size (specified in the SZ field) and the number from the CNT field is decremented. The RWA module then indicates it is ready for the next access.
3. Repeat Step 3 in Single write access until the internal CNT value is zero (0). When this occurs, the DV bit within the RWCS is cleared to indicate the end of the block write access.

Note

The actual RWA register value as well as the CNT field within the RWCS are not changed when executing a block write access. The original values can be read by the external development tool at any time.

9.4.3 Single read access

1. Initialize the RWA register through the JTAG. Configure as follows:
   - Read Address = 32h'xxxxxxxx (read address)
2. Initialize the RWCS register through the JTAG. Configure the bits as follows:
   - Access Control (AC) = 1b'1 (to indicate start access)
   - Map Select (MAP) = 3b'000 (primary memory map)
   - Access Priority (PR) = 2b'00 (lowest priority)
   - Read/Write (RW) = 1b'0 (read access)
   - Word Size (SZ) = 3b'0xx (32-bit, 16-bit, 8-bit)
   - Access Count (CNT) = 14h'0000 or 14h'0001(single access)

Note

Access Count (CNT) of 14'h0000 or 14'h0001 performs a single access.

3. The RWA module then transfers the data to the RWD register. When the transfer is completed without error (ERR = 1'b0), the RWA module sets the DV bit in the RWCS register. This indicates that the RWA module is ready for the next access.
4. The data can then be read from the RWD register through the JTAG.

9.4.4 Block read access

1. For a block read access, follow Steps 1 and 2 outlined in Single read access to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS register.
2. The RWA module then transfers the data to the RWD register. When the transfer has completed without error (ERR = 1'b0), the address from the RWA register is incremented to the next word size (specified in the SZ field) and the number from the CNT field is decremented. The RWA module then indicates that the device is ready for the next access.
3. The data can then be read from the RWD register through JTAG.

4. Repeat Steps 3 and 4 in Single read access until the CNT value is zero (0). When this occurs, the DV bit within the RWCS register is set to indicate the end of the block read access.

Note

The data values must be shifted out 32-bits at a time LSB first. The actual RWA value as well as the CNT field within the RWCS register are not changed when executing a block read access. The original values can be read by the external development tool at any time.

Appendix A
Release Notes for Revision 6

A.1 General changes

- No general changes.

A.2 Emulation and Debug Device Intro changes

- In section Block diagram, figure Figure 1-1 :
  - Around the middle, added IRC clock.
  - On the lower left, corrected "Aurora Port Output lanes plus LVDS dock In" to "Nexus Aurora Port Output lanes plus LVDS clock in".
  - On the middle right, corrected "(LVDS or CMOS)" to "(CMOS)".

A.3 Buddy Device—Crossbar and RAM changes

- No substantial content changes

A.4 Buddy Device SIUL changes

- No substantial content changes

A.5 DCI changes

- No substantial content changes

A.6 JTAGC module changes

- No substantial content changes

A.7 NAR changes

- No substantial content changes

A.8 NAL changes

- No substantial content changes

A.9 Nexus Aurora Phy changes

- No substantial content changes

A.10 RWA changes

- No substantial content changes

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. Freescale reserves the right to make changes without further notice to any products herein.

Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Freescale, the Freescale logo, Qorivva, SafeAssure, and the SafeAssure logo are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. All other product or service names are the property of their respective owners. The Power Architecture and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2012–2016 Freescale Semiconductor, Inc.

Document Number MPC5746RBDRM
Revision 6, 03/2016

